



- [Home](#)  
[Report home for your next assignment](#)
- [Training](#)



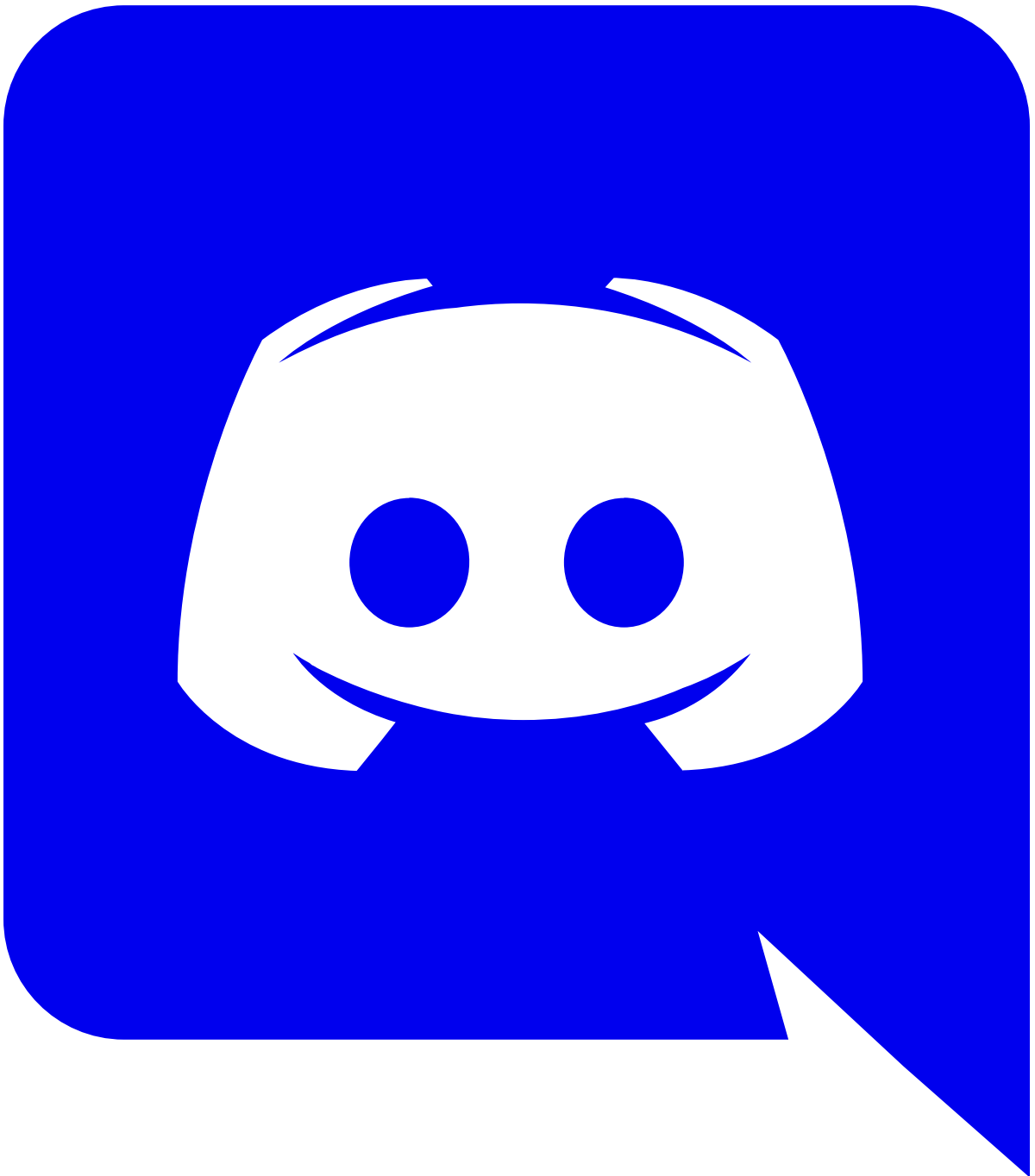
- [Practice](#)  
[Complete challenging \*\*Kata\*\* to earn honor and ranks. Re-train to hone technique](#)



- [Freestyle Sparring](#)  
[Take turns remixing and refactoring others code through Kumite](#)
- Community



- [Leaderboards](#)  
[Achieve honor and move up the global leaderboards](#)

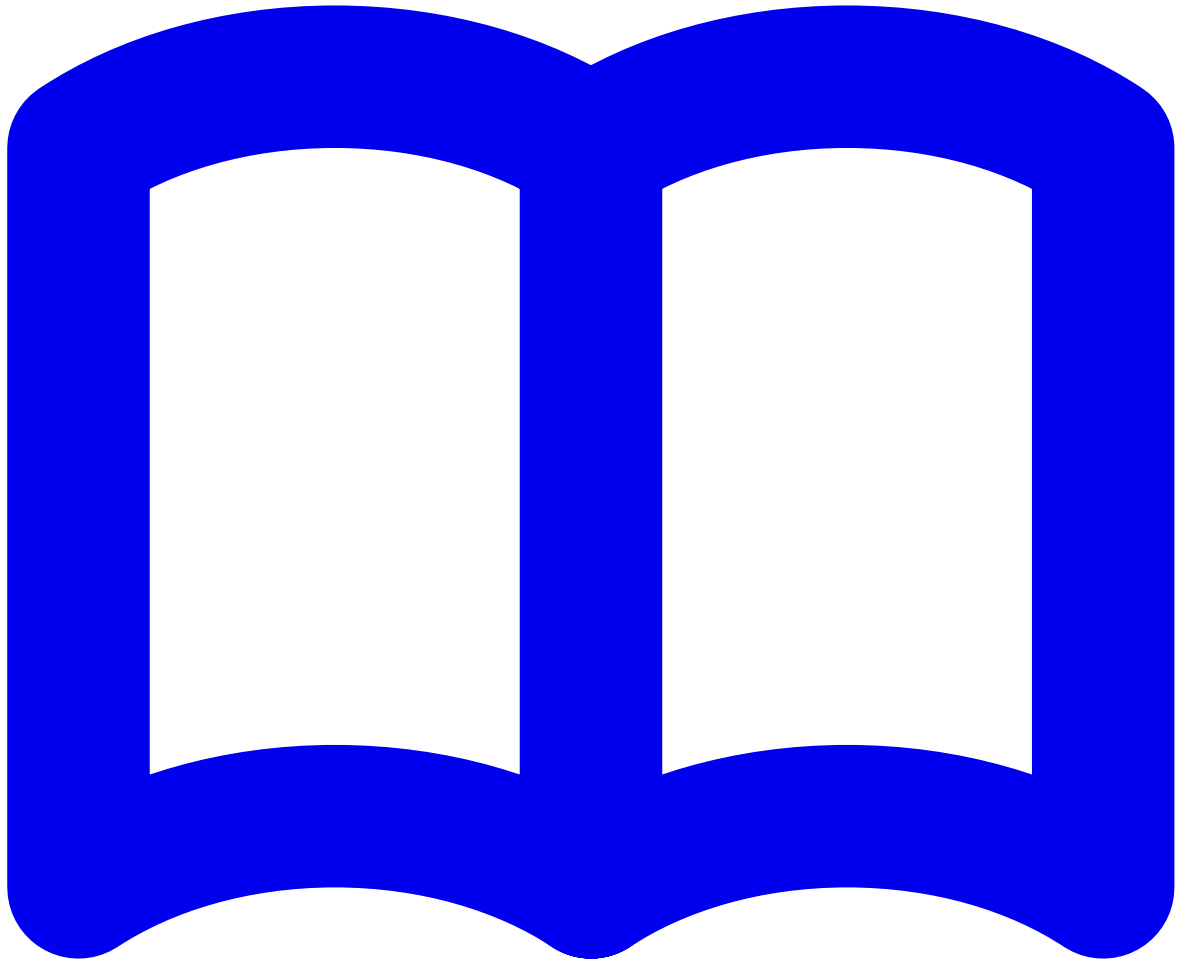


• [Chat](#)

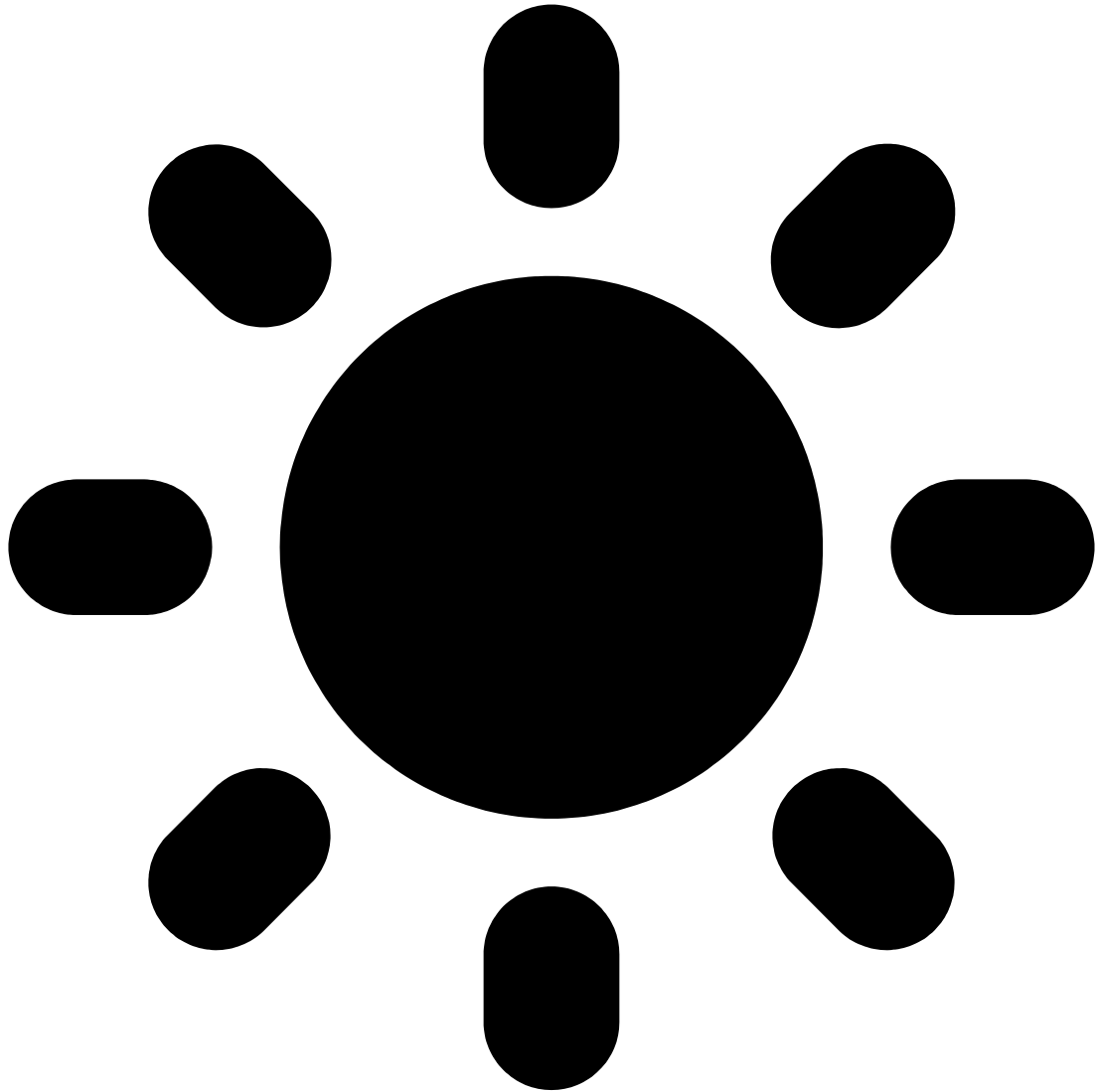
[Join our \*\*Discord\*\* server and chat with your fellow code warriors](#)



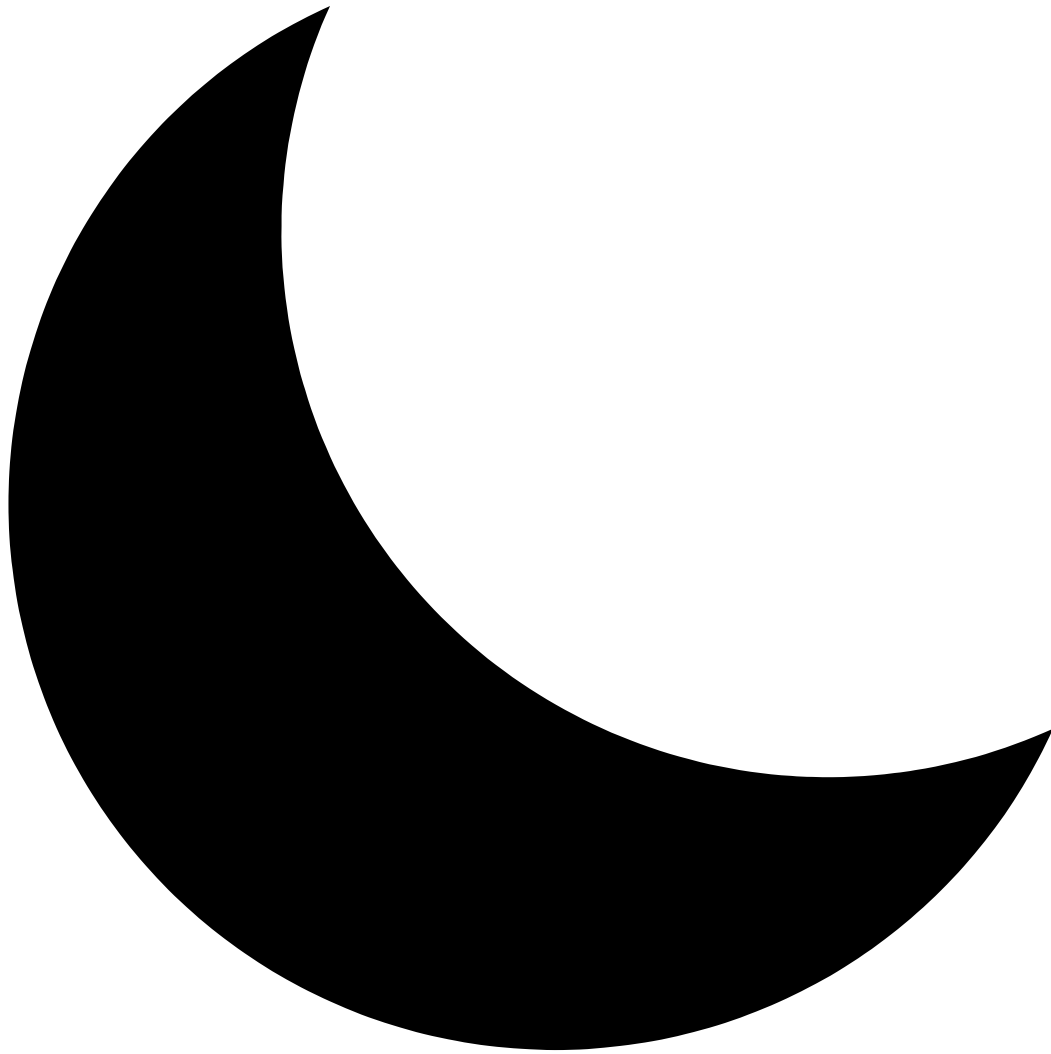
- [Discussions](#)  
[View our \*\*Github Discussions\*\* board to discuss general Codewars topics](#)
- [About](#)



- [Docs](#)  
[Learn about all of the different aspects of Codewars](#)



.



- - [6 kyu](#)  
[Missing Angle](#)
  - [7 kyu](#)  
[Rotate for a Max](#)
  - [6 kyu](#)  
[Union of Intervals](#)
  - [7 kyu](#)  
[Most valuable character](#)
  - [7 kyu](#)  
[Single digit](#)
  - [6 kyu](#)  
[Maximum Product of Parts](#)
  - [7 kyu](#)  
[isEven? - Bitwise Series](#)
  - [6 kyu](#)  
[Sort the number sequence](#)





[Filter](#)

[Mutate My Strings](#)  
4000+ Honor: You now have the ability to be a kata contributor!

11 days ago

[ion marks series #9: Remove or add a exclamation mark at the end of words of the sentence](#)

yu

[or Counting Total Occurence Of Specific Digits](#)

3 kyu

4,017

7 kyu

- [View Profile](#)
- [Sorting Arrays](#)
- [Account Settings](#)
- [Training Setup](#)
- [Upgrade to Red](#)
- [New Kata](#)
- [Unamenable Magic #2 - True for Any?](#)
- [New Kumite](#)
- [Sign out](#)

[ole Magic #27 - The Least and the Greatest](#)

yu

[ole Magic #12 - Find Minimum Value by Comparison](#)

8 kyu

3 kyu andreapt82 4,017

3 kyu andreapt82 4,017

**Name:** André Terceiro  
**Clan:** None  
**Skills:** ruby, php, python, javascript, .net, groovy  
**Member Since:** Jun 2015  
**Last Seen:** Apr 2024  
**Profiles:**

[99 Problems, #5: reversing a list](#)

7 kyu

[Guess My Number](#)

**Following:** 619

**Followers:** 594

**Allies:** 592

[View Profile](#)

Earn extra honor and gain new allies!

[Emptying the Values](#)

Honor is earned for each new codewarrior who joins.

Learn more

Use the referral URL to invite your friends to Codewars. Once they complete the initiation and confirm their account, you will be awarded honor.

[Simple reversal game](#)

[www.codewars.com/r/ZpBPkg](http://www.codewars.com/r/ZpBPkg)

Your Referrals [Sequence classifier](#)

edu\_ktemail un-confirmed [Beta](#)

[Sort by number](#)



[Linked Lists - Length & Count](#)

[Developer Productivity: A guide to finding flow](#)

[Find out to harness your flow for more higher productivity.](#)

- [Stats](#)
- [Square Pi's](#)
- [Kata](#)
- [Solutions](#)
- [Translations](#)
- [Collection](#)
- [Pi's Meter Sum](#)
- [Kumite](#)
- [Social](#)
- [Discourse](#)
- [Next Featured Number Higher than a Given Value](#)
- [Completed \(938\)](#)

- [Unfinished](#) [8 kyu](#)
- [Obsolete](#)

7 kyu [Simple validation of a username with regex](#)

[Wealth equality, finally!](#)  
 ◦ [Beta](#)

Python: [Finding people with a speeding record](#)

```
def redistribute_wealth(wealth):
    total_elements = len(wealth)
    mean = sum(wealth) / total_elements
    for index, item in enumerate(wealth):
        wealth[index] = mean
```

[Return a sorted list of objects](#)

- [13 minutes ago](#) [20 kyu](#)
- [Refactor](#)
- [Discuss](#) [Product of the main diagonal of a square matrix.](#)

8 kyu ◦ [7 kyu](#)

[I love you, a little, a lot, passionately ... not at all](#)  
[Simple Fun #144: Distinct Digit Year](#)

Ruby: ◦ [6 kyu](#)

```
def how_much_i_love_you(nb_petals)
    puts nb_petals
    puts nb_petals % 6
    rounded_petals = (nb_petals % 6)
    while rounded_petals > 6
        rounded_petals = rounded_petals
    end
    ◦ 6 kyu
    if rounded_petals == 1
        return "I love you"
    elsif rounded_petals == 2
        return "a little"
    elsif rounded_petals == 3
        return "a lot"
    elsif rounded_petals == 4
        return "passionately"
    elsif rounded_petals == 5
        return "madly"
    elsif rounded_petals == 6 or rounded_petals == 0
        return "not at all"
    end
end
◦ 6 kyu
```

- [2 days ago](#) [The Office V - Find a Chair](#)
- [Refactor](#)
- [Discuss](#) [6 kyu](#)

6 kyu [Add commas to a number](#)

[WeIrD StRiNg CaSe](#)  
 ◦ [7 kyu](#)

Ruby: [Users with More Than 2 Videos Watched](#)

```
def weirdcase string
    ret = ""
    cont = 0
    string.each_char { |char|
        if char == " "
            ret = ret + " "
            cont = 0
        else
            if cont % 2 == 0
                ret = ret + char.upcase
            else
                ret = ret + char.downcase
            end
            cont = cont + 1
        end
    }
    ret
end
◦ 7 kyu
```

- [3 days ago](#) [7 kyu](#)
- [Refactor](#) [Minimum Perimeter of a Rectangle](#)
- [Discuss](#)

◦ [7 kyu](#)

7 kyu [Array comparator Eligibility: part 1](#)

Ruby: ◦ [6 kyu](#)

```
def match_arrays(v, r)
    count = 0
    v.each { |item|
        count = count + 1 if r.include? item
    }
    count
end
◦ 6 kyu
```

- [8 days ago](#) [Two Arrays by Id](#)
- [Refactor](#)

- [Discuss](#) [6 kyu](#)

7 kyu

[From A to Z Youngest Team Members](#)JavaScript: [Beta](#)[Car Depreciation](#)

```
function gimmeTheLetters(sp) {
  let start = sp.charCodeAt(0);
  let end = sp.charCodeAt(2);
  let ret = "";
  let char = Sort rectangles and circles by area II
```

```
for (char = start; char <= end ; char++) {
  ret += String.fromCharCode(char);
}
```

return ret; [5 kyu](#)

- 11 days ago [Calculate age in years](#)
- [Refactor](#) [7 kyu](#)
- [Discuss](#)

7 kyu [Sorted hashes](#)[Ordering the words!](#)Ruby: [Training JS #18: Methods of String object--concat\(\) split\(\) and its good friend join\(\)](#)

```
def order2word(s) 7 kyu
  return "Invalid String!" if s.nil? or s.empty?
  s.split(" ").sort{|a,b| a.length < b.length}.join(" ")
end
```

- 12 days ago [8 kyu](#)
- [Refactor](#) [Count the number of cubes with paint on](#)
- [Discuss](#)

7 kyu [7 kyu](#)[Integer Difference of Phone Numbers!](#)Ruby: [7 kyu](#)[Histogram - H1](#)

```
def int_diff(arr, n)
  r = 0
  arr.each_with_index {|i, index1|
    arr.each_with_index {|j, index2|
      break if index1 == index2
      next if index1 == index2
      if i - j == n || j - i == n
        r = r + 1
      end
    }
  }
  r
end
```

- 13 days ago [6 kyu](#)
- [Refactor](#) [Backwards Read Primes](#)
- [Discuss](#) [More Zeros than Ones](#)

7 kyu

[Without the letter 'e'](#)JavaScript: [Calculate Serve Point Win Percentage in Tennis](#)

```
function findE(str){
  if (str == null) return null;
  if (str.trim() === "") return "";
  let totalMausculos = str.split("E").length - 1;
  let totalMinusculos = str.split("e").length - 1;
  let total = totalMausculos + totalMinusculos;
  if (total < 1) return "There is no 'e'.";
  return String(total);
}
```

- 7 years ago [7 kyu](#)
- [Refactor](#) [Average overlapping strings](#)

7 kyu [7 kyu](#)[Simple Fun #201 First Reverse Twice of a word in a string!](#)Ruby: [7 kyu](#)

```
def first_reverse_try(arr)
  return [] if arr.length == 0
  tmp = arr[0]
  arr[0] = arr[-1]
  arr[-1] = tmp
  arr
end
```

- 15 days ago [Beta](#)
- [Refactor](#) [Check my Sum](#)

- [Discuss](#) [7 kyu](#)

7 kyu

[Product of List Sums](#)Ruby: [7 kyu](#)

```
def max_product(a)
  a.max(2).reduce(*)
end
```

- 16 days ago [Tower Bouncing](#)
- [Refactor](#) [5 kyu](#)
- [Discuss](#)

7 kyu [Find the unique string](#)[Sum of a nested list](#) [7 kyu](#)Ruby: [regex validation of 24 hours time.](#)

```
def sum_nested(lst)
  lst.flatten.sum
end
```

- 16 days ago [Beta](#)
- [Refactor](#)
- [Discuss](#) [Approximate a number as fraction](#)

7 kyu [isEven? - Bitwise Series](#)[Range Extraction](#)JavaScript: [7 kyu](#)

```
var isEven = function (n) { //if n is even return true, otherwise, return false
  let binaryNumber = (n >>> 0).toString(2);
  if (binaryNumber[binaryNumber.length - 1] == "0") {
    return true;
  }
  return false;
}
```

- 16 days ago [7 kyu](#)
- [Refactor](#) [You are a Cube!](#)
- [Discuss](#)

7 kyu [Change two-dimensional array.](#)JavaScript: [7 kyu](#)

```
function matrix(array) {
  let i = 0;
  while (i < array.length) {
    if (array[i][i] == 1) {
      array[i][i] = 0;
    } else {
      array[i][i] = 1;
    }
    i = i + 1;
  }
  return array;
}
```

- 17 days ago [7 kyu](#)
- [Refactor](#)
- [Discuss](#) [Is valid identifier?](#)

7 kyu [Unique Sum](#)[Sort odd and even numbers in different order](#)Ruby: [7 kyu](#)

```
def unique_sum(lst)
  ret = lst.uniq.sum
  lst.count == 0 ? nil : ret
end
```

- last month [Decimal Time Conversion](#)
- [Refactor](#) [6 kyu](#)
- [Discuss](#)

7 kyu [Same matrix \(2 \\* 2\).](#)[Count the Ones](#) [6 kyu](#)Ruby: [Verify if it's valid \(n x n\) Magic Square with custom rules](#)

```
def hamming_weight(x)
  x.to_s(2).gsub(/0/, "").length
end
```

- last month
- [Refactor](#)
- [Discuss](#)

7 kyu

[Complete The Pattern #6 - Odd Ladder](#)

Ruby:

```
def pattern(n)
  ret = ""
  i = 1
  while i <= n
    ret = ret + i.to_s * i + "\n"
    i = i + 2
  end

  ret[0..-2]
end
```

- last month
- [Refactor](#)
- [Discuss](#)

7 kyu

[Complete The Pattern #5 - Even Ladder](#)

Ruby:

```
def pattern(n)
  return "" if n <= 1

  ret = ""
  i = 0

  while i * 2 <= n
    number = i * 2
    ret += number.to_s * number + "\n"
    i = i + 1
    puts i
  end

  ret[1..999999][0..-2]
end
```

- last month
- [Refactor](#)
- [Discuss](#)

7 kyu

[All Star Code Challenge #1](#)

Ruby:

```
def sum_ppg(player_one, player_two)
  player_one['ppg'] + player_two['ppg']
end
```

- last month
- [Refactor](#)
- [Discuss](#)

7 kyu

[Email Address Obfuscator](#)

Ruby:

```
def obfuscate(email)
  email = email.gsub /\@/, " [at] "
  email.gsub /\./, " [dot] "
end
```

- last month
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sentence to words](#)

JavaScript:

```
function splitSentence(s) {
  return s.split(" ");
}
```

- last month
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sort Numbers](#)

Ruby:

```
def solution(nums)
  return [] if nums.nil?
  nums.sort()
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Localize The Barycenter of a Triangle](#)

Ruby:

```
def bar_triang(p1,p2,p3)
  [((p1[0] + p2[0] + p3[0]) / 3.0).round(4), ((p1[1] + p2[1] + p3[1]) / 3.0).round(4)]
end
```

- last month
- [Refactor](#)
- [Discuss](#)

7 kyu

[Slice the middle of a list backwards](#)

JavaScript:

```
function reverseMiddle(array) {
  const length = array.length;

  let ret = [];

  for (let i in array) {
    if (length % 2 == 1) {
      if ((i == (length / 2) + 0.5) || (i == (length / 2) - 0.5) || (i == (length / 2) - 1.5)) {
        ret.unshift(array[i]);
      }
    } else {
      if ((i == length / 2) || (i == (length / 2) - 1)) {
        ret.unshift(array[i]);
      }
    }
  }

  return ret;
}
```

- last month
- [Refactor](#)
- [Discuss](#)

7 kyu

[Resistor Color Codes](#)

Ruby:

```
def decode_resistor_colors(bands)
  bands = bands.split(" ")
  numbers_bands = bands_to_numbers bands
  (transform_to_letters numbers_bands) + " ohms, " + numbers_bands[3]
end
```

```
def transform_to_letters number_bands
  start = (number_bands[0].to_s + number_bands[1].to_s)
  ending = (number_bands[2].to_i)
```

```
  start = start.to_f
  total = start * ending

  if (total >= 1000000)
    total = (total / 1000000)
    if total % 1 > 0
      total = total.to_f
    else
      total = total.to_i
    end
    total = total.to_s + "M"
  elsif (total >= 1000)
    total = (total / 1000)
    if total % 1 > 0
      total = total.to_f
    else
      total = total.to_i
    end
    total = total.to_s + "k"
  else
    if total % 1 > 0
      total = total.to_f
    else
      total = total.to_i
    end
    total = total.to_s
  end

  total
end
```

```

def bands_to_numbers bands
  bands[0] = get_color_value(bands[0])
  bands[1] = get_color_value(bands[1])
  bands[2] = 10 ** get_color_value(bands[2])
  bands[3] = get_color_pct(bands[3])
  bands
end

def get_color_pct color
  if color == "gold"
    return "5%"
  elsif color == "silver"
    return "10%"
  end

  return "20%"
end

def get_color_value(color)
  if color == "black"
    return 0
  elsif color == "brown"
    return 1
  elsif color == "red"
    return 2
  elsif color == "orange"
    return 3
  elsif color == "yellow"
    return 4
  elsif color == "green"
    return 5
  elsif color == "blue"
    return 6
  elsif color == "violet"
    return 7
  elsif color == "gray"
    return 8
  elsif color == "white"
    return 9
  end
end

```

- last month
- [Refactor](#)
- [Discuss](#)

7 kyu

[Is it a letter?](#)

Python:

```

import re
def is_it_letter(s):
  return re.match('[A-Za-z]*', s)[0] == s

```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Find the missing letter](#)

PHP:

```

function find_missing_letter(array $array): string {
  $letraEsperada = $array[0];
  foreach($array as $letra) {
    if ($letraEsperada != $letra) {
      return $letraEsperada;
    }
    $letraEsperada = chr(ord($letra) + 1);
  }
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```

function findMissingLetter(array)
{
  let ordCaracterAnterior = null;
  let ordCaracterAtual = null
  for (let caracterAtual of array) {
    ordCaracterAtual = caracterAtual.charCodeAt(0);
    if (ordCaracterAnterior != null && ordCaracterAtual > ordCaracterAnterior + 1) {
      return String.fromCharCode(ordCaracterAnterior + 1);
    }
    ordCaracterAnterior = ordCaracterAtual;
  }

  return null;
}

```

- 7 years ago
- [Refactor](#)

- [Discuss](#)

8 kyu

[Invert values](#)

Ruby:

```
def invert(list)
  r = []
  list.each do |l|
    r.push(l * -1)
  end
  r
end
```

end

- 2 months ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function invert($a): array {
    $r = [];
    for ($i = 0; $i < count($a); $i++) {
        array_push($r, -1 * $a[$i]);
    }
    var_dump($r[1]);
    return empty($r) ? [] : $r;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[ASCII letters from Number](#)

Ruby:

```
def convert(number)
  current_position = 0
  ret = ""

  while current_position < number.length
    part = number[(current_position) .. (current_position + 1)]
    ret += part.to_i.chr
    current_position += 2
  end

  ret
end
```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Valid Phone Number](#)

Ruby:

```
def validPhoneNumber(phoneNumber)
  return false if phoneNumber.match /[a-zA-Z]+/
  (phoneNumber.match /\{1}[0-9]{3}\-[0-9]{4}/).nil? ? false : true
end
```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

```
def validPhoneNumber(phoneNumber)
  return false if phoneNumber.match /abc/
  (phoneNumber.match /\{1}[0-9]{3}\-[0-9]{4}/).nil? ? false : true
end
```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Delete occurrences of an element if it occurs more than n times](#)

Ruby:

```
def delete_nth(order,max_e)
  count = {}
  ret = []

  order.each { |item|
    count[item] = 0 if count[item] == nil
    count[item] = count[item] + 1
  }
  ret
```



```

    if count[item] <= max_e
      ret.push item
    end
  }

  ret
end

```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

Retired

[Non-consecutive Pairs](#)

Ruby:

```

def non_consec_pairs(my_array)
  ret = []
  my_array.each_with_index { |item1, index1|
    my_array.each_with_index { |item2, index2|
      item1_eval = (item1.is_a?(Integer) || item1.is_a?(Float)) ? item1.to_s : item1
      item2_eval = (item2.is_a?(Integer) || item2.is_a?(Float)) ? item2.to_s : item2
      next if index1 >= index2
      if item1_eval.ord - item2_eval.ord > 1 or item2_eval.ord - item1_eval.ord > 1 || ((item1.is_a?(Float) && item2.is_a?(Float) && (item1 - item2 >
        ret.push [item1, item2]).sort!
      end
    }
  }
  ret
end

```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Leap Years](#)

Ruby:

```

def is_leap_year(year)
  begin
    Date.parse "#{year}-02-29"
    return true
  rescue
    return false
  end
end

```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

Beta

[Convert integer to binary](#)

Ruby:

```

def n_to_bits(n, min_length = 1)
  n.to_s(2).rjust min_length, "0"
end

```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Every possible sum of two digits](#)

Ruby:

```

def digits(num)
  arr = num.to_s.split("")

  ret = []
  arr.each_with_index { |element1, index1|
    arr.each_with_index { |element2, index2|
      next if index1 >= index2
      ret.push element1.to_i + element2.to_i
    }
  }

  ret
end

```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[ORing arrays](#)

Ruby:

```
def or_arrays(*args)
  arr1 = args[0]
  arr2 = args[1]
  default_value = args[2].nil? ? 0 : args[2]

  array_to_iterate = arr1.size > arr2.size ? arr1 : arr2

  array_to_iterate.each_with_index { |item, index|
    a = arr1[index].nil? ? default_value : arr1[index]
    b = arr2[index].nil? ? default_value : arr2[index]
    array_to_iterate[index] = a | b
  }
end
```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[The most common letter](#)

JavaScript:

```
function replaceCommon(string, letter) {
  let counter = {};
  for (let i = 0; i < string.length; i++) {
    if (string[i] !== ' ') {
      if (counter[string[i]] == null) {
        counter[string[i]] = 1;
      } else {
        counter[string[i]] = counter[string[i]] + 1;
      }
    }
  }

  let letterMostFrequent = '';
  let countMostFrequent = 0;

  for (let _letter in counter) {
    if (counter[_letter] > countMostFrequent) {
      letterMostFrequent = _letter;
      countMostFrequent = counter[_letter];
    }
  }

  return string.replaceAll(letterMostFrequent, letter);
}
```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Flick Switch](#)

JavaScript:

```
function flickSwitch(arr){
  let ret = [];
  let retItem = true;

  for (let item of arr) {
    if (item == "flick") {
      retItem = !retItem;
    }
    ret.push(retItem);
  }

  return ret;
}
```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Difference of 2](#)

Ruby:

```
def twos_difference(lst)
  ret = []
  for i in lst
    for j in lst
      next if i == j
      if i - j == 2
        ret.push [i, j].sort!
      end
      j = j + 1
    end
    i = i + 1
  end
  ret.sort!
end
```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

Beta

[Where Are My Glasses?](#)

Python:

```
import re

def find_glasses(lst):
    for index, item in enumerate(lst):
        if (re.search("0\\-+0", item) is not None):
            return index
```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Is your period late?](#)

JavaScript:

```
function periodIsLate(last, today, cycleLength) {
    last.setDate(last.getDate() + cycleLength);
    return last < today;
}
```

- 6 months ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def period_is_late(last, today, cycle_length)
    last + cycle_length < today
end
```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[All or Nothing](#)

JavaScript:

```
function possiblyPerfect(key, answers) {
    let wrong = 0;
    let contUnderline = 0;

    for (i in key) {
        if (key[i] != "_" && key[i] != answers[i]) {
            wrong = wrong + 1;
        }

        if (key[i] == "_") {
            contUnderline = contUnderline + 1;
        }
    }

    return wrong == 0 || wrong == key.length - contUnderline;
}
```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Difference Of Squares](#)

Ruby:

```
def difference_of_squares(n)
    ns = (0..n).sum
    n2 = ns ** 2
    total = 0

    (0..n).each {|i|
        total += i ** 2
    }

    n2 - total
end
```

- 4 months ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function difference_of_squares(int $n): int {
    $sum = 0;
```

```

    $i = 0;
    while ($i <= $n) {
        $sum += $i;
        $i++;
    }

    $n2 = pow($sum, 2);

    $total = 0;
    $i = 0;

    while ($i <= $n) {
        $total += pow($i, 2);
        $i++;
    }

    return $n2 - $total;
}

```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Building Strings From a Hash](#)

Ruby:

```

def solution(pairs)
  ret = ""
  pairs.each {|key, value|
    ret += key.to_s + " = " + value.to_s + ","
  }
  ret[0..-2]
end

```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```

function solution(pairs) {
  let ret = '';
  for (let i in pairs) {
    ret += i + " = " + pairs[i] + ","
  }
  return ret.substring(0, ret.length - 1);
}

```

- 4 months ago
- [Refactor](#)
- [Discuss](#)

4 kyu

[Strip Comments](#)

JavaScript:

```

function solution(input, markers){
  var linhas = input.split("\n");
  retorno = [];
  for (let linha of linhas) {
    linha = linha.split(/[^\!$]/);
    retorno.push(linha[0].trim()+"\n");
  }
  retorno = retorno.join('');
  return retorno.substr(0,retorno.length-1);
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```

def solution(input, markers)
  linhas = input.split("\n");
  retorno = [];

  for linha in linhas
    linha = linha.split(/[^\!@-\$%]/)[0].strip
    retorno.push(linha + "\n")
  end
  (retorno.join " ")[0..-2]
end

```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[isReallyNaN](#)

JavaScript:

```
const isReallyNaN = (n) => {
  if (n === undefined || typeof n == "string" || typeof n == "object" || typeof n == "function") {
    return false;
  }
  return isNaN(n);
};
```

- 3 months ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export function isReallyNaN(n: any): boolean {
  if (n === undefined || typeof n == "string" || typeof n == "object" || typeof n == "function") {
    return false;
  }
  return isNaN(n);
};
```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Reverse and Invert](#)

Ruby:

```
def reverse_invert(array)
  ret = []

  array.each {|item|
    if item.is_a? Integer
      item = item.to_s + ""

      if item[0] == "-"
        item = "-" + item[1..99].reverse
      else
        item.reverse!
      end

      item = item.to_i
      item = item * -1
      ret.push item
    end
  }

  ret
end
```

- 2 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Create Phone Number](#)

Ruby:

```
def create_phone_number(numbers)
  n = numbers
  "#{n[0]}#{n[1]}#{n[2]} #{n[3]}#{n[4]}#{n[5]}-#{n[6]}#{n[7]}#{n[8]}#{n[9]}"
end
```

- 2 months ago
- [Refactor](#)

```
def createPhoneNumber(numbers)
  "#{numbers[0..2].join} #{numbers[3..5].join}-#{numbers[6..10].join}"
end
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function createPhoneNumber(numbers){
  return `(${numbers.slice(0,3).join('')} ${numbers.slice(3,6).join('')}-${numbers.slice(6,10).join('')}`;
}
```

- 7 years ago
- [Refactor](#)

```
function createPhoneNumber(numbers){
  return `(${numbers.slice(0,3).join('')} ${numbers.slice(3,6).join('')}-${numbers.slice(6,10).join('')}`;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function createPhoneNumber($numbersArray) {
    return "(" . implode(array_slice($numbersArray,0,3),'') . ") " . implode(array_slice($numbersArray,3,3),'') . "-" . implode(array_slice($numbersArr
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

```
function createPhoneNumber($numbersArray) {
    return preg_replace('/^\d{3}\d{3}\d{4}$/', '($1) $2-$3', implode("", $numbersArray));
}
```

- 2 years ago
- [Refactor](#)

Groovy:

```
class Kata {
    static String createPhoneNumber(numbers){
        "(" + numbers[0] + numbers[1] + numbers[2] + ")" + numbers[3] + numbers[4] + numbers[5] + "-" + numbers[6] + numbers[7] + numbers[8] + number
    }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Easy SQL: Rounding Decimals](#)

SQL:

```
select floor(number1) as number1, ceil(number2) as number2 from decimals
```

- 3 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Responsible Drinking](#)

Ruby:

```
def hydrate(s)
    total = 0
    number_of_drinks = s.gsub!(/[a-zA-Z]+/, '')

    number_of_drinks.each_char {|char|
        total = total + char.to_i
    }

    if total == 1
        ret = total.to_s + " glass of water"
    else
        ret = total.to_s + " glasses of water"
    end

    ret
end
```

- 3 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Bit Counting](#)

Ruby:

```
def count_bits(n)
    n.to_s(2).gsub(/0+/, "").length
end
```

- 3 months ago
- [Refactor](#)
- [Discuss](#)

```
def count_bits(n)
    total = 0
    ("%b" % n).each_char {|i| total = total + i.to_i}
    total
end
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Beta

[Sum Only Numbers](#)

JavaScript:

```
function sumNumbers() {
  let sum = 0
  for (let i of arguments) {
    if (typeof(i) == 'number') {
      sum = sum + i;
    }
  }

  return sum;
}
```

- 3 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[How many consecutive numbers are needed?](#)

Ruby:

```
def consecutive(arr)
  return 0 if arr.empty?
  consecutive_number_needed = 0
  arr.sort!
  i = arr[0]

  while i < arr[-1]
    i = i + 1
    consecutive_number_needed = consecutive_number_needed + 1 unless arr.include? i
  end

  consecutive_number_needed
end
```

- 3 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Data Reverse](#)

Ruby:

```
def data_reverse(data)
  slots = []

  data.each_with_index{ |data, index|
    slots[index / 8] = [] if slots[index / 8].nil?
    slots[index / 8].push data
    puts index / 8
  }

  slots.reverse.flatten
end
```

- 3 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Parallel resistors](#)

JavaScript:

```
function resistor_parallel(){
  let rInvertedSum = 0;

  for (const argument of arguments) {
    rInvertedSum += 1/argument;
  }

  return 1/rInvertedSum;
}
```

- 4 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[String to list of integers.](#)

Ruby:

```
def string_to_int_list(s)
  ret = []
  s.split(',').each{ |number_string|
    ret.push number_string.to_i if !number_string.empty?
  }
  ret
end
```

- 5 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Is There an Odd Bit?](#)

Ruby:

```
def any_odd?(x)
  ret = false
  x_string = x.to_s(2)

  x_string.reverse.split("").each_with_index {|digit, index|
    next if (index) % 2 == 0

    if digit == "1"
      ret = true
      break
    end
  }
  ret
end
```

- 5 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Reversing Fun](#)

Ruby:

```
def reverse_fun(n)
  cont = 0
  ret = n.reverse
  while cont < n.length
    ret = ret[0..cont] + ret[(cont + 1)..n.length].reverse
    cont = cont + 1
  end
  ret
end
```

- 5 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Neutralisation](#)

JavaScript:

```
function neutralise(s1, s2) {
  let ret = ""
  for (var i in s1) {
    if (s1[i] == "+" && s2[i] == "+") {
      ret = ret + "+";
    } else if (s1[i] == "-" && s2[i] == "-") {
      ret = ret + "-";
    } else {
      ret = ret + "0";
    }
  }
  return ret;
}
```

- 5 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Arithmetic List!](#)

Ruby:

```
def seqlist(first,c,l)
  current = first
  ret = []

  while true
    if ret.length == l
      return ret
    end
    ret.push current
    current += c
  end
end
```

- 5 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Closest to Zero](#)

Ruby:



```
def closest(arr)
  ret = 1000000000
  count = 0

  arr.each{ |item|
    if ((item != ret) && ((item.abs == ret) || (item == ret.abs)))
      count = count + 1
      puts "item: #{item} "
      puts count
    end

    if item.abs < ret.abs
      ret = item
      count = 1
      puts count
    end
  }

  return nil if count > 1
  return ret
end
```

- 5 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Only one](#)

JavaScript:

```
function onlyOne() {
  let countArguments = 0

  for (const argument of arguments) {
    if (argument == true) {
      countArguments++
    }

    if (countArguments > 1) {
      break;
    }
  }

  return countArguments == 1
}
```

- 5 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[ONE ONE One one](#)

JavaScript:

```
function consecutiveOnes(nums) {
  let max = 0;
  let count = 0;

  for (let num of nums) {
    if (num == 1) {
      count = count + 1
      if (count > max) {
        max = count;
      }
    } else {
      count = 0;
    }
  }

  return max;
};
```

- 6 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[lucky number](#)

Ruby:

```
def is_lucky(n)
  n_sum = 0
  n.to_s.split("").each{|i| n_sum += i.to_i }
  return true if n_sum == "0"
  return true if n_sum % 9.0 == 0
  false
end
```

- 6 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Inspiring Strings](#)

Ruby:

```
def longest_word(string_of_words)
  major_length = 0
  ret = ""

  string_of_words.split(" ").each{ |word|
    if word.length >= major_length
      major_length = word.length
      ret = word
    end
  }

  ret
end
```

- 6 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Consecutive items](#)

Ruby:

```
def consecutive(arr, a, b)
  first_index = arr.index(a)
  second_index = arr.index(b)
  first_index + 1 == second_index || second_index + 1 == first_index
end
```

- 6 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Return String of First Characters](#)

Ruby:

```
def make_string(s)
  ret = ""

  s.split(" ").each{ |substring|
    ret += substring[0]
  }

  ret
end
```

- 6 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Simple Fun #144: Distinct Digit Year](#)

Ruby:

```
def distinct_digit_year(year)
  year = year + 1

  while true
    array_string_digits = year.to_s.split('')

    uniq_array_string_digits = array_string_digits.uniq

    return year if array_string_digits.length == uniq_array_string_digits.length

    year = year + 1
  end
end
```

- 6 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Even or Odd](#)

Ruby:

```
def even_or_odd(number)
  (number) % 2 == 0 ? "Even" : "Odd"
end
```

- 6 months ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function even_or_odd(number) {
  if (typeof(number) != "number") {
```

```

    return null;
  }

  if (Math.abs(number % 2) == 1 ) {
    return "Odd";
  }

  return "Even";
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

```

function even_or_odd(number) {
  if (number & 1 == 1) {
    return "Odd";
  }
  return "Even";
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```

function even_or_odd($number) {
  if (abs($number) % 2 == 1) {
    return "Odd";
  }
  return "Even";
}

```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[reverseIt](#)

Ruby:

```

def reverse_it(data)
  convert_to_integer = false
  original_class = ""

  if data.is_a? Numeric
    original_class = data.class
    data = data.to_s
    convert_to_integer = true
  end

  if data.is_a? String
    data = data.reverse
  end

  if convert_to_integer
    data = data.to_f

    if original_class == Integer
      data = data.to_i
    end
  end

  data
end

```

- 7 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Elapsed Seconds](#)

Ruby:

```

def elapsed_seconds(start_time, end_time)
  end_time - start_time
end

```

- 7 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Thoughtful - Logic Drills: Traffic light](#)

Python:

```

def update_light(current):
  if current == "green":
    return "yellow"

  if current == "yellow":
    return "red"

```

```
if current == "red":
    return "green"
```

- 6 years ago
- [Refactor](#)

C#:

```
public class Kata
{
    public static string UpdateLight(string current)
    {
        if (current == "green") {
            return "yellow";
        } else if (current == "yellow") {
            return "red";
        } else {
            return "green";
        }
    }
}
```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function updateLight(current) {
    if (current == "green") {
        return "yellow"
    } else if (current == "yellow") {
        return "red"
    }
    return "green"
}
```

- 5 years ago
- [Refactor](#)

```
function updateLight(current) {
    if (current == "green") {
        return "yellow";
    }

    if (current == "yellow") {
        return "red";
    } else {
        return "green";
    }
}
```

- 5 years ago
- [Refactor](#)

```
function updateLight(current) {
    if (current == "green") {
        return "yellow";
    }
    else if (current == "yellow") {
        return "red";
    }
    else {
        return "green";
    }
}
```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

Java:

```
public class TrafficLights {

    public static String updateLight(String current) {
        if (current.equals("green")) {
            return "yellow";
        }

        if (current.equals("yellow")) {
            return "red";
        }

        return "green";
    }
}
```

- 8 months ago
- [Refactor](#)

```
public class TrafficLights {

    public static String updateLight(String current) {
```

```

    if (current == "green") {
      return "yellow";
    }

    if (current == "yellow") {
      return "red";
    }

    return "green";
  }
}

```

- 8 months ago
- [Refactor](#)

```

public class TrafficLights {

  public static String updateLight(String current) {
    if (current == "green") {
      return "yellow";
    } else if (current == "yellow") {
      return "red";
    }

    return "green";
  }
}

```

- 5 years ago
- [Refactor](#)

```

public class TrafficLights {

  public static String updateLight(String current) {
    if (current == "green") {
      return "yellow";
    }

    return current == "yellow" ? "red" : "green";
  }
}

```

- 5 years ago
- [Refactor](#)

```

public class TrafficLights {

  public static String updateLight(String current) {
    if (current == "green") {
      return "yellow";
    }

    if (current == "yellow") {
      return "red";
    } else {
      return "green";
    }
  }
}

```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[String Merge!](#)

Ruby:

```

def string_merge(word1, word2, letter)
  puts word1[0..word1.index(letter)]
  puts word2[word2.index(letter)..-1]
  word1[0..word1.index(letter)] + word2[word2.index(letter)+1..-1]
end

```

- 8 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Are there doubles?](#)

Ruby:

```

def double_check(str)
  previous_char = ""

  str.each_char { |char|
    char = char.downcase
    return true if char == previous_char
    previous_char = char
  }

  false
end

```

- 8 months ago

- [Refactor](#)
- [Discuss](#)

7 kyu

[Average Scores](#)

Ruby:

```
def average(array)
  (array.sum / array.size.to_f).round
end
```

- 9 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Smallest Product](#)

JavaScript:

```
function smallestProduct(arr) {
  let minor = 100000;
  for (let item of arr) {
    let multiplication = 1;
    for (let innerItem of item) {
      multiplication = multiplication * innerItem;
    }

    if (minor > multiplication) {
      minor = multiplication;
    }
  }

  return minor;
}
```

- 9 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Float Precision](#)

Ruby:

```
def solution(value)
  value = (value * 100).round.to_i
  value/100.0
end
```

- 9 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Menu Display](#)

Python:

```
class Menu:
  def __init__(self, items):
    self.position = 0
    self.items = items

  def to_the_right(self):
    if self.position == len(self.items) - 1:
      self.position = 0
    else:
      self.position = self.position + 1

  def to_the_left(self):
    if self.position == 0:
      self.position = len(self.items) - 1
    else:
      self.position = self.position - 1

  def display(self):
    cont = 0
    ret = "["
    while cont < len(self.items):
      if cont == self.position:
        print(type(self.items[cont]))
        if (type(self.items[cont]) is str):
          ret = ret + "[" + str(self.items[cont]) + "]" + ", "
        else:
          ret = ret + "[" + str(self.items[cont]) + "]" + ", "
      else:
        if (type(self.items[cont]) is str):
          ret = ret + "" + str(self.items[cont]) + "" + ", "
        else:
          ret = ret + str(self.items[cont]) + ", "
      cont = cont + 1
    return ret[0: len(ret) - 2] + "]"
```

- 9 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Diagonals sum](#)

JavaScript:

```
function sum(matrix) {
  let cont = 0;
  let sum = 0;

  while (cont < matrix.length) {
    sum = sum + matrix[cont][cont];
    sum = sum + matrix[cont][matrix.length - cont - 1];
    cont = cont + 1
  }

  return sum;
}
```

- 9 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Converting 12-hour time to 24-hour time](#)

JavaScript:

```
function to24hourtime(hour, minute, period) {
  let h = hour;

  console.log(hour);

  if (period == "pm") {
    h = h + 12;
  }

  if (h == 24) {
    h = 12;
  }

  if (period == "am" && hour == 12) {
    h = 0;
  }

  return (h + "").padStart(2, '0') + (minute + "").padStart(2, 0);
}
```

- 9 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Simple Fun #270: Evil Code Medal](#)

Ruby:

```
def evil_code_medal(user_time, gold, silver, bronze)
  if user_time < gold
    return "Gold"
  elsif user_time < silver
    return "Silver"
  elsif user_time < bronze
    return "Bronze"
  end
  "None"
end
```

- 10 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Alternating between three values](#)

JavaScript:

```
function f(x, cc) {
  if (x == cc['a']) {
    return cc['b'];
  } else if (x == cc['b']) {
    return cc['c'];
  }
  return cc['a'];
}
```

- 10 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sum and Multiply](#)

JavaScript:

```
var sumAndMultiply = function(sum, multiply) {
  // Uma forma não muito eficiente, mas creio que a solução correta (fórmula) é complexa...

  let x = 0;
  let y = 0;
  while (x < 10000) {
    while (y < 10000) {
      if (x + y == sum && x * y == multiply) {
        return [x, y];
      }
      y = y + 1;
    }
    x = x + 1;
    y = 0;
  }

  return null;
}
```

- 10 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Simple Fun #352: Reagent Formula](#)

JavaScript:

```
function isValid(formula){
  if (formula.indexOf(1) > -1 && formula.indexOf(2) > -1) {
    return false;
  }

  if (formula.indexOf(3) > -1 && formula.indexOf(4) > -1) {
    return false;
  }

  if (formula.indexOf(5) != -1) {
    if (formula.indexOf(6) == -1) {
      return false;
    }
  }

  if (formula.indexOf(6) != -1) {
    if (formula.indexOf(5) == -1) {
      return false;
    }
  }

  if (formula.indexOf(7) == -1 && formula.indexOf(8) == -1) {
    return false;
  }

  return true;
}
```

- 10 months ago
- [Refactor](#)
- [Discuss](#)

Beta

[Geometric sequence - sum of all elements](#)

Ruby:

```
def geometric_sequence_sum(a, r, n)
  sum = 0
  previous = 1

  while n > 0
    n = n - 1
    sum = sum + (previous * a)
    previous = previous * r
  end

  sum
end
```

- 10 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[What dominates your array?](#)

Ruby:

```
def dominator(arr)
  totals = {}

  arr.sort.each {|item|
    totals[item] = totals[item].to_i + 1
  }

  totals.each {|key, total|
    return key if total > (arr.length / 2)
  }
end
```



```

}

return -1
end

```

- 10 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Write shortest function to calculate Average number of Array](#)

JavaScript:

```

function avg(a){
  return a.reduce((x, y) => x + y, 0)/a.length
}

```

- 10 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Insert Dashes 2](#)

Ruby:

```

def insert_dash2(num)
  num = num.to_s.split('')
  prev = -1
  ret = ''

  num.each{|i|
    if prev != -1
      if (prev != 0 && prev % 2 == 0 && i.to_i % 2 == 0 && i != '0')
        ret = ret + '**'
      elsif prev % 2 == 1 && i.to_i % 2 == 1
        ret = ret + '- '
      end
    end

    ret = ret + i
    prev = i.to_i
  }

  ret
end

```

- 10 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Take a Ten Minutes Walk](#)

Ruby:

```

def is_valid_walk(walk)
  latitude = 0
  longitude = 0

  walk.each{|direction|
    latitude = latitude + 1 if direction == "w"
    latitude = latitude - 1 if direction == "e"
    longitude = longitude + 1 if direction == "n"
    longitude = longitude - 1 if direction == "s"
  }

  walk.length==10 && latitude == 0 && longitude == 0
end

```

- 10 months ago
- [Refactor](#)
- [Discuss](#)

Beta

[Last Digit of an Array](#)

JavaScript:

```

function lastDigit(arr) {
  let ret;

  for (let item of arr) {
    if (typeof item == "number") {
      item = item + ""
      if (item.indexOf(".") == -1) {
        ret = item[item.length - 1]
      }
    }
  }

  if (ret === undefined) {
    return "No integers found.";
  }
}

```

```
return parseInt(ret);

// code goes here.
}
```

- 10 months ago
- [Refactor](#)
- [Discuss](#)

Draft

[Tug-o'-War](#)

Ruby:

```
def tug_o_war(teams)
  wins_1 = 0
  wins_2 = 0

  teams[0].each_with_index {|value, index|
    if value > teams[1][index]
      wins_1 += value - teams[1][index]
    elsif value < teams[1][index]
      wins_2 += teams[1][index] - value
    end
  }

  if wins_1 == wins_2
    if teams[0][0] > teams[1][-1]
      wins_1 = wins_1 + 1
    elsif teams[0][0] < teams[1][-1]
      wins_2 = wins_2 + 1
    end
  end

  if wins_1 > wins_2
    return "Team 1 wins!"
  elsif wins_1 < wins_2
    return "Team 2 wins!"
  end

  "It's a tie!"
end
```

- 10 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[1. Find all active students](#)

SQL:

```
-- Type your query here
select * from students where isActive=1
```

- 10 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find array.](#)

JavaScript:

```
function findArray(arr1, arr2){
  let ret = [];

  for (let i of arr2) {
    if (arr1[i] !== undefined) {
      ret.push(arr1[i]);
    }
  }

  return ret;
}
```

- 10 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Eliminate the intruders! Bit manipulation](#)

Ruby:

```
def eliminate_set_bits number
  (number.gsub /0+/, "").to_i(2)
end
```

- 10 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Sum of Digits / Digital Root](#)

Ruby:

```
def digital_root(n)
  ret = n

  while true
    ret = sum ret
    break if ret <= 9
  end

  ret
end

def sum s
  sum = 0
  s.to_s.each_char {|item|
    sum += item.to_i
  }
  sum
end
```

- 10 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Basic JS - Building a calculator](#)

JavaScript:

```
var Calculator = {};

Calculator.add = function(a, b) {
  return a + b;
}

Calculator.subtract = function(a, b) {
  return a - b;
}

Calculator.multiply = function(a, b) {
  return a * b;
}

Calculator.divide = function(a, b) {
  if (b == 0) {
    return false;
  }
  return a / b;
}
```

- 10 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[SQL Grasshopper: Select Columns](#)

SQL:

```
select custid, custname, custstate from customers
```

- 11 months ago
- [Refactor](#)
- [Discuss](#)

Retired

[Find Unique Integer](#)

JavaScript:

```
function findUniqueInteger(numbers) {
  let count = {}

  for (let number of numbers) {
    console.log(number);
    if (count[number] === undefined) {
      count[number] = 1;
    } else {
      count[number] = parseInt(count[number]) + 1;
    }
  }

  for (let item in count) {
    if (count[item] == 1) {
      return parseInt(item);
    }
  }
}
```

- 11 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Get key/value pairs as arrays](#)

Ruby:

```
def keysAndValues(data)
  indexes = []
  values = []

  data.each { |key, value|
    indexes.push key
    values.push value
  }

  [indexes, values]
end
```

- 11 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[No ifs no buts](#)

JavaScript:

```
function noIfsNoButs(a, b) {
  switch (a < b) {
    case true:
      return a + " is smaller than " + b;
  }

  switch (a > b) {
    case true:
      return a + " is greater than " + b;
  }

  switch (a == b) {
    case true:
      return a + " is equal to " + b;
  }
}
```

- 11 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

["this" is a problem](#)

JavaScript:

```
function NameMe(first, last) {
  return {
    'name': first + ' ' + last,
    'firstName': first,
    'lastName': last
  };
}
```

- 11 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Remove Odd Hashes](#)

Ruby:

```
def remove_odd_hashes(array, key_1, key_2)
  ret = []
  array.each {|hash|
    if (hash[key_1] + hash[key_2]) % 2 == 0
      ret.push hash
    end
  }
  ret
end
```

- 11 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Multiply array values and filter non-numeric](#)

JavaScript:

```
function multiplyAndFilter(array, multiplier){
  let ret = []

  for (let i of array) {
    if (typeof i !== "object") {
      if (!isNaN(i)) {
        ret.push(i * multiplier);
      }
    }
  }
}
```

```
return ret;
}
```

- 11 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Array Array Array](#)

JavaScript:

```
function explode(x){
  let i = 0;
  let s1 = parseInt(x[0]);
  let s2 = parseInt(x[1]);

  if (isNaN(s1)) {
    s1 = 0;
  }
  if (isNaN(s2)) {
    s2 = 0;
  }

  let ret = [];
  while (i < (s1 + s2) ) {
    ret.push([
      x[0],
      x[1]
    ]);
    i = i + 1;
  }

  console.log(ret.length)
  if (ret.length == 0 && isNaN(parseInt(x[0])) && isNaN(parseInt(x[1])) ) {
    return 'Void!';
  }

  return ret;
}
```

- 11 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[So basic](#)

JavaScript:

```
function convertBase20ToDecimal(init){
  // teh awesome codez
  let ret = parseInt(init, "20");

  if (isNaN(ret)) {
    return -1;
  }

  return ret;
}
```

- 11 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Basic method](#)

JavaScript:

```
// Your code
Array.prototype.max = function() {
  let value = this.sort()
  return parseInt(value[value.length - 1])
}
```

- 11 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[ES6 string addition](#)

JavaScript:

```
function joinStrings(string1, string2){
  return `${string1} ${string2}`;
}
```

- 11 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Running out of space](#)

Ruby:

```
def spacey(array)
  ret = []
  string = ""
  new_string = ""

  array.each {|item|
    new_string = string + item
    ret.push new_string
    string = new_string
  }

  ret
end
```

- 11 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Easy SQL: LowerCase](#)

SQL:

```
/* SQL */
select id, name, birthday, lower(race) as race from demographics
```

- 11 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Character Concatenation](#)

Ruby:

```
def char_concat(word)
  ret = ""

  word.each_char.with_index { |char, index|
    break if index == word.length / 2
    ret += char + word[word.length - index - 1] + (index + 1).to_s
  }

  ret
end
```

- 12 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Two Sum](#)

Ruby:

```
def two_sum(numbers, target)
  numbers.each_with_index { |number1, index1|
    numbers.each_with_index { |number2, index2|
      next if index1 == index2
      return [index1, index2] if number1 + number2 == target
    }
  }
end
```

- 12 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Nth Root of a Number](#)

Ruby:

```
def root(x, n)
  x ** (1.0/n)
end
```

- 12 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sum Factorial](#)

Ruby:

```
def sum_factorial(lst)
  total = 0
  lst.each { |number|
    total = total + factorial(number)
  }
  total
end
```

```

end

def factorial number
  total = 1

  while number > 1
    total = total * number
    number = number - 1
  end

  total
end

```

- 12 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Quadrants](#)

JavaScript:

```

function quadrant(x, y) {
  if (x > 0) {
    if (y > 0) {
      return 1;
    } else {
      return 4;
    }
  } else {
    if (y > 0) {
      return 2;
    } else {
      return 3;
    }
  }
}

```

- 12 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[OOP: Object Oriented Piracy](#)

Ruby:

```

class Ship
  def initialize(draft,crew)
    @draft=draft
    @crew=crew
  end

  def is_worth_it
    weight = @draft - @crew * 1.5
    return false if weight <= 20
    true
  end
end

```

- 12 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Hex Word Sum](#)

Ruby:

```

def hex_word_sum(s)
  s = s.gsub /S/, "5"
  s = s.gsub /O/, "0"

  sum = 0

  s.split(" ").each { |word|
    unless (word.match /[G-Zg-z]/).to_s.length > 0
      sum += word.to_i(16)
    end
  }

  sum
end

```

- 12 months ago
- [Refactor](#)
- [Discuss](#)

```

def hex_word_sum(s)
  s = s.gsub /S/, "5"
  s = s.gsub /O/, "0"

```

sum = 0

```

  filtered_word = ""
  s.split(" ").each { |word|
    unless (word.match /[G-Zg-z]/).to_s.length > 0
      sum += word.to_i(16)

```

```

    filtered_word += word
  end
}

sum
end

```

- 12 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Node.js Intro](#)

JavaScript:

```

String.prototype.toBase64 = function() {
  return Buffer.from(this, 'utf8').toString('base64')
}

String.prototype.fromBase64 = function() {
  return Buffer.from(this, 'base64').toString('utf8')
}

```

- 12 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Alternate Square Sum](#)

JavaScript:

```

function alternateSqSum(arr){
  let ret = 0;
  for (let i in arr) {
    if (i % 2 == 0) {
      ret = ret + arr[i];
    } else {
      ret = ret + (arr[i] * arr[i]);
    }
  }
  return ret;
}

```

- 13 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Between Extremes](#)

Ruby:

```

def between_extremes(numbers)
  numbers.sort!
  numbers[-1] - numbers[0]
end

```

- 13 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Say hello!](#)

Ruby:

```

def greet(name)
  return nil if name.nil? or name.empty?
  return "hello #{name}!"
end

```

- 13 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Decreasing Inputs](#)

Ruby:

```

def add(*args)
  sum = 0.0
  args.each_with_index {|arg, index|
    sum = sum + (arg/(index+1.0))
  }
  sum.round
end

```

- 13 months ago
- [Refactor](#)
- [Discuss](#)



7 kyu  
[Speed Limit](#)

Python:

```
def speed_limit(speed, signals):
    total = 0

    for limit in signals:
        if speed >= limit + 30:
            total = total + 500
        elif speed >= limit + 20:
            total = total + 250
        elif speed >= limit + 10:
            total = total + 100

    return total
```

- 13 months ago
- [Refactor](#)
- [Discuss](#)

Retired

[bruh](#)

Python:

```
def newmax(arr):
    return max(arr)

def newmin(arr):
    return min(arr)

def newmean(arr):
    return int(sum(arr) / len(arr))
```

- 13 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Multiply the strings in the array.](#)

JavaScript:

```
function arrMultiply(arr){
    return parseInt(arr[0]) * parseInt(arr[1]) + "";
}
```

- 13 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Find all occurrences of an element in an array.](#)

Ruby:

```
def find_all arr,n
  ret = []

  arr.each.with_index { |number, index|
    ret.push index if n == number
  }

  ret
end
```

- 13 months ago
- [Refactor](#)
- [Discuss](#)

5 kyu  
[Interleaving Arrays](#)

Ruby:

```
def interleave(*param)
  ret = []
  major_size= 0
  cont = 0

  param.each { |arr|
    if arr.size > major_size
      major_size = arr.size
    end
  }

  while cont < major_size
    param.each { |arr|
      ret.push arr[cont]
    }
  }
end
```

```

    cont = cont + 1
  end

  return ret
end

```

- 13 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Find twins](#)

Ruby:

```

def elimination(arr)
  arr.each {|number|
    total = arr.count number
    return number if total > 1
  }

  return nil
end

```

- 13 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Asterisk it](#)

Ruby:

```

def asterisk_it(inp)
  if inp.is_a? Integer
    inp = inp.to_s
  elsif inp.is_a? Array
    inp = inp.join('')
  end

  ret = ""
  inp.each_char.with_index{|char, index|
    ret = ret + char
    puts inp[index]
    if char.to_i % 2 == 0 && inp[index+1].to_i % 2 == 0
      ret = ret + "*"
    end
  }

  if ret[-1] == "*"
    ret = ret[0..-2]
  end

  ret
end

```

- 13 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Find min and max](#)

Ruby:

```

def get_min_max(seq)
  [seq.min, seq.max]
end

```

- 13 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Simple string reversal II](#)

Ruby:

```

def solve st,a,b
  start = a == 0 ? "" : st[0..a-1]
  start = start.nil? ? "" : start
  #puts start

  middle = st[a..b].reverse
  middle = middle.nil? ? "" : middle
  #puts middle

  ending = st[b+1..-1]
  ending = ending.nil? ? "" : ending
  #puts ending
  puts ending.nil?
  #puts "---"

  start + middle + ending
end

```

- 13 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Compare Versions](#)

Ruby:

```
def compare_versions(version1, version2)
  major_version_array = version1.split(".")
  minor_version_array = version2.split(".")

  minor_version_array.each_with_index { |minor_version_part, index|
    return false if major_version_array[index].nil?

    return false if minor_version_part.to_i > major_version_array[index].to_i
  }

  true
end
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Remove the noise from the string](#)

JavaScript:

```
function removeNoise(str){
  let ret = str.replaceAll("%", "")
  .replaceAll("$", "")
  .replaceAll("&", "")
  .replaceAll("/", "")
  .replaceAll("#", "")
  .replaceAll("-", "")
  .replaceAll("@", "")
  .replaceAll("|", "")
  .replaceAll("e", "")
  .replaceAll("\\", "")
  .replaceAll("a", "")

  console.log(ret)
  return ret;
}
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Matrix Addition](#)

Ruby:

```
def matrixAddition(a, b)
  ret = []
  a.each_with_index{ |external_item, external_index|
    ret[external_index] = []
    external_item.each_with_index{ |internal_item, internal_index|
      ret[external_index][internal_index] = internal_item + b[external_index][internal_index]
    }
  }
  ret
end
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find the Squares](#)

Ruby:

```
def find_squares(num)
  i = 2

  while true
    difference = (i * i) - ((i - 1) * (i - 1))

    if difference == num
      return (i * i).to_s + "-" + ((i - 1) * (i - 1)).to_s
    elsif difference > num
      return "1-0"
    end
    i = i + 1
  end
end
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Can you keep a secret?](#)

JavaScript:

```
function createSecretHolder(secret) {
  let _secret = secret;

  return {
    getSecret: function() {
      return _secret;
    },

    setSecret: function (value) {
      _secret = value;
    }
  }
}
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Vowel one](#)

JavaScript:

```
function vowelOne(s){
  let ret = "";
  s = s.toLowerCase();
  for (let char of s) {
    if (char == 'a' || char == 'e' || char == 'i' || char == 'o' || char == 'u') {
      ret += "1";
    } else {
      ret += "0";
    }
  }
  return ret;
}
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Keep the Order](#)

Ruby:

```
def keep_order(ary, val)
  position = 1
  ary.each{|number|
    if number >= val
      return position - 1
    end

    position = position + 1
  }
  position - 1
end
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Simple string characters](#)

Ruby:

```
def solve s
  uppercase_occurrences = 0
  lowercase_occurrences = 0
  numbers_occurrences = 0
  special_characters_occurrences = 0
  s.each_char {|occurrence|
    puts occurrence
    puts occurrence.ord
    puts "---"
    if occurrence.ord >= 65 && occurrence.ord <= 90
      uppercase_occurrences = uppercase_occurrences + 1
    elsif occurrence.ord >= 97 && occurrence.ord <= 122
      lowercase_occurrences = lowercase_occurrences + 1
    elsif occurrence.ord >= 48 && occurrence.ord <= 57
      numbers_occurrences = numbers_occurrences + 1
    else
      special_characters_occurrences = special_characters_occurrences + 1
    end
  }
  puts "*****"
  [
    uppercase_occurrences,
    lowercase_occurrences,
    numbers_occurrences,
    special_characters_occurrences
  ]
end
```

```
]
end
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find the index of the first occurrence of an item in a list \(with a twist\)](#)

Python:

```
def index_finder(lst, x):
    for index, item in enumerate(lst):
        if index == 0:
            continue
        if item == x:
            return index
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find the index of the second occurrence of a letter in a string](#)

Python:

```
def second_symbol(s, symbol):
    already_founded = False
    cont = 0
    for char in s:
        if char == symbol:
            if already_founded:
                return cont
            else:
                already_founded = True
                cont = cont + 1
    return -1
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Multiply](#)

JavaScript:

```
function multiply(a, b){
    return a * b
}
```

- 14 months ago
- [Refactor](#)

```
function multiply(a, b){
    return a * b;
}
```

- 5 years ago
- [Refactor](#)

```
function multiply(a, b){
    return a * b;
}
```

- 5 years ago
- [Refactor](#)

```
function multiply(a, b){
    return a * b
}
```

- 9 years ago
- [Refactor](#)

Python:

```
def multiply(a, b):
    return a * b
```

- 6 years ago
- [Refactor](#)

```
def multiply(a, b):
    return a * b
```

- 7 years ago
- [Refactor](#)

Java:

```
public class Multiply {
    public static Double multiply(Double a, Double b) {
        return a * b;
    }
}
```

- 7 years ago
- [Refactor](#)

PHP:

```
function multiply($a, $b) {
    return $a * $b;
}
```

- 7 years ago
- [Refactor](#)

```
function multiply($a, $b) {
    return $a * $b;
}
```

- 7 years ago
- [Refactor](#)

C:

```
int multiply(int a, char *b) {
    return a * (int) b;
}
```

- 5 years ago
- [Refactor](#)

```
int multiply(int a, int b) {
    return a * b;
}
```

- 6 years ago
- [Refactor](#)

Kotlin:

```
fun multiply(x:Double, y:Double):Double {
    return x * y;
}
```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

Objective-C:

```
int multiply(int a, int b) {
    return a * b;
}
```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

R:

```
mul <- function(a, b) {
    a * b;
}
```

- 2 years ago
- [Refactor](#)

```
mul <- function(a, b) {
    a * b # try to figure out why it doesn't work!
}
```

- 5 years ago
- [Refactor](#)

```
mul <- function(a, b) {
    result <- a * b;
}
```

- 6 years ago
- [Refactor](#)

TypeScript:

```
export function multiply(a, b){
    return a * b;
}
```

- 5 years ago

- [Refactor](#)
- [Discuss](#)

```
export function multiply(a, b){
  return a * b;
}
```

- 6 years ago
- [Refactor](#)

C#:

```
public class CustomMath {
  public static int multiply(int a, int b) {
    return a * b;
  }
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Go:

```
package multiply

func Multiply(a, b int) int {
  return a * b
}
```

- 5 years ago
- [Refactor](#)

```
package multiply

func Multiply(a, b int) int {
  return a * b
}
```

- 6 years ago
- [Refactor](#)

PowerShell:

```
function Multiply($a, $b) {
  return $a * $b;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

```
function Multiply($a, $b) {
  return $a * $b
}
```

- 5 years ago
- [Refactor](#)

Solidity:

```
pragma solidity ^0.4.13;

contract DummyToken {
  function multiply(int a, int b) returns (int) {
    return a * b;
  }
}
```

- 4 years ago
- [Refactor](#)

```
pragma solidity ^0.4.13;

contract DummyToken {
  function multiply(int a, int b) returns (int) {
    return a * b;
  }
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

C++:

```
int multiply(int a, int b)
{
  return a * b;
}
```

- 5 years ago
- [Refactor](#)

- [Discuss](#)

Crystal:

```
def multiply(x, y)
  x * y
end
```

- 4 years ago
- [Refactor](#)

```
def multiply(x, y)
  return x * y
end
```

- 4 years ago
- [Refactor](#)

Clojure:

```
(ns multiply.bug.fix)
```

```
(defn multiply [a, b]
  (* a b))
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

```
(ns multiply.bug.fix)
```

```
(defn multiply [a b]
  (* a b))
```

- 4 years ago
- [Refactor](#)

CoffeeScript:

```
multiply = (a, b) -> a * b
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Dart:

```
int multiply(int a, int b) {
  return a * b;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Elixir:

```
defmodule Multiply do
  def multiply(a, b) do
    a * b
  end
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

```
defmodule Multiply do
  def multiply(a, b) do
    a * b;
  end
end
```

- 5 years ago
- [Refactor](#)

Elm:

```
module MultiplyBugFix exposing (..)
```

```
multiply : Int -> Int -> Int
multiply x y = x * y
```

- 14 months ago
- [Refactor](#)

```
module MultiplyBugFix exposing (..)
```

```
multiply : Int -> Int -> Int
multiply x y = x * y
```

- 5 years ago



- [Refactor](#)

```
module MultiplyBugFix exposing (..)
```

```
multiply : Int -> Int -> Int
multiply x y = x*y
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Erlang:

```
-module(bug_fix).
-export([multiply/2]).
```

```
-spec multiply(integer(), integer()) -> integer().
multiply(A, B) -> A * B.
```

- 14 months ago
- [Refactor](#)

```
-module(bug_fix).
-export([multiply/2]).
```

```
-spec multiply(integer(), integer()) -> integer().
multiply(A, B) -> A*B.
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

F#:

```
let multiply a b = a * b
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Groovy:

```
class Multiply {
    static multiply(a, b) {
        a * b
    }
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Julia:

```
module Solution
export multiply
function multiply(a, b)
    a * b
end
end
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Lua:

```
local kata = {}
```

```
function kata.multiply(a, b)
    return a * b;
end
```

```
return kata
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

```
local kata = {}
```

```
function kata.multiply(a, b)
    return a * b
end
```

```
return kata
```

- 5 years ago
- [Refactor](#)

Nim:

```
proc multiply*(a:int, b: int): int = return a * b
```

- 16 months ago
- [Refactor](#)
- [Discuss](#)

PureScript:

```
module MultiplyBugFix where

import Prelude

multiply :: Int -> Int -> Int
multiply x y = x * y
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Reason:

```
let multiply = (a, b) => a * b;
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def multiply(a, b)
  a * b
end
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Rust:

```
fn multiply(a: u32, b: u32) -> u32 {
  return a * b;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Scala:

```
object Multiply {
  def multiply(a: Int, b: Int) = a * b
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

```
object Multiply {
  def multiply(a: Int, b: Int) = a * b
}
```

- 5 years ago
- [Refactor](#)

Shell:

```
#!/bin/bash -e
a=$1
b=$2
echo $((a*b))
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Swift:

```
func multiply(_ a: Double, _ b: Double) -> Double {
  return a * b;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

SQL:

```
SELECT price * amount AS total FROM items
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Agda:

```
{-# OPTIONS --safe #-}
module Solution where

open import Data.Nat

multiply : N → N → N
multiply a b = a * b
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Racket:

```
#lang racket
(provide multiply)

(define (multiply a b) (* a b))
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

VB:

```
Public Module Example
    Public Function Multiply(ByVal a As Integer, ByVal b As Integer) As Integer
        Return a * b
    End Function
End Module
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

CFML:

```
component {
    function multiply(a, b) {
        return a * b;
    }
}
```

- 4 years ago
- [Refactor](#)

Haxe:

```
class Kata {
    public static function multiply(a, b) {
        return a * b;
    }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

COBOL:

```
123456*
IDENTIFICATION DIVISION.
PROGRAM-ID. SOLUTION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 PRODAND-1 PIC 9(04) VALUE 1.
01 PRODAND-2 PIC 9(04) VALUE 1.
01 RESULT PIC 9(04).
PROCEDURE DIVISION.
GOBACK.
F01-MULT SECTION.
MULTIPLY PRODAND-1 BY PRODAND-2 GIVING RESULT.
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

CommonLisp:

```
(defpackage #:challenge/solution
  (:use #:cl)
  (:export #:multiply))
(in-package #:challenge/solution)

(defun multiply (a b) (* a b))
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Perl:

```
package Solution;

use 5.030;
use strict;
use warnings;
use Exporter qw(import);

our @EXPORT_OK = qw(multiply);

sub multiply {
    my $a = shift;
    my $b = shift;
    return $a * $b;
}

1;
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Raku:

```
use v6;
unit module Solution;

sub multiply(Int $a, Int $b --> Int) is export {
    $a * $b;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Pascal:

```
unit BugFixMultiply;

{$mode objfpc}{$H+}

interface

function Multiply(const A: Integer; const B: Integer): Integer;

implementation

function Multiply(const A: Integer; const B: Integer): Integer;
begin
    Result := A * B;
end;

end.
```

- 3 years ago
- [Refactor](#)

D:

```
module solution;

export int multiply(int a, int b) {
    return a * b;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Powers of 3](#)

JavaScript:

```
function largestPower(n){
    let ret = 0;
    let i = 0;

    if (n == 1) {
        return -1;
    }

    while (true) {
        let tmp;

        tmp = Math.pow(3, i);
        console.log("-----")
        console.log(tmp);
        console.log(n);
        console.log(ret);
        console.log("-----")
    }
}
```

```

    if (tmp < n) {
      ret = i;
    } else {
      break;
    }
    i++;
  }
  return ret;
}

```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Head, Tail, Init and Last](#)

Ruby:

```

def head array
  array[0]
end

def tail array
  array[1..-1]
end

def init array
  array[0..-2]
end

def last array
  array[-1]
end

```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Unexpected parsing](#)

Ruby:

```

def get_status(is_busy)
  status = is_busy ? "busy" : "available"
  ret = Hash.new
  ret['status'] = status
  return ret
end

```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Trimming a string](#)

JavaScript:

```

function trim(str, size) {
  if (str.length - size <= 0) {
    return str;
  }

  if (size <= 3) {
    return str.substring(0, size) + "...";
  }

  return str.substring(0, size - 3) + "...";
}

```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Fix the loop!](#)

Python:

```

def list_animals(animals):
  list = ''
  for i, name in enumerate(animals):
    list += str(i + 1) + '. ' + name + '\n'
  return list

```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Training JS #12: loop statement --for.in and for.of](#)

JavaScript:

```
function giveMeFive(obj){
  let ret = []

  for (let i in obj) {
    if (i.length == 5) {
      ret.push(i);
    }
    if (obj[i].length == 5) {
      ret.push(obj[i]);
    }
  }
  return ret;
}
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Miles per gallon to kilometers per liter](#)

Ruby:

```
def converter(mpg)
  (mpg * 0.35400604).round(2)
end
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[SQL: Disorder](#)

SQL:

```
select number from numbers order by random()
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Fizz Buzz](#)

Ruby:

```
# return an array
def fizzbuzz(n)
  i = 1
  ret = []
  while i < n + 1
    text = i

    if i % 3 == 0 && i % 5 == 0
      text = "FizzBuzz"
    elsif i % 3 == 0
      text = "Fizz"
    elsif i % 5 == 0
      text = "Buzz"
    end

    ret.push text
    i = i + 1
  end

  ret
end
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Identify Case](#)

Ruby:

```
def id(c_str)
  has_underline = false
  has_dash = false
  has_camel_case = false
  puts c_str

  unless c_str.match(/_/).nil?
    has_underline = true
  end

  unless c_str.match(/_{2,}/).nil?
    return "none"
  end

  unless c_str.match(/-/).nil?
```

```

    has_dash = true
  end

  unless c_str.match(/-{2,}/).nil?
    return "none"
  end

  unless c_str.match(/[A-Z]/).nil?
    has_camel_case = true
  end

  unless c_str.match(/[A-Z]{2,}/).nil?
    return "none"
  end

  if has_underline && ! has_camel_case && ! has_dash
    return "snake"
  end

  if ! has_underline && has_camel_case && ! has_dash
    return "camel"
  end

  if ! has_underline && ! has_camel_case && has_dash
    return "kebab"
  end

  return "none"
end

```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Arithmetic Sequence!](#)

JavaScript:

```

var nthterm = function(number, n, c){
  let count = 0
  while (true) {
    if (count == n) {
      break;
    }
    number = number + c;
    count = count + 1;
  }
  return number;
}

```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

Python:

```

def nthterm(number, n, c):
    count = 0

    while (True):
        if (count == n):
            break

        number = number + c;
        count = count + 1;

    return number;

```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Square Every Digit](#)

Ruby:

```

def square_digits num
  ret = ""
  num.to_s.split("").each { |char|
    ret += (char.to_i * char.to_i).to_s
  }
  ret.to_i
end

```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

```

def square_digits num
  num = num.to_s
  ret = ""
  num.split("").each {|c|
    ret = ret + (c.to_i ** 2).to_s
  }

```

```
ret.to_i
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Python:

```
def square_digits(num):
    num_string = str(num)
    ret = ""

    for c in range(len(num_string)):
        ret = ret + str(int(num_string[c]) ** 2)

    return int(ret)
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Transportation on vacation](#)

Ruby:

```
def rental_car_cost(d)
  puts d
  total = d * 40
  if d >= 7
    total -= 50
  elsif d >= 3
    total -= 20
  end

  total
end
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

Python:

```
def rental_car_cost(d):
    total = d * 40
    if d >= 7:
        total -= 50
    elif d >= 3:
        total -= 20

    return total
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Calculate Two People's Individual Ages](#)

JavaScript:

```
function getAges(sum,difference) {
  if (difference < 0 || sum < 0) {
    return null;
  }

  let a = (sum + difference) / 2;
  let b = sum - a;

  if (a < 0 || b < 0) {
    return null;
  }

  return [a, b];
};
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

Python:

```
def get_ages(sum_, difference):
    if (difference < 0 or sum_ < 0):
        return None

    a = (sum_ + difference) / 2;
    b = sum_ - a;

    if a % 1 > 0:
        a = float(a)
```



```
if b % 1 > 0:
    b = float(b)
```

```
if (a < 0 or b < 0):
    return None
```

```
return (a, b);
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Number Of Occurrences](#)

JavaScript:

```
Array.prototype.numberOfOccurrences = function() {
    let cont = 0

    for (let i = 0; i < this.length; i++) {
        if (this[i] == arguments[0]) {
            cont = cont + 1
        }
    }

    return cont;
}
```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

Python:

```
def number_of_occurrences(element, sample):
    total = 0

    for i in sample:
        if i == element:
            total = total + 1

    return total
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Replace With Alphabet Position](#)

Ruby:

```
def alphabet_position(text)
    ret = ""
    text = text.gsub /\s+/, ""
    text.downcase!

    text.each_char{|letter|
        ord = letter.ord

        if ord >=97 && ord <= 122
            ret += (ord - 96).to_s + " "
        end
    }

    ret[0..-2]
end
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

Retired

[Truncate a string!](#)

Ruby:

```
def truncate_string(str,n)
    length = str.length

    ret = str

    if n < length
        if n == 0
            ret = "... "
        elsif n <= 3
            ret = str[0..(n-1)] + "... "
        else
            ret = str[0..(n-4)] + "... "
        end
    end

    ret
end
```

- 14 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Encrypt this!](#)

Ruby:

```
def encrypt_this(text)
  ret = ""
  text.split(" ").each{|word|
    ret += encrypt(word) + " "
  }
  ret.strip
end

def encrypt text
  begin
    if text.length > 2
      tmp = text[1] if text.length > 0
      text[1] = text[-1] if text.length > 0
      text[-1] = tmp if text.length > 0
    end
  rescue
  end

  text[0] = text[0].ord.to_s
  text
end
```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[All Star Code Challenge #22](#)

Ruby:

```
def to_time(seconds)
  minutes = (seconds / 60).floor
  hours = 0
  while minutes > 59
    minutes = minutes - 60
    hours = hours + 1
  end

  hours.to_s + " hour(s) and " + minutes.to_s + " minute(s)"
end
```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function toTime(seconds) {
  let hours = Math.floor(seconds / 3600)
  seconds = seconds - hours * 3600
  let minutes = Math.floor(seconds / 60)
  return hours + " hour(s) and " + minutes + " minute(s)"
}
```

- 16 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Twisted Sum](#)

Ruby:

```
def solution(n)
  current = 1
  sum = 0
  number_to_sum = 0
  while current <= n
    number_to_sum = current
    if number_to_sum > 9
      num = current.to_s.split("")

      partial_sum = 0
      num.each {|n|
        partial_sum = partial_sum + n.to_i
      }
      sum = sum + partial_sum
    else
      sum = sum + number_to_sum
    end
    current = current + 1
  end
  sum
end
```

- 15 months ago

- [Refactor](#)
- [Discuss](#)

6 kyu

[Count the smiley faces!](#)

Ruby:

```
def count_smileys(arr)
  count = 0
  arr.each{|face|
    next if face.index(":").nil? && face.index(";").nil?
    next if face[1] == " "
    next if face.index("D").nil? && face.index("D").nil?
    count = count + 1
  }
  return count
end
```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[CompoundArray](#)

Ruby:

```
def compound_array(a, b)
  ret = []

  if a.length > b.length
    array_with_major_length = a
    array_with_minor_length = b
  else
    array_with_major_length = b
    array_with_minor_length = a
  end

  array_with_major_length.each_with_index{|item, index|
    unless a[index].nil?
      ret.push a[index]
    end
    unless b[index].nil?
      ret.push b[index]
    end
  }

  ret
end
```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Area of a Circle](#)

JavaScript:

```
var circleArea = function(radius) {
  if (radius <= 0 || isNaN(radius)) {
    return false;
  }
  return Math.round(Math.PI * radius * radius * 100)/100;
};
```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Odd-Even String Sort](#)

Ruby:

```
def sort_my_string(s)
  evens = []
  odds = []

  s.each_char.with_index{|char, index|
    if index % 2 == 0
      evens.push char
    else
      odds.push char
    end
  }

  evens.join("") + " " + odds.join("")
end
```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

```

def sort_my_string(s)
  evens = []
  odds = []

  s.each_char.with_index(s.length){|char, index|
    if index % 2 == 0
      evens.push char
    else
      odds.push(char)
    end
  }

  if s.length % 2 == 0
    i = 0
    ret = ""
    while (i < evens.length)
      ret = ret + evens[i]
      i = i + 1
    end

    ret = ret + " "

    i = 0
    while (i < odds.length)
      ret = ret + odds[i]
      i = i + 1
    end
  else
    i = 0
    ret = ""
    while (i < odds.length)
      ret = ret + odds[i]
      i = i + 1
    end

    ret = ret + " "

    i = 0
    while (i < evens.length)
      ret = ret + evens[i]
      i = i + 1
    end
  end

  ret
end

```

- 16 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Training JS #6: Basic data types--Boolean and conditional statements if, else](#)

JavaScript:

```

function trueOrFalse(val) {
  if ((isNaN(val) && val !== undefined) || eval(val) || val == true) {
    return "true";
  }
  return "false";
}

```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Common Substrings](#)

Ruby:

```

def substring_test(str1, str2)
  previous_index = nil
  ret = false
  ret1 = true
  ret2 = true
  str1.downcase!
  str2.downcase!

  return false if str1.empty? || str2.empty?

  str1.each_char { |char|
    index = str2.index(char)

    if !previous_index.nil? && (index == previous_index + 1)
      ret = true
    end

    previous_index = index
  }
  ret
end

```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Count the Digit](#)

Ruby:

```
def nb_dig(n, d)
  numbers = []
  count_n = 0
  total_digit = 0

  while count_n <= n
    numbers.push count_n * count_n
    count_n += 1
  end

  numbers.each {|number|
    total_digit += (number.to_s).count(d.to_s)
  }

  total_digit
end
```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Debug Sum of Digits of a Number](#)

JavaScript:

```
function getSumOfDigits(integer) {
  let stringOfInteger = integer + "";
  let sum = null;
  const digits = stringOfInteger.split("");

  for (let digit of digits) {
    sum += parseInt(digit);
  }

  return sum;
}
```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[The unknown but known variables: Addition](#)

JavaScript:

```
function theVar(theVariables) {
  let a = theVariables.charCodeAt(0)-96;
  let b = theVariables.charCodeAt(2)-96;

  return a + b;
}
```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Especially Joyful Numbers](#)

PHP:

```
function number_joy(int $n): bool {
  $numbers = str_split($n, 1);

  $sumNumbers = array_sum($numbers);
  $sumNumbersReversed = (int) strrev('' . $sumNumbers);

  return $sumNumbers * $sumNumbersReversed == $n;
}
```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[simple calculator](#)

PHP:

```
function calculator($a, $b, $sign) {
  if (! is_float($a) && ! is_integer($a)) {
    return "unknown value";
  }

  if (! is_float($b) && ! is_integer($b)) {
    return "unknown value";
  }

  var_dump($a);
  echo $b;
  echo $sign;
}
```

```

echo "-----";

if ($sign != "+" && $sign != "-" && $sign != "*" && $sign != "/") {
  return "unknown value";
}

if ($sign == "+") {
  return $a + $b;
}
if ($sign == "-") {
  return $a - $b;
}
if ($sign == "*") {
  return $a * $b;
}
return $a / $b;
}

```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Discover The Original Price](#)

Ruby:

```

def discover_original_price(discounted_price, sale_percentage)
  # original_price * (1 - sale_percentage/100.0) = discounted_price
  # original_price = discounted_price / (1 - sale_percentage/100.0)
  # Ex 1: 75 / (1 - 25/100)
  ret = (((discounted_price / (1 - sale_percentage/100.0)) * 100).round) / 100.0
  ret
end

```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Count consonants](#)

Ruby:

```

def consonant_count(str)
  total = 0
  str.downcase!
  str.each_char{|c|
    total = total + 1 if c != "a" && c != "e" && c != "i" && c != "o" && c != "u" && c != " " && c.ord > 95 && c.ord < 126
  }
  total
end

```

- 15 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[greetings with First Name AND Last Name](#)

Ruby:

```

#using classes is good practice!

class Person
  def initialize(fn, ln)
    @first_name = fn
    @last_name = ln
  end

  def greet
    "Hello, #{@first_name} #{@last_name}!"
  end
end

```

- 16 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Kebabize](#)

Ruby:

```

def kebabize(str)
  ret = ""
  str.each_byte do |c|
    if c >= 65 && c <= 90
      c = c + 32
      ret = ret + "-" + c.chr
    elsif !(c >=48 && c <=57)
      ret = ret + c.chr
    end
  end
  ret
end

```

```

if ret[0] == "-"
  ret = ret[1..1000]
end

if str[-1] == "-"
  ret = ret[0..-1]
end

ret
end

```

- 16 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Sorting the Odd way!](#)

Ruby:

```

def sort_it_out(array)
  odds = []
  evens = []

  array.each{|i|
    a = i
    i = i.to_i

    if i % 2 == 0
      evens.push a
    else
      odds.push a
    end
  }

  odds.sort + evens.sort.reverse
end

```

- 16 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Multiply the number](#)

JavaScript:

```

function multiply(number){
  let numberString = number + '';
  let possibleCoefficient = parseInt(numberString.length);
  let coefficient;
  if (number >= 0) {
    coefficient = possibleCoefficient;
  } else {
    coefficient = possibleCoefficient - 1;
  }
  return number * Math.pow(5, coefficient);
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```

def multiply(n)
  n * 5 ** (n.abs.to_s.size.to_i)
end

def multiply(n)
  x = 0

  if n < 0
    x = 5 ** (n.to_s.length.to_i - 1)
  else
    x = 5 ** (n.to_s.length.to_i)
  end

  n * x
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Cat Years, Dog Years \(2\)](#)

Ruby:

```

def owned_cat_and_dog(cat_years, dog_years)
  return [get_cat_years(cat_years), get_dog_years(dog_years)]
end

```

```

def get_cat_years years
  ret = 0

  if years >= 15
    ret = 1
    years = years - 15

    if years >= 9
      ret = 2
      years = years - 9

      while years >= 4
        ret = ret + 1
        years = years - 4
      end
    end
  end

  ret
end

```

```

def get_dog_years years
  ret = 0

  if years >= 15
    ret = 1
    years = years - 15

    if years >= 9
      ret = 2
      years = years - 9

      while years >= 5
        ret = ret + 1
        years = years - 5
      end
    end
  end

  ret
end

```

- 16 months ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Title Case](#)

Ruby:

```

def title_case(title, minor_words = '')
  ret = ""
  minor_words = minor_words.split(" ")
  minor_words.each_with_index {|word, index|
    minor_words[index] = word.downcase
  }
  title.split(" ").each {|word|
    word = word.downcase
    if minor_words.index(word).nil?
      word = word.capitalize
    end
    ret = ret + word + " "
  }

  puts minor_words
  return "" if ret.empty?
  ret = ret[0].capitalize + ret[1..99]
  ret.strip
end

```

- 16 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Multiples and Digit Sums](#)

JavaScript:

```

function procedure(n){
  let multiples = getMultiples(n);
  let sum = 0;

  console.log(multiples);

  for (let i of multiples) {
    sum += getSumOfDigits(i);
  }

  return sum;
}

function getMultiples(n) {
  let multiples = [];
  let total = 0;
  let cont = 1;

  while (true) {
    total = cont * n;

```



```

    if (total > 100) {
      break;
    }
    multiples.push(total);
    cont++;
  }
  return multiples;
}

function getSumOfDigits(n) {
  let nString = String(n);
  let sum = 0;

  for (const i of nString) {
    sum += parseInt(i);
  }

  return sum;
}

```

- 17 months ago
- [Refactor](#)
- [Discuss](#)

Retired

[Apartment rent for the couple.](#)

Python:

```

def floor_rent(RentTopFloor, FloorWanted):
    return str(RentTopFloor + (20 - FloorWanted) * 200) + " Dollars"

```

- 17 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Enumerable Magic #4 - True for None?](#)

JavaScript:

```

function none(arr, fun){
  let ret = true;

  for (let i of arr) {
    ret = ret && !fun(i);
  }

  return ret;
}

```

- 17 months ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Pythagorean Triple](#)

JavaScript:

```

function isPythagoreanTriple(integers) {
  if (Math.pow(integers[0], 2) == Math.pow(integers[1], 2) + Math.pow(integers[2], 2) ){
    return true;
  }
  if (Math.pow(integers[1], 2) == Math.pow(integers[0], 2) + Math.pow(integers[2], 2) ){
    return true;
  }
  if (Math.pow(integers[2], 2) == Math.pow(integers[0], 2) + Math.pow(integers[1], 2) ){
    return true;
  }

  return false;
}

```

- 17 months ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[The Coupon Code](#)

JavaScript:

```

function checkCoupon(enteredCode, correctCode, currentDate, expirationDate) {
  if (enteredCode !== correctCode) {
    return false;
  }

  if (new Date(currentDate) > new Date(expirationDate)) {
    return false;
  }

  return true;
}

```

- 2 years ago

- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export function checkCoupon(enteredCode: string, correctCode: string, currentDate: string, expirationDate: string): boolean {
  if (enteredCode !== correctCode) {
    return false;
  }

  if (new Date(currentDate) > new Date(expirationDate)) {
    return false;
  }

  return true;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Translate English to Code: Usain Bolt](#)

Python:

```
def Faster_Than_Usain_Bolt(person_speed):
    if person_speed > 37.5:
        return "Person";
    elif person_speed < 37.5:
        return "Usain Bolt"

    return "Tie"
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[An old taste of JavaScript](#)

JavaScript:

<!---->

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[\[Geometry A-2\]: Length of a vector](#)

Ruby:

```
def vector_length(vector)
  Math.sqrt((vector[1][0] - vector[0][0]) ** 2 + (vector[0][1] - vector[1][1]) ** 2)
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Training JS #8: Conditional statement--switch](#)

JavaScript:

```
function howManydays(month){
  var days;
  switch (month) {
    case 1:
      return 31;
    case 2:
      return 28;
    case 3:
      return 31;
    case 4:
      return 30;
    case 5:
      return 31;
    case 6:
      return 30;
    case 7:
      return 31;
    case 8:
      return 31;
    case 9:
      return 30;
    case 10:
      return 31;
    case 11:
      return 30;
    case 12:
      return 31;
  }
}
```

```
return days;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Training JS #10: loop statement --for](#)

JavaScript:

```
function pickIt(arr){
  var odd=[],even=[];

  for (let item of arr) {
    if (item % 2 == 0) {
      even.push(item);
    } else {
      odd.push(item);
    }
  }

  return [odd,even];
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Mirror Byte](#)

JavaScript:

```
function mirrorByte(byteToMirror) {
  byteToMirror = byteToMirror.toString(2);
  byteToMirror = (" " + byteToMirror).padStart(8, '0');
  var byteMirrored = (byteToMirror).split("").reverse().join(""); //mirroring code here

  return parseInt(byteMirrored,2);
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Double value every next call](#)

PHP:

```
class A
{
  static $value = 0.5;
  public static function getNumber(): int
  {
    self::$value = self::$value*2;
    return self::$value;
  }
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Hello new meta-class!](#)

Ruby:

```
module Foo
  def self.const_missing(name)
    "Hello, " + name.id2name
  end
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find the smallest power higher than a given a value](#)

Ruby:

```
def find_next_power(val, pow_)
  intermediate_value = ((val * 1.0) ** (1.0/pow_)).ceil

  if (intermediate_value ** pow_) == val
    intermediate_value += 1
  end
end
```

```
(intermediate_value ** pow_).floor
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sort Out The Men From Boys](#)

Ruby:

```
def men_from_boys(arr)
  evens = []
  odds = []

  arr.each {|item|
    if item % 2 == 0
      evens.push item
    else
      odds.push item
    end
  }

  evens.uniq.sort + odds.sort.uniq.reverse
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function menFromBoys($arr) {
    $evens = [];
    $odds = [];

    foreach($arr as $item) {
        if ($item % 2 == 0) {
            array_push($evens, $item);
        } else {
            array_push($odds, $item);
        }
    }
    sort($evens);
    rsort($odds);

    $evens = array_unique($evens);
    $odds = array_unique($odds);

    $ret = [];
    foreach($evens as $item) {
        array_push($ret, $item);
    }

    foreach($odds as $item) {
        array_push($ret, $item);
    }

    return $ret;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Bumps in the Road](#)

Ruby:

```
def bump(x)
  x = x.gsub /_/ , ""
  x.length <= 15 ? 'Woohoo!' : 'Car Dead'
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Char Code Calculation](#)

Ruby:

```
def calc(s)
  puts "s:" + s.to_s
  char_code_number = ""
  char_code_number_without_7 = ""

  s.each_char{|char|
    char_code_number = char_code_number + char.ord.to_s
  }

  char_code_number_without_7 = char_code_number.gsub /7/, "1"
```

```

puts char_code_number_without_7
puts char_code_number

sum1 = 0
sum2 = 0

char_code_number.each_char{|c|
  sum1 = sum1 + c.to_i
}

char_code_number_without_7.each_char{|c|
  sum2 = sum2 + c.to_i
}

sum1 - sum2
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find the calculation type](#)

JavaScript:

```

function calcType(a, b, res) {
  if (a + b == res) {
    return "addition"
  } if (a * b == res) {
    return "multiplication"
  } if (a / b == res) {
    return "division"
  } if (a - b == res) {
    return "subtraction"
  }
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```

def calc_type(a, b, res)
  if (a + b == res)
    return "addition"
  elsif (a * b == res)
    return "multiplication"
  elsif (a / b == res)
    return "division"
  elsif (a - b == res)
    return "subtraction"
  end
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```

export function calcType(a: number, b: number, res: number): string {
  if (a + b == res) {
    return "addition";
  } if (a * b == res) {
    return "multiplication";
  } if (a / b == res) {
    return "division";
  } if (a - b == res) {
    return "subtraction";
  }
  return "";
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Which are in?](#)

JavaScript:

```

function inArray(array1,array2){
  let results = [];
  for (let searchedString of array1) {
    for (let itemHaystack of array2) {
      if (itemHaystack.indexOf(searchedString) > -1 && results.indexOf(searchedString) == -1) {
        results.push(searchedString);
      }
    }
  }
  results.sort()
  return results
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export function inArray(a1: string[], a2: string[]): string[] {
  let results = [];
  for (let searchedString of a1) {
    for (let itemHaystack of a2) {
      if (itemHaystack.indexOf(searchedString) > -1 && results.indexOf(searchedString) == -1) {
        results.push(searchedString);
      }
    }
  }
  results.sort();
  return results;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Printing Array elements with Comma delimiters](#)

JavaScript:

```
function printArray(array){
  let ret = ""
  for (let i of array) {
    ret += i + ","
  }
  return ret.slice(0, ret.length-1)
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

```
function printArray(array){
  let ret = ""
  for (let i of array) {
    ret += i + ","
  }
  console.log(ret.slice(0, ret.length-1))
  return ret.slice(0, ret.length-1)
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export function printArray(array:any[]){
  let ret:String = "";
  for (let i of array) {
    ret += String(i) + ",";
  }
  return ret.slice(0, ret.length-1);
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Filling an array \(part 1\)](#)

JavaScript:

```
const arr = N =>{
  cont = 0
  ret = []
  while (cont < N) {
    ret.push(cont)
    cont++
  }
  return ret
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export const arr = (n: number = 0): number[] => {
  let cont:number = 0;
  let ret:number[] = [];
  while (cont < n) {
    ret.push(cont);
  }
}
```

```

    cont++;
  }
  return ret;
};

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Grasshopper - Basic Function Fixer](#)

JavaScript:

```

function addFive(num) {
  var total = num + 5
  return total
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```

def addFive(num)
  num + 5
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```

export const addFive = (num : number) : number => {
  let total = num + 5;
  return total;
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Exes and Ohs](#)

JavaScript:

```

function XO(str) {
  let count0 = 0;
  let countX = 0;

  for (c of str) {
    if (c == "o" || c == "0") {
      count0 = count0 + 1;
    } else if (c == "x" || c == "X") {
      countX = countX + 1;
    }
  }
  return count0 == countX;
}

```

- 3 years ago
- [Refactor](#)

TypeScript:

```

export function xo(str: string) {
  let count0 = 0;
  let countX = 0;

  for (let c of str) {
    if (c == "o" || c == "0") {
      count0 = count0 + 1;
    } else if (c == "x" || c == "X") {
      countX = countX + 1;
    }
  }
  return count0 == countX;
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Convert a string to an array](#)

JavaScript:

```

function stringToArray(string){
  return string.split(" ")
}

```

}

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def string_to_array(string)
  string.split(" ")
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export function stringToArray(s: string): string[] {
  return s.split(" ");
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Computer problem series #1: Fill the Hard Disk Drive](#)

JavaScript:

```
function save(sizes, hd) {
  let sum = 0;
  let cont = 0;
  for (let fileSize of sizes) {
    sum = sum + fileSize;
    if (sum > hd) {
      break;
    }
    cont = cont + 1;
  }
  return cont;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export function save(sizes: number[], hd: number) {
  let sum = 0;
  let cont = 0;
  for (let fileSize of sizes) {
    sum = sum + fileSize;
    if (sum > hd) {
      break;
    }
    cont = cont + 1;
  }
  return cont;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Is n divisible by x and y?](#)

Ruby:

```
def is_divisible(n,x,y)
  n % x == 0 && n % y == 0;
end
```

- 5 years ago
- [Refactor](#)

```
def is_divisible(n,x,y)
  r1 = n % x
  r2 = n % y
  r1 == 0 and r2 == 0
end
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

C:

```
#include <stdbool.h>
```



```
bool isDivisible(int n, int x, int y) {
    return n % x == 0 && n % y == 0;
}
```

- 5 years ago
- [Refactor](#)

```
#include <stdbool.h>
```

```
bool isDivisible(int n, int x, int y) {
    return (n%y == 0 && n % x == 0);
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

```
#include <stdbool.h>
```

```
bool isDivisible(int n, int x, int y) {
    int r1 = n % x;
    int r2 = n % y;

    return r1 == 0 && r2 == 0;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

C#:

```
public class DivisibleNb {
    public static bool isDivisible(long n, long x, long y) {
        return n % x == 0 && n % y == 0;
    }
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function isDivisible(n, x, y) {
    return (n%x == 0 && n%y == 0);
}
```

- 4 years ago
- [Refactor](#)

```
function isDivisible(n, x, y) {
    return n % y == 0 && n % x == 0
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Python:

```
def is_divisible(n,x,y):
    return n % x == 0 and n % y == 0;
```

- 5 years ago
- [Refactor](#)

```
def is_divisible(n,x,y):
    return (n%y == 0 and n % x == 0 );
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Java:

```
public class DivisibleNb {
    public static boolean isDivisible(long n, long x, long y) {
        return n % x == 0 && n % y == 0;
    }
}
```

- 5 years ago
- [Refactor](#)

```
public class DivisibleNb {
    public static boolean isDivisible(long n, long x, long y) {
        return n % x == 0 && n % y == 0;
    }
}
```

- 5 years ago

- [Refactor](#)

```
public class DivisibleNb {
    public static boolean isDivisible(long n, long x, long y) {
        return (n%y == 0 && n % x == 0);
    }
}
```

- 5 years ago
- [Refactor](#)

CoffeeScript:

```
isDivisible = (n, x, y) -> n % x ==0 && n % y ==0;
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Groovy:

```
class Kata {
    static def isDivisible(n, x, y) {
        n % x == 0 && n % y == 0
    }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export function isDivisible(n:number, x:number, y:number):boolean {
    return n % y == 0 && n % x == 0;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Pyramid Array](#)

JavaScript:

```
function pyramid(n) {
    let r = [];
    for (let i = 1; i <=n ; i++) {
        r.push(build(i));
    }
    return r;
}
```

```
function build(n) {
    let r = [];
    for (let i = 1; i <=n ; i++) {
        r.push(1)
    }
    return r;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export function pyramid(n: number) {
    let r = [];
    for (let i = 1; i <=n ; i++) {
        r.push(build(i));
    }
    return r;
}
```

```
export function build(n: number) {
    let r = [];
    for (let i = 1; i <=n ; i++) {
        r.push(1)
    }
    return r;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Plural](#)

JavaScript:

```
function plural(n) {
  return n !== 1;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export function plural(n:number):boolean {
  return n !== 1;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Truthy and Falsy](#)

JavaScript:

```
const truthy = [1,2,3,4,5];
const falsy = [undefined, 0, false, null, ""];
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export const truthy = [1,2,3,4,5];
export const falsy = [undefined, 0, false, null, ""];
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Don't give me five!](#)

JavaScript:

```
function dontGiveMeFive(start, end)
{
  let sum = 0

  let cont = start

  while (cont <= end) {
    if (String(cont).indexOf(5) == -1) {
      sum += 1;
    }
    cont = cont + 1;
  }

  return sum
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export function dontGiveMeFive(start:number, end:number) : number
{
  let sum = 0;

  let cont = start;

  while (cont <= end) {
    if (String(cont).indexOf("5") == -1) {
      sum += 1;
    }
    cont = cont + 1;
  }

  return sum;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Barking mad](#)

Ruby:

```
class Dog
  def initialize(breed)
```

```

    @breed=breed
  end

  def bark()
    "Woof"
  end
end

class Snoop < Dog
end

class Scoobydoo < Dog
end

snoopy=Dog.new("Beagle")

```

```
scoobydoo=Dog.new("Great Dane")
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Calculate Parity bit!](#)

Ruby:

```

def check_parity(parity, bin_str)
  count_1 = 0
  bin_str.each_char{|bit|
    bit = bit.to_i
    count_1 = count_1 + 1 if bit % 2 == 1
  }

  return 1 if ((parity == "even" && count_1 % 2 == 1) || (parity == "odd" && count_1 % 2 == 0))
  return 0
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Pure Functions](#)

TypeScript:

```

type State = {modifier: number}

const state:State = {modifier: 2}

export function solution(arr: number[], options:State) {
  let other: any = Object.assign([],arr);

  for (let i = 0; i < other.length; ++i) {
    other[i] += 2 * options.modifier;
  }

  return other;
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Find the odd int](#)

TypeScript:

```

export const findOdd = (xs: number[]): number => {
  let occurrences = {};

  for (let i of xs) {
    if (occurrences[i] == undefined) {
      occurrences[i] = 1;
    } else {
      occurrences[i]++;
    }
  }

  for (let i in occurrences) {
    if (occurrences[i] % 2 == 1) {
      return parseInt(i);
    }
  }
};

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function findOdd(A) {
  let occurrences = {}

  for (let i of A) {
    if (occurrences[i] == undefined) {
      occurrences[i] = 1;
    } else {
      occurrences[i]++;
    }
  }

  for (let i in occurrences) {
    if (occurrences[i] % 2 == 1) {
      return parseInt(i);
    }
  }
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Beta

[Tinder for Programmers](#)

JavaScript:

```
const rateProfile = (profile, swipeLeft, swipeRight) => {
  if (profile.bio.indexOf("JavaScript") > 0) {
    swipeRight();
  } else {
    swipeLeft();
  }
};
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
import { Profile } from "./preloaded";

export const rateProfile = (profile: Profile, swipeLeft: ()=>void, swipeRight: ()=>void): void => {
  if (profile.bio.indexOf("TypeScript") > 0) {
    swipeRight();
  } else {
    swipeLeft();
  }
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Classy Extentions](#)

JavaScript:

```
class Cat extends Animal {
  speak() {
    return this.name + " meows.";
  }
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Predict your age!](#)

Ruby:

```
def predict_age(* ages)
  sum = 0
  ages.each {|age|
    sum = sum + (age * age)
  }

  result = Math.sqrt(sum).floor

  result/2
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Consecutive strings](#)

PHP:

```
function longestConsec($strarr, $k) {
  if ($k > count($strarr)) {
    return '';
  }

  $longest = '';

  foreach($strarr as $index => $item) {
    $cont = 0;
    $newString = '';
    while ($cont < $k) {
      $newString .= $strarr[$index + $cont];
      $cont++;
    }

    if (mb_strlen($newString) > mb_strlen($longest)) {
      $longest = $newString;
    }
  }

  return $longest;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Insert dashes](#)

Ruby:

```
def insert_dash(num)
  num_string = num.to_s
  sum_to_position = 0
  ret = ""
  previous_odd = false

  num_string.split('').each_with_index {|char_string, index|
    char_integer = char_string.to_i
    if char_integer % 2 == 1
      if previous_odd
        ret = ret + "-" + char_string
      else
        previous_odd = false
        ret += char_string
      end
      previous_odd = true
    else
      previous_odd = false
      ret += char_string
    end
  }
  ret
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Decimal to binary converter](#)

JavaScript:

```
function decToBin(d) {
  if (d == 0) {
    return '0';
  }

  let currentNumber = d;
  let result = "";

  while (currentNumber >= 2) {
    result = ((currentNumber % 2) + "") + result;
    currentNumber = Math.floor(currentNumber / 2)
  }

  if (currentNumber == 1) {
    result = currentNumber + result;
  }

  return result;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Simple Fun #10: Range Bit Counting](#)

Ruby:

```
def range_bit_count(a, b)
  sum = 0
  count = 0
```

```

while a <= b
  number_string = a.to_s(2)
  number_string.each_char{ |n|
    sum = sum + n.to_i
  }
  a = a + 1
end

sum
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Remove the time](#)

PHP:

```

function shortenToDate($longDate) {
  $position = strpos($longDate, 'am');

  if ($position == false) {
    $position = strpos($longDate, 'pm');
  }

  $test = substr($longDate, 0, strlen($longdate) - 5);
  if ($test[strlen($test) - 1] == ",") {
    return substr($test, 0, strlen($test) - 1);
  } else {
    return $test;
  }
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Substituting Variables Into Strings: Padded Numbers](#)

Ruby:

```

def solution(value)
  "Value is " + value.to_s.rjust(5, "0")
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[The old switcheroo](#)

Ruby:

```

def vowel_2_index(string)
  cont = 1
  ret = ""

  string.each_char { |c|
    if c == "a" || c == "e" || c == "i" || c == "o" || c == "u" || c == "A" || c == "E" || c == "I" || c == "O" || c == "U"
      ret += cont.to_s
    else
      ret += c
    end

    cont = cont + 1
  }

  ret
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Alphabet symmetry](#)

PHP:

```

define('INITIAL', 96);

function solve($arr) {
  $arr = toLower($arr);
  $cont = 1;
  $ret = [];

  foreach($arr as $item) {
    $total = 0;
    $cont = 0;
    while ($cont < strlen($item) + 1) {
      if (ord(substr($item, $cont - 1, 1)) == $cont + INITIAL) {

```

```

    $total++;
  }
  $cont++;
}
$ret[] = $total;
}

return $ret;
}

function toLower($arr) {
  $ret = [];

  foreach($arr as $item) {
    $ret[] = strtolower($item);
  }

  return $ret;
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Maximum Gap \(Array Series #4\)](#)

PHP:

```

function maxGap($nums) {
  $maxGap = 0;
  sort($nums);
  $previous = null;

  foreach($nums as $num) {
    if (! is_null($previous)) {
      if ($maxGap < $num - $previous) {
        $maxGap = $num - $previous;
      }
    }
    $previous = $num;
  }

  return $maxGap;
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Numbers to Letters](#)

PHP:

```

function switcher($arr)
{
  $ret = '';
  foreach ($arr as $item) {
    $ret .= chr(- ($item-123));
  }
  return str_replace("-", "!",
    str_replace("_", "?",
      str_replace("^", " ", $ret)
    )
  );
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Coding Meetup #14 - Higher-Order Functions Series - Order the food](#)

JavaScript:

```

function orderFood(list) {
  let resp = {};
  let ret = {};

  for (let item of list) {
    if (resp[item.meal] == undefined) {
      resp[item.meal] = 1;
    } else {
      resp[item.meal]++;
    }
  }

  return resp
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu



[Unique In Order](#)

Ruby:

```
def unique_in_order(iterable)
  ret = []
  iterable2 = iterable

  if iterable2.is_a? Array
    iterable2 = iterable.join ''
  end

  iterable2.each_char {|char|
    unless ret[-1] == char || ret[-1] == char.to_i # ok, isn't perfect, but to this Kata tests is ok
      if iterable[0].is_a? Integer
        ret.push char.to_i
      else
        ret.push char
      end
    end
  }
  ret
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

```
def unique_in_order(iterable)
  ret = []
  iterable2 = iterable

  if iterable2.is_a? Array
    iterable2 = iterable.join ''
  end

  iterable2.each_char {|char|
    unless ret[-1] == char || ret[-1] == char.to_i
      if iterable[0].is_a? Integer
        ret.push char.to_i
      else
        ret.push char
      end
    end
  }
  ret
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find the stray number](#)

Ruby:

```
def stray (numbers)
  stray = []
  previous = []

  numbers.each {|number|
    unless previous.include? number
      previous.push number
      stray.push number
    else
      stray.delete number
    end
  }

  stray.first
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Smallest value of an array](#)

Ruby:

```
def find_smallest(numbers,to_return)
  if to_return == "value"
    numbers.sort!
    return numbers[0]
  else
    minor = 999999999
    numbers.each_with_index {|number, index|
      if number < minor
        minor = number
      end
    }
    return numbers.index minor
  end
end
```

- 2 years ago

- [Refactor](#)
- [Discuss](#)

7 kyu

[Strong Number \(Special Numbers Series #2\)](#)

Ruby:

```
def strong_num(n)
  sum = 0
  n.to_s.each_char { |char_|
    sum = sum + factorial(char_.to_i)
  }
  sum == n ? "STRONG!!!!" : "Not Strong !!"
end

def factorial n
  sum = 1

  while n > 1
    sum = sum * n
    n = n - 1
  end

  sum
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Narcissistic Numbers](#)

Ruby:

```
def is_narcissistic(n)
  sum = 0
  ns = n.to_s
  power = ns.length

  ns.each_char { |char|
    sum += char.to_i ** power
  }

  sum == n ? true : false
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Write Number in Expanded Form](#)

Ruby:

```
def expanded_form(num)
  ret = ""
  multiplier = 1

  num.to_s.reverse.each_char{ |char|
    digit = ((char % 10).to_i * multiplier).to_s

    if digit != "0"
      ret = digit + " + " + ret
    end

    multiplier = multiplier * 10
  }

  ret[0..-4]
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Training JS #3: Basic data types--String](#)

JavaScript:

```
var a1="A",a2="a",b1="B",b2="b",c1="C",c2="c",d1="D",d2="d",e1="E",e2="e",n1="N",n2="n"
function Dad(){
  //select some variable to combine "Dad"
  return d1 + a2 + d2;
}
function Bee(){
  //select some variable to combine "Bee"
  return b1 + e2 + e2;
}
function banana(){
  //select some variable to combine "banana"
  return b2 + a2 + n2 + a2 + n2 + a2;
}
```

```
//answer some questions if you finished works above
function answer1(){
  //the answer should be "yes" or "no"
  return "no";
}
function answer2(){
  //the answer should be "yes" or "no"
  return "no";
}
function answer3(){
  //the answer should be "yes" or "no"
  return "yes";
}
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Training JS #5: Basic data types--Object](#)

JavaScript:

```
function animal(obj){
  if (obj.name != undefined && obj.color != undefined && obj.legs != undefined) {
    return "This " + obj.color + " " + obj.name + " has " + obj.legs + " legs.";
  }

  return false;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Training JS #4: Basic data types--Array](#)

JavaScript:

```
function getLength(arr){
  //return length of arr
  return arr.length;
}
function getFirst(arr){
  //return the first element of arr
  return arr[0];
}
function getLast(arr){
  //return the last element of arr
  return arr[arr.length - 1];
}
function pushElement(arr){
  arr.push("el");
  //push el to arr

  return arr
}
function popElement(arr){
  //pop an element from arr
  arr.pop();

  return arr;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Reverser](#)

Ruby:

```
def reverser(number)
  number.to_s.reverse.to_i
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Squares sequence](#)

Ruby:

```
def squares(x, n)
  return [] if n <= 0
  ret = [x]
  cont = 1

  while (cont < n)
    x = x ** 2
```

```

    ret.push(x)
    cont = cont + 1
  end

  ret
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[+1 Array](#)

Ruby:

```

def up_array(arr)
  if arr.class != Array || arr.empty?
    return nil
  end

  arr = arr.reverse
  ret = []
  accumulator = 1

  arr.each { |i|
    if i < 0 || i > 9 || i.class == String || i == "!"
      return nil
    end

    value = i + accumulator

    if (value >= 10)
      value = value % 10
    else
      accumulator = 0
    end

    ret.push value
  }

  if accumulator == 1
    ret.push(1)
  end

  ret.reverse
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Draft

[Alternating array index](#)

Python:

```

def array_index(arr):
    cont = 0
    ret = []
    for item in arr:
        if cont % 2 == 0:
            print(item)
            ret.append(item + cont)
        else:
            ret.append(item - cont)
        cont = cont + 1

    return ret

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[SevenAte9](#)

Ruby:

```

def seven_ate9(str)
  ret = ""
  prev = ""

  arr_str = str.split("")

  arr_str.each_with_index{|char, index|
    unless char == "9" and arr_str[index - 1] == "7" and arr_str[index + 1] == "7"
      ret +=char
    end
  }
  ret
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Duck Duck Goose](#)

Ruby:

```
def duck_duck_goose(players, goose)
  goose = (goose) % players.length
  players[goose -1].name
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Who is going to pay for the wall?](#)

Ruby:

```
def who_is_paying(name)
  reduced = name[0..1]
  return name == reduced ? [name] : [name, reduced]
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[By 3, or not by 3? That is the question ...](#)

PHP:

```
function divisibleByThree($str) {
    $sum = 0;
    $split = str_split($str);

    foreach ($split as $item) {
        $sum += (int) $item;
    }

    if ($sum % 3 === 0) {
        return true;
    }

    return false;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Likes Vs Dislikes](#)

Python:

```
def like_or_dislike(lst):
    count_like_in_a_row = 0
    count_dislike_in_a_row = 0
    previous = ""

    for i in lst:
        if i == previous:
            if previous == "Like":
                count_like_in_a_row = count_like_in_a_row + 1
            else:
                count_dislike_in_a_row = count_dislike_in_a_row + 1
        else:
            if i == "Like":
                count_like_in_a_row = 1
            else:
                count_dislike_in_a_row = 1

            previous = i

    print(count_dislike_in_a_row)
    if previous == "Like" and count_like_in_a_row % 2 == 1:
        return "Like"
    elif previous == "Dislike" and count_dislike_in_a_row % 2 == 1:
        return "Dislike"

    return "Nothing"
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sort the Gift Code](#)

Ruby:

```
def sort_gift_code code
  code.split("").uniq.sort.join("")
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Lost numbers](#)

JavaScript:

```
const findAndSumm = (arr1, arr2) => {
  let num1 = 0;
  let num2 = 0;

  while (true) {
    if (typeof arr1 !== "object") {
      if (typeof arr1 === "undefined") {
        arr1 = 0;
      }
      num1 = arr1;
      break;
    } else {
      arr1 = arr1[0];
    }
  }

  while (true) {
    if (typeof arr2 !== "object") {
      if (typeof arr2 === "null") {
        arr2 = 0;
      }
      num2 = arr2;
      break;
    } else {
      arr2 = arr2[0];
    }
  }

  return num1 + num2;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Draft

[New Wordle Order](#)

JavaScript:

```
function wordle(word,guess){
  let guess_array = guess.split("");

  let ret = [];
  for (let index in guess_array) {
    if (guess_array[index] == word[index]) {
      ret.push("green");
    } else if (word.indexOf(guess_array[index]) != -1) {
      ret.push("yellow");
    } else {
      ret.push("black");
    }
  }

  return ret;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Take a Number And Sum Its Digits Raised To The Consecutive Powers And ....;Eureka!!](#)

Ruby:

```
def sum_dig_pow(a, b)
  ret = []
  while a <= b
    cont = 1
    sum = 0

    a.to_s.each_char{ |char|
      sum += char.to_i ** cont
      cont = cont + 1
    }

    ret.push(sum) if sum == a
    a = a + 1
  end
  ret
end
```

- 2 years ago
- [Refactor](#)

- [Discuss](#)

7 kyu

[Length and two values.](#)

JavaScript:

```
function opposite(n, firstValue, secondValue){
  let i = 0;
  let ret = [];

  while (i < n) {
    if (i % 2 == 0) {
      ret.push(firstValue);
    } else {
      ret.push(secondValue);
    }
    i++;
  }

  return ret;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Regex validate PIN code](#)

Ruby:

```
def validate_pin pin
  return false unless (/[0-9]*/.match pin)[0]
  return false unless (/[0-9]*/.match pin)[0] == pin
  size = pin.strip.size
  size = size + 1 if pin.to_i < 0
  return false if pin.to_i == 0 and pin != "0000" and pin != "000000"
  return true if size==4 || size==6

  false
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Initialize my name](#)

JavaScript:

```
function initializeNames(name){
  let ret = ""
  let parts = name.split(" ")

  for (let index in parts) {
    if ((index != 0) && (index != parts.length - 1)) {
      ret = ret + parts[index][0].toUpperCase() + ". "
    } else {
      ret += parts[index] + " "
    }
  }
  ret = ret.trim()
  return ret
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Interview Question \(easy\)](#)

Ruby:

```
def get_letters(city)
  city = city.downcase.gsub(/\s*/, "")
  asterisks = {}

  city.each_char { |char|
    unless asterisks[char].nil?
      asterisks[char] = asterisks[char] + "*"
    else
      asterisks[char] = ":*"
    end
  }

  ret = ""
  asterisks.each_with_index{|asterisks, index|
    ret += asterisks[0] + asterisks[1] + ", "
  }
  ret[0..-2]
end
```

- 2 years ago

- [Refactor](#)
- [Discuss](#)

7 kyu

[Minimum Steps \(Array Series #6\)](#)

PHP:

```
function minimumSteps($nums, $value) {
    sort($nums);
    $cont = 0;
    $sum = $nums[0] + $nums[1];
    $cont = 0;
    echo $sum;
    var_dump($nums);

    while ($sum <= $value) {
        echo "x";
        if ($cont == 0) {
            $cont = 1;
        }
        if ($cont > 0 && $sum == $value) {
            break;
        }
        $cont++;
        $sum += $nums[$cont];
    }

    return $cont;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Thinkful - List Drills: Longest word](#)

PHP:

```
function longest($words) {
    $longest = 0;

    foreach($words as $word) {
        $length = strlen($word);

        if ($length > $longest) {
            $longest = $length;
        }
    }

    return $longest;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[UEFA EURO 2016](#)

Ruby:

```
def uefa_euro_2016(teams, scores)
    if scores[0] == scores[1]
        return "At match " + teams[0] + " - " + teams[1] + ", teams played draw."
    elsif scores[0] > scores[1]
        return "At match " + teams[0] + " - " + teams[1] + ", " + teams[0] + " won!"
    else
        return "At match " + teams[0] + " - " + teams[1] + ", " + teams[1] + " won!"
    end
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Coding Meetup #2 - Higher-Order Functions Series - Greet developers](#)

PHP:

```
function greet_developers($a) {
    foreach ($a as &$item) {
        $item['greeting'] = 'Hi ' . $item['first_name'] . ', what do you like the most about ' . $item['language'] . '?';
    }
    return $a;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Fix the Bugs \(Syntax\) - My First Kata](#)



PHP:

```
function my_first_kata($a, $b) {
    if ((!is_int($a) and !is_float($a)) or (!is_int($b) and !is_float($b))) {
        return false;
    } else {
        return $a % $b + $b % $a;
    }
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Greet Me](#)

PHP:

```
function greet($name) {
    return "Hello " . ucfirst(strtolower($name)) . "!";
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Leonardo Dicaprio and Oscars](#)

Ruby:

```
def leo(oscar)
    if oscar == 88
        ret = "Leo finally won the oscar! Leo is happy"
    elsif oscar == 86
        ret = "Not even for Wolf of wallstreet?!"
    elsif oscar < 88
        ret = "When will you give Leo an Oscar?"
    else
        ret = "Leo got one already!"
    end

    ret
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Triangular Treasure](#)

Ruby:

```
# Return the nth triangular number
def triangular( n )
    return 0 if n < 0

    cont = 1
    ret = 0
    i = 0

    while cont <= n
        i += 1
        cont += 1
        ret += i
    end

    ret
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Classy Classes](#)

Ruby:

```
class Person
    def initialize name, age
        @name = name
        @age = age
    end

    def info
        "#{@name}'s age is #{@age}"
    end
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Regexp Basics - is it a digit?](#)

Ruby:

```
class String
  def digit?
    return true if self == "0"
    self.to_i > 0 && self.size === 1
  end
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Arrays Similar](#)

JavaScript:

```
function arraysSimilar(arr1, arr2) {
  arr1 = arr1.sort()
  arr2 = arr2.sort()

  for (let i in arr2) {
    if (arr1[i] !== arr2[i]) {
      return false;
    }
  }

  return true;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Shifty Closures](#)

JavaScript:

```
var greet_abe = function() {
  let name = 'Abe';
  return "Hello, " + name + '!';
};

var greet_ben = function() {
  let name = 'Ben';
  return "Hello, " + name + '!';
};
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Mr. Freeze](#)

JavaScript:

```
// mark the MrFreeze object instance as frozen
Object.freeze(MrFreeze);
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Playing with cubes I](#)

Ruby:

```
# Code the Cube ^-^
# Build your Cube without using the initialize function
# Your cube needs the following:
# side = an integer representing the length of the side of the cube
# get_side = a function to return side
# set_side = a function accepting an int; set side to that int

class Cube
  @side = 0

  def set_side side
    @side = side
  end

  def get_side
    return @side.nil? ? 0 : @side
  end
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Ordered Count of Characters](#)

Ruby:

```
def ordered_count(str)
  str_array = str.split('')
  pre_ret = []
  ret = []
  count = []

  str.each_char{|char|
    unless pre_ret.include? char
      pre_ret.push char
      count.push str_array.count char
    end
  }
  count.each_with_index{|n, index|
    ret.push [pre_ret[index], n]
  }

  ret
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Welcome to the City](#)

Ruby:

```
def say_hello(name, city, state)
  'Hello, ' + name.join(" ") + "! Welcome to " + city + ", " + state + '!'
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Contamination #1 -String-](#)

Ruby:

```
def contamination(text, char)
  return "" if text.empty? || char.empty?

  ret = ""

  text.each_char{|c|
    ret = ret + char
  }
  ret
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Disarium Number \(Special Numbers Series #3\)](#)

Ruby:

```
def disarium_number(n)
  n = n.to_s

  sum = 0
  i = 1
  n.each_char{|char|
    sum += char.to_i ** i
    i = i + 1
  }

  sum.to_s == n ? "Disarium !!" : "Not !!"
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Exclamation marks series #17: Put the exclamation marks and question marks on the balance - are they balanced?](#)

Ruby:

```
def balance(left, right)
  s1 = 0
```

```

s2 = 0

left.each_char { |char|
  if char == "?"
    s1 += 3
  else
    s1 += 2
  end
}

right.each_char { |char|
  if char == "?"
    s2 += 3
  else
    s2 += 2
  end
}

if s1 > s2
  return "Left"
elsif s1 < s2
  return "Right"
end
"Balance"
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Is every value in the array an array?](#)

PHP:

```

function arr_check(array $a): bool {
  foreach ($a as $item) {
    if (gettype($item) != "array") {
      return false;
    }
  }
  return true;
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Exclamation marks series #5: Remove all exclamation marks from the end of words](#)

PHP:

```

function remove(string $s): string {
  $arrayString = explode(' ', $s);

  $cont = 0;
  while (count($arrayString) > $cont) {
    if ($arrayString[$cont][-1] == "!") {
      $arrayString[$cont] = substr($arrayString[$cont], 0, -1);
    } else {
      $cont++;
    }
  }

  return implode(' ', $arrayString);
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Age Range Compatibility Equation](#)

PHP:

```

function datingRange($age) {
  $min = 0;
  $max = 0;

  if ($age <= 14) {
    $min = $age - 0.10 * $age;
    $max = $age + 0.10 * $age;
  } else {
    $min = $age/2 + 7;
    $max = ($age - 7) * 2;
  }

  return floor($min) . '-' . floor($max);
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Add new item \(collections are passed by reference\)](#)

Ruby:

```
def add_extra(list_of_numbers)
  lon = list_of_numbers.dup
  lon.unshift(1)
  lon
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Training JS #1: create your first JS function and print "Hello World!"](#)

JavaScript:

```
function helloWorld() {
  var str = "que bosta...";
  console.log("Hello World!");
  return str;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find the lucky numbers](#)

PHP:

```
function filter_lucky(array $a): array {
  $ret = [];
  foreach ($a as $item) {
    if (strpos((string) $item, '7') !== false) {
      $ret[] = $item;
    }
  }

  return $ret;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Exclamation marks series #13: Count the number of exclamation marks and question marks, return the product](#)

PHP:

```
function product(string $s): int {
  $lengthTotal = strlen($s);
  $lengthExclamation = $lengthTotal - strlen(str_replace('!', '', $s));
  $lengthQuotes = $lengthTotal - strlen(str_replace('?', '', $s));

  return $lengthExclamation * $lengthQuotes;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Calculate mean and concatenate string](#)

PHP:

```
function mean(array $a): array {
  $sum = 0;
  $string = '';

  foreach($a as $item) {
    $sum += (float) $item;

    if (!(float) $item == $item) {
      $string .= $item;
    }
  }

  return [$sum / 10, $string];
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Number of Decimal Digits](#)

Java:

```
public class DecTools {
  public static int Digits(long n) {
    return String.valueOf(n).length();
  }
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Negation of a Value](#)

Dart:

```
bool negationValue(String str, bool val) {
  if (str.length % 2 == 0) {
    return val;
  }
  return !val;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Divide and Conquer](#)

Ruby:

```
def div_con(x)
  sum = 0
  minus = 0

  x.each {|i|
    if i.is_a? Numeric
      sum += i
    else
      minus += i.to_i
    end
  }

  sum - minus
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Find Nearest square number](#)

Ruby:

```
def nearest_sq(n)
  return n if Math.sqrt(n) % 1 == 0

  minor = n
  while true
    if Math.sqrt(minor) % 1 == 0
      break
    end
    minor = minor - 1
  end

  major = n
  while true
    if Math.sqrt(major) % 1 == 0
      break
    end
    major = major + 1
  end

  diff_minor = n - minor
  diff_major = major - n

  diff_major <= diff_minor ? major : minor
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Fix your code before the garden dies!](#)

Ruby:

```
def rain_amount(mm)
  if (mm < 40)
    return "You need to give your plant " + (40 - mm).to_s + "mm of water"
  else
    return "Your plant has had more than enough water for today!"
  end
end
```

```
end
end

• 2 years ago
• Refactor
• Discuss
```

7 kyu  
[Spacify](#)

Ruby:

```
def spacify(str)
  ret = ""
  str.each_char { |c|
    ret += c + " "
  }
  ret[0..-2]
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Basic subclasses - Adam and Eve](#)

Ruby:

```
# define your classes
class Human
end
class Man < Human
end
class Woman < Human
end

def god
  [Man.new, Woman.new]
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[String Templates - Bug Fixing #5](#)

Ruby:

```
def build_string(*args)
  string_args = ""
  args.each {|arg|
    string_args += arg + ", "
  }
  string_args = string_args[0..-3]
  "I like " + string_args + "!"
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Unfinished Loop - Bug Fixing #1](#)

Ruby:

```
def create_array(n)
  res=[]
  i=1
  while i<=n
    res+=i
    i = i + 1
  end
  return res
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Retired  
[Playing with Streams: Sum](#)

Java:

```
import java.util.*;

public class Kata {
  public static int sum(List<Integer> list) {
    Integer ret = 0;
    for (Integer item: list) {
      ret += item;
    }
  }
}
```

```

    }
    return ret;
  }
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Nth Smallest Element \(Array Series #4\)](#)

Ruby:

```

def nth_smallest(arr, pos)
  arr.sort[pos-1]
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Indexed capitalization](#)

Ruby:

```

def capitalize(s,ind)
  ret = ""

  index = -1
  s.each_char { |c|
    if ind.include? index
      ret += c.capitalize
    else
      ret = ret + c
    end if
    index = index + 1
  }
  ret
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Grasshopper - Combine strings](#)

Ruby:

```

def combine_names first_name, last_name
  first_name + " " + last_name
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find the nth Digit of a Number](#)

Ruby:

```

def find_digit(num, nth)
  num = num.to_s.reverse!
  return -1 if nth < 1
  num.slice(nth - 1,1).to_i
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[The Vowel Code](#)

Ruby:

```

def encode(s)
  s.gsub! /a/, "1"
  s.gsub! /e/, "2"
  s.gsub! /i/, "3"
  s.gsub! /o/, "4"
  s.gsub! /u/, "5"
  puts s
  s
end

```

```

def decode(s)
  s.gsub! /1/, "a"
  s.gsub! /2/, "e"
  s.gsub! /3/, "i"
  s.gsub! /4/, "o"

```



```
s.gsub! /5/, "u"
s
```

end

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Flatten](#)

Ruby:

```
def flatten(array)
  ret = []
  array.each { |item|
    if item.is_a? Array
      item.each { |subitem|
        ret.push subitem
      }
    else
      ret.push item
    end
  }
  ret
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Grader](#)

Ruby:

```
def grader(score)
  if score > 1 || score < 0.6
    return "F"
  elsif score >= 0.9
    return "A"
  elsif score >= 0.8
    return "B"
  elsif score >= 0.7
    return "C"
  elsif score >= 0.6
    return "D"
  end
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Training JS #9: loop statement --while and do..while](#)

JavaScript:

```
function padIt(str,n){
  let turn = "left";

  let cont = 0;
  let ret = str;

  while (cont != n) {
    if (turn == "left") {
      turn = "right";
      ret = "*" + ret;
    } else {
      turn = "left";
      ret = ret + "*";
    }
    cont++;
  }

  return ret;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[What is type of variable?](#)

JavaScript:

```
function type(value) {
  if (value instanceof Array) {
    return 'array';
  }
  if (value instanceof Date) {
    return 'date';
  }
}
```

```

if (value === null) {
  return 'null';
}
return typeof value;
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Greatest common divisor](#)

JavaScript:

```

function mygcd(x,y){
  let cont = 1;
  let common = 0;

  while (cont <= x + 1 && cont <= y + 1) {
    if (x % cont == 0 && y % cont == 0) {
      common = cont;
    }
    cont++;
  }

  return common;
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[BASIC: Making Six Toast.](#)

Ruby:

```

def six_toast(num)
  if num < 6
    return num
  else
    return num - 6
  end
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Redact a Key-Value Pair from a Hash in Ruby – "The Holy Rail" – unquest\(\).](#)

Ruby:

```

def unquest(prommer)
  prommer.delete :quest
  prommer
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Beta

[Album lengths](#)

JavaScript:

```

function albumLength(trackLengths) {
  let hours = 0;
  let minutes = 0;
  let seconds = 0;
  for (let track of trackLengths) {
    let trackData = track.split(":");

    if (!isNaN(seconds)) {
      seconds = seconds + parseInt(trackData[2]);
    }

    if (!isNaN(minutes)) {
      minutes = minutes + parseInt(trackData[1]);
    }

    if (!isNaN(hours)) {
      hours = hours + parseInt(trackData[0]);
    }
  }

  let prevHours = hours;
  let prevSeconds = seconds;

  seconds = seconds % 60;
  let prevMinutes = minutes + Math.floor(parseInt(prevSeconds / 60));
  minutes = prevMinutes % 60;
  hours = hours + Math.floor(parseInt(prevMinutes / 60));
}

```

```

if (hours < 10) {
  hours = '0' + hours;
}
if (minutes < 10) {
  minutes = '0' + minutes;
}
if (seconds < 10) {
  seconds = '0' + seconds;
}

return hours + ":" + minutes + ":" + seconds;
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Percentage of primary color in HEX color](#)

TypeScript:

```

type PrimaryColorName = "red" | "green" | "blue";

// return the two oldest/oldest ages within the array of ages passed in.
// it should return the two ages as a sorted array, youngest age first
export function getPrimaryColorPercentage(color: string, primaryColorName: PrimaryColorName): number {
  if (color.length == 4) {
    color = color.substring(0,2) + "0" + color.substring(2,3) + "0" + color.substring(3,4) + "0"
  }
  let red = parseInt(color.substring(1,3), 16);
  let green = parseInt(color.substring(3,5), 16);
  let blue = parseInt(color.substring(5,7), 16);
  let alpha = parseInt(color.substring(7,9), 16);
  if (isNaN(alpha)) {
    alpha = 0;
  }
  let total = red + green + blue;

  let pctAlpha = Math.round((alpha / 255) * 100) / 100;
  if (pctAlpha == 0) {
    pctAlpha = 1;
  }

  if (primaryColorName == "red") {
    return Math.round((red / total) * 100) * pctAlpha;
  } else if (primaryColorName == "green") {
    return Math.round((green / total) * 100) * pctAlpha;
  } else {
    return Math.round((blue / total) * 100) * pctAlpha;
  }
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Powers of 2](#)

Ruby:

```

def powers_of_two(n)
  ret = []
  while (n > -1)
    ret.push(2**n)
    n = n - 1
  end
  ret.reverse
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```

function powersOfTwo(n){
  let ret = [];
  for (let i=0; i <= n; i++) {
    ret.push(Math.pow(2, i));
  }
  console.log(ret);
  return ret;
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Implement isEmpty function](#)

JavaScript:

```

const isEmpty = (obj) => Object.keys(obj).length == 0

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Semi-Optional](#)

JavaScript:

```
function wrap(value) {
  return {
    value:value
  };
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Most digits](#)

JavaScript:

```
function findLongest(array){
  let seleccionado = 0;

  for (item of array) {
    if (seleccionado === null || item.toString().length > seleccionado.toString().length) {
      seleccionado = item;
    }
  }

  return seleccionado;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def find_longest(arr)
  max_length = 0
  max_item = 0
  arr.each { |item|
    if item.to_s.size > max_length
      max_length = item.to_s.size
      max_item = item
    end
  }
  max_item
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Number-Star ladder](#)

Ruby:

```
def pattern(n)
  current = 1
  ret = ''

  while current <= n
    if current == 1
      ret = ret + "1\n"
      current = current + 1
    else
      ret = ret + "1"

      x = 1
      while x < current
        ret = ret + '*'
        x = x + 1
      end

      ret = ret + current.to_s
      current = current + 1
    end

    if current <= n
      ret = ret + "\n"
    end
  end

  ret
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Remove All The Marked Elements of a List](#)

Ruby:

```
class Array
  def remove_(integer_list, values_list)
    ret = []
    integer_list.each {|number|
      unless values_list.include? number
        ret.push number
      end
    }
    ret
  end
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Password Hashes](#)

Ruby:

```
def pass_hash(str)
  Digest::MD5.hexdigest(str)
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Case Swapping](#)

Ruby:

```
def swap(string)
  ret = ""
  string.split("").each {|letter|
    if letter.ord < 97
      ret = ret + letter.downcase
    else
      ret = ret + letter.upcase
    end
  }
  ret
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[The Office IV - Find a Meeting Room](#)

Ruby:

```
def meeting(rooms)
  rooms.each_with_index {|room, index|
    return index if room == "0"
  }
  'None available!'
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Filter Long Words](#)

Ruby:

```
def filter_long_words(sentence, n)
  ret = []
  sentence.split(" ").each { |word|
    ret.push(word) if word.length > n
  }
  ret
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Array Leaders \(Array Series #3\)](#)

PHP:

```
function arrayLeaders($numbers) {
    $ret = [];
    $total = count($numbers);

    foreach($numbers as $index => $number) {
        $sum = 0;
        for ($i = $index + 1; $i < $total;$i++) {
            $sum += $numbers[$i];
        }

        if ($number > $sum) {
            array_push($ret, $number);
        }
    }
    return $ret;
}
```

```
// your code here
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sum of all arguments](#)

PHP:

```
function sum() {
    $sum = 0;

    foreach (func_get_args() as $arg) {
        $sum = $sum + $arg;
    }

    return $sum;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Row Weights](#)

Ruby:

```
def row_weights(array)
    t1 = 0
    t2 = 0

    array.each_with_index { |item, index|
        if index % 2 == 0
            t1 += item
        else
            t2 += item
        end
    }

    [t1, t2]
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[All Star Code Challenge #14 - Find the median](#)

Ruby:

```
def median(array)
    array.sort!
    lp2 = array.length % 2
    if (lp2 == 0)
        return (array[(array.length / 2) - 1] + array[(array.length / 2)]) / 2.0
    end
    array[(array.length / 2.0)]
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

```
def median(array)
    array = array.sort!
    lp2 = array.length % 2
    if (lp2 == 0)
        return (array[(array.length / 2) - 1] + array[(array.length / 2)]) / 2.0
    end
    array[(array.length / 2.0)]
end
```

- 2 years ago

- [Refactor](#)
- [Discuss](#)

```
def median(array)
  array.sort!

  if array.length % 2 == 1
    return array[(array.length / 2)]
  end

  return (array[(array.length / 2)] + array[(array.length / 2) - 1]) / 2.0
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[All Star Code Challenge #18](#)

Ruby:

```
def str_count(word, letter)
  counter = 0
  word.split("").each {|l|
    counter = counter + 1 if letter == l
  }
  counter
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

```
def str_count(word, letter)
  cont = 0

  word.each_char { |l|
    if l == letter
      cont = cont + 1
    end
  }

  cont
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Sum of Multiples](#)

Ruby:

```
def sum_mul(n, m)
  puts "n:" + n.to_s
  puts "m:" + m.to_s
  current = n
  return "INVALID" if n <= 0 || m <= 0
  sum = 0
  while current < m
    sum = sum + current
    current = current + n
  end
  return "INVALID" if sum == 0
  sum
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Scoring Tests](#)

Ruby:

```
def score_test(tests, right, omit, wrong)
  answers = []
  answers.push(0)
  answers.push(0)
  answers.push(0)

  tests.each {|answer_result|
    answers[answer_result] += 1
  }

  answers[0] * right + answers[1] * omit - answers[2] * wrong
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Compress sentences](#)

JavaScript:

```
function compress(sentence) {
  let words = sentence.split(" ");
  sentence = "";
  for (let word of words) {
    sentence = sentence + " " + word.toLowerCase();
  }
  sentence = sentence.slice(1, sentence.length);

  words = sentence.split(" ");
  let ret = "";

  let wordsIndex = [...new Set(words)]

  let count = 0;
  for (let word of words) {
    ret = ret + wordsIndex.indexOf(word);
  }
  return ret;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Ones' Complement](#)

Ruby:

```
def ones_complement(binary_number)
  ret = ""
  binary_number.split("").each { |i|
    if i == "0"
      ret = ret + "1"
    else
      ret = ret + "0"
    end
  }
  ret
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Move 10](#)

JavaScript:

```
function moveTen(s){
  let sArray = s.split("");
  let ret = "";
  for (let char of sArray) {
    let ord = char.charCodeAt(0);
    let plus10 = ord + 10;
    if (plus10 > 122) {
      plus10 = plus10 - 26;
    }

    ret = ret + String.fromCharCode(plus10);
  }

  return ret;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Function 3 - multiplying two numbers](#)

Ruby:

```
def multiply a, b
  a * b
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function multiply($a, $b) {
  return $a * $b;
}
```



- 3 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function multiply(a, b) {
  return a * b;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

```
function multiply(a, b) {
  return a * b;
}
```

- 3 years ago
- [Refactor](#)

C:

```
int multiply(int x, int y) {
  return x * y;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Java:

```
public class Kata {
  public static int multiply(int num1, int num2) {
    return num1 * num2;
  }
}
```

- 2 years ago
- [Refactor](#)

```
public class Kata {
  public static int multiply(int num1, int num2) {
    return num1 * num2;
  }
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Python:

```
#your code here
def multiply(a, b):
  return a * b
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

C#:

```
public class Kata
{
  public static int Multiply(int a, int b)
  {
    return a * b;
  }
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Swift:

```
func multiply(_ a: Double, _ b: Double) -> Double {
  return a * b;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[What comes after?](#)

Ruby:

```
def comes_after(str, letter)
  ret = ""
  str.split("").each_with_index{|l, key|
    if l.upcase == letter.upcase
      if key + 1 < str.length and ((str[key+ 1].ord >= 97 and str[key+ 1].ord <=122) or (str[key+ 1].ord >= 65 and str[key+ 1].ord <=90))
        ret = ret + str[key+ 1]
      end
    end
  }
  ret
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Merging sorted integer arrays \(without duplicates\)](#)

Ruby:

```
def merge_arrays(a, b)
  (a + b).sort.uniq
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Help the bookseller !](#)

Ruby:

```
def stockList(listOfArt, listOfCat)
  ret = ""
  accumulator = {}
  listOfCat.each {|category|
    accumulator[category] = 0
  }
  listOfCat.each{|category|
    listOfArt.each{|book|
      if book[0] == category
        value = "0"
        book.each_char {|char|
          if char.ord >=48 && char.ord <= 57
            value = value + char
          end
        }
        accumulator[book[0]] = 0 if accumulator[book[0]] .nil?
        accumulator[book[0]] += value.to_i
      end
    }
  }

  may_ret = false
  accumulator.each_with_index {|value, key|
    ret = ret + "(" + value[0] + " : " + value[1].to_s + ") - "
    if value[1] > 0
      may_ret = true
    end
  }

  if may_ret
    ret[0..-4]
  else
    ""
  end
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Digits explosion](#)

Ruby:

```
def explode(s)
  ret = ""

  s.split("").each {|n|
    ret = (ret + (n * n.to_i)).to_s
  }

  ret
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Remove consecutive duplicate words](#)

Ruby:

```
def remove_consecutive_duplicates(s)
  ret = []
  previous = ""

  s.split(" ").each {|w|
    unless previous == w
      ret.push(w)
    end
    previous = w
  }

  ret.join(" ").strip
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Yield to the Block](#)

Ruby:

```
def compute
  return "Do not compute" unless block_given?
  "Running the block"
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Largest Elements](#)

JavaScript:

```
function largest(n,xs){
  xs.sort((a, b) => a - b);
  xs.reverse()

  let ret = [];

  for (let i = 0; i < n ; i++) {
    ret.push(xs[i]);
  }

  ret = ret.sort((a, b) => a - b);

  return ret;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[KISS - Keep It Simple Stupid](#)

JavaScript:

```
function isKiss( words ){
  words = words.split(" ");
  for (let word of words) {
    if (word.length > words.length) {
      return "Keep It Simple Stupid";
    }
  }

  return "Good work Joe!";
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Password maker](#)

Ruby:

```
def make_password(phrase)
  phrase = phrase.gsub(/[iI]/, "1")
  phrase = phrase.gsub(/[oO]/, "0")
  phrase = phrase.gsub(/[sS]/, "5")

  ret = ""
  phrase.split(" ").each{|w|
    ret = ret + w[0]
  }
  ret
end
```

- 2 years ago

- [Refactor](#)
- [Discuss](#)

8 kyu

[Did she say hallo?](#)

Ruby:

```
def validate_hello(greeting)
  return true if greeting.downcase.match /hello/
  return true if greeting.downcase.match /ciao/
  return true if greeting.downcase.match /salut/
  return true if greeting.downcase.match /hallo/
  return true if greeting.downcase.match /hola/
  return true if greeting.downcase.match /ahoj/
  return true if greeting.downcase.match /czesc/

  false
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[For Twins: 2. Math operations](#)

Ruby:

```
def ice_brick_volume(radius, bottle_length, rim_length)
  l = 2*radius/Math.sqrt(2)
  (1 * l * (bottle_length - rim_length)).round
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Snake\\_Casify\\_Keys](#)

Python:

```
import re

def snake_casify(dictionary):
  ret = {}
  for key in dictionary:
    result = re.findall("[A-Z]",key)

    tmp = key
    for i in result:
      tmp = tmp.replace(i, "_" + chr(ord(i) + 32))

    ret[tmp] = dictionary[key]

  return ret
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Pair Zeros](#)

Ruby:

```
def pair_zeros(arr)
  ret = []
  num_zero = 0
  arr.each {|i|
    puts i
    ret.push(i) unless num_zero % 2 == 1 && i == 0
    num_zero = num_zero + 1 if i == 0
  }
  ret
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Get number from string](#)

Ruby:

```
def get_number_from_string(s)
  r = ""
  s.each_char{|c|
    if c.ord >=48 && c.ord <= 57
      r = r + c.to_s
    end
  }
end
```

```

    r.to_i
  end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Rock Paper Scissors Lizard Spock](#)

JavaScript:

```

function rpsls(p1,p2){
  if (p1=="rock" && (p2 == "lizard" || p2 == "scissors")) {
    return "Player 1 Won!";
  } else if (p2=="rock" && (p1 == "lizard" || p1 == "scissors")) {
    return "Player 2 Won!";
  } else if (p1=="paper" && (p2 == "rock" || p2 == "spock")) {
    return "Player 1 Won!";
  } else if (p2=="paper" && (p1 == "rock" || p1 == "spock")) {
    return "Player 2 Won!";
  } else if (p1=="scissors" && (p2 == "paper" || p2 == "lizard")) {
    return "Player 1 Won!";
  } else if (p2=="scissors" && (p1 == "paper" || p1 == "lizard")) {
    return "Player 2 Won!";
  } else if (p1=="lizard" && (p2 == "paper" || p2 == "spock")) {
    return "Player 1 Won!";
  } else if (p2=="lizard" && (p1 == "paper" || p1 == "spock")) {
    return "Player 2 Won!";
  } else if (p1=="spock" && (p2 == "scissors" || p2 == "rock")) {
    return "Player 1 Won!";
  } else if (p2=="spock" && (p1 == "scissors" || p1 == "rock")) {
    return "Player 2 Won!";
  } else if (p1 == p2) {
    return "Draw!";
  }
}

return "Player 2 Won!";
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Identical Elements](#)

Ruby:

```

def duplicate_elements(m, n)
  m.each {|item|
    return true if n.include? item
  }
  false
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Pillars](#)

Ruby:

```

def pillars(num_of_pillars, distance, width)
  dist = (num_of_pillars - 2) * width + distance * (num_of_pillars - 1) * 100
  return 0 if dist < 0
  dist
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Strings, strings, strings \(Easy\)](#)

JavaScript:

```

// Recover toString() here :)
String.prototype.toString = function() {
}

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Convert a Boolean to a String](#)

Ruby:

```
def boolean_to_string(b)
  b == true ? "true" : "false"
end
```

- 2 years ago
- [Refactor](#)

```
def boolean_to_string(b)
  if b == true
    "true"
  else
    "false"
  end
end
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Rearrange Number to Get its Maximum](#)

Ruby:

```
def max_redigit(num)
  return 321 if num == 321
  return nil if num < 1 or num.to_s.size != 3
  num.to_s.split("").sort.reverse.join("").to_i
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Grasshopper - Check for factor](#)

VB:

```
Public Module Kata
  Public Function CheckForFactor(ByVal base As Integer, ByVal factor As Integer) As Boolean
    Return base mod factor = 0
  End Function
End Module
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function checkForFactor (base, factor) {
  return base % factor === 0;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def check_for_factor(base, factor)
  base % factor == 0
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Java:

```
public class Kata {
  public static boolean checkForFactor(int base, int factor) {
    return base % factor == 0 ;
  }
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Rock Paper Scissors!](#)

Ruby:

```
def rps(p1, p2)
  if (p1 == p2)
    return 'Draw!'
  elsif ((p1 == 'rock' and p2 == 'scissors') or (p1 == 'scissors' and p2 == 'paper') or (p1 == 'paper' and p2 == 'rock'))
    return 'Player 1 won!'
  else
    return 'Player 2 won!'
  end
end
```

```

end
end

• 4 years ago
• Refactor
• Discuss

def rps(p1, p2)
  return "Draw!" if p1 == p2
  return "Player 1 won!" if (p1 == "scissors" and p2 == "paper") || (p1 == "paper" and p2 == "rock") || (p1 == "rock" and p2 == "scissors")
  return "Player 2 won!"
end

```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

```

def rps(p1, p2)
  return "Draw!" if p1 == p2
  return "Player 1 won!" if (p1 == "scissors" and p2 == "paper") || (p1 == "paper" and p2 == "rock") || (p1 == "rock" and p2 == "scissors")
  return "Player 2 won!" if (p2 == "scissors" and p1 == "paper") || (p2 == "paper" and p1 == "rock") || (p2 == "rock" and p1 == "scissors")
  nil
end

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```

const rps = (p1, p2) => {
  if (p1 == "scissors" && p2 == "rock") {
    return "Player 2 won!";
  }

  if (p1 == 'scissors' && p2 == "paper") {
    return 'Player 1 won!';
  }

  if (p1 == 'scissors' && p2 == "scissors") {
    return 'Draw!';
  }

  if (p1 == 'paper' && p2 == "scissors") {
    return 'Player 2 won!';
  }

  if (p1 == 'paper' && p2 == "rock") {
    return 'Player 1 won!';
  }

  if (p1 == 'paper' && p2 == "paper") {
    return 'Draw!';
  }

  if (p1 == 'rock' && p2 == "paper") {
    return 'Player 2 won!';
  }

  if (p1 == 'rock' && p2 == "scissors") {
    return 'Player 1 won!';
  }

  if (p1 == 'rock' && p2 == "rock") {
    return 'Draw!';
  }
}

```

};

- 6 years ago
- [Refactor](#)
- [Discuss](#)

```

const rps = (p1, p2) => {
  if (p1 == 'rock' && p2 == 'scissors' || p1 == 'paper' && p2 == 'rock' || p1 == 'scissors' && p2 == 'paper') {
    return 'Player 1 won!';
  } else if (p2 == 'rock' && p1 == 'scissors' || p2 == 'paper' && p1 == 'rock' || p2 == 'scissors' && p1 == 'paper') {
    return 'Player 2 won!';
  }
  return 'Draw!';
};

```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

Python:

```

def rps(p1, p2):
  if p1 == p2:
    return "Draw!"

  if (p1 == "scissors" and p2 == "paper") or (p1 == "rock" and p2 == "scissors") or (p1 == "paper" and p2 == "rock"):
    return "Player 1 won!"
  else:
    return "Player 2 won!"

```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

C#:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;

public class Kata
{
    public string Rps(string p1, string p2)
    {
        if (p1 == "paper" && p2 == "rock" || p1 == "scissors" && p2 == "paper" || p1 == "rock" && p2 == "scissors") {
            return "Player 1 won!";
        }
        if (p2 == "paper" && p1 == "rock" || p2 == "scissors" && p1 == "paper" || p2 == "rock" && p1 == "scissors") {
            return "Player 2 won!";
        }

        return "Draw!";
    }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Java:

```
public class Kata {
    public static String rps(String p1, String p2) {
        if (p1 == "scissors") {
            if (p2 == "paper") {
                return "Player 1 won!";
            } else if (p2 == "rock") {
                return "Player 2 won!";
            }

            return "Draw!";
        }

        if (p1 == "paper") {
            if (p2 == "rock") {
                return "Player 1 won!";
            } else if (p2 == "scissors") {
                return "Player 2 won!";
            }

            return "Draw!";
        }

        if (p1 == "rock") {
            if (p2 == "scissors") {
                return "Player 1 won!";
            } else if (p2 == "paper") {
                return "Player 2 won!";
            }

            return "Draw!";
        }

        return null;
    }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

```
public class Kata {
    public static String rps(String p1, String p2) {
        if (p1 == "paper" && p2 == "rock" || p1 == "scissors" && p2 == "paper" || p1 == "rock" && p2 == "scissors") {
            return "Player 1 won!";
        }
        if (p2 == "paper" && p1 == "rock" || p2 == "scissors" && p1 == "paper" || p2 == "rock" && p1 == "scissors") {
            return "Player 2 won!";
        }

        return "Draw!";
    }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function rpc ($p1, $p2) {
    if ($p1 == $p2) {
        return 'Draw!';
    } elseif (($p1 == 'rock' && $p2 == 'scissors') || ($p1 == 'scissors' && $p2 == 'paper') || ($p1 == 'paper' && $p2 == 'rock')) {
        return 'Player 1 won!';
    } else {
        return 'Player 2 won!';
    }
}
```



```

    }
  }
}

• 4 years ago
• Refactor
• Discuss

```

6 kyu  
[Who likes it?](#)

Ruby:

```

def likes(names)
  return "no one likes this" if names.size == 0

  ret = ""

  if names.size == 1
    return names[0] + " likes this"
  elsif names.size == 2
    return names[0] + " and " + names[1] + " like this"
  elsif names.size == 3
    return names[0] + ", " + names[1] + " and " + names[2] + " like this"
  end
  names[0] + ", " + names[1] + " and " + (names.size - 2).to_s + " others like this" if names.size > 1
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu  
[Pair of gloves](#)

Ruby:

```

def number_of_pairs(gloves)
  totals = {}
  gloves.each { |glove|
    if totals[glove].nil?
      totals[glove] = 1
    else
      totals[glove] = totals[glove] + 1
    end
  }

  total = 0
  totals.each {|item|
    puts item
    total = total + item[1] / 2
  }

  total
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu  
[Hamming Distance](#)

Ruby:

```

def hamming(a, b)
  i = 0
  r = 0

  major_length = a.length > b.length ? a.length : b.length

  while i < major_length
    if a[i] != b[i]
      r = r + 1
    end
    i = i + 1
  end
  r
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Incrementer](#)

Ruby:

```

def incrementer(nums)
  ret = []
  nums.each_with_index {|n, index|
    val = n + index + 1
    while val > 9
      val = val - 10
    end
  }
end

```

```

    ret.push val
  }
  ret
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find the capitals](#)

JavaScript:

```

var capitals = function (word) {
  let i = 0;
  let ret = [];
  while (i <= word.length) {
    let ascii = word.charCodeAt(i);
    if (ascii >= 65 && ascii <=90) {
      ret.push(i);
    }
    i = i + 1;
  }
  return ret;
};

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Regular Ball Super Ball](#)

JavaScript:

```

var Ball = function (t){
  this.ballType = "regular"
  if (typeof t !== "undefined") {
    this.ballType = t;
  }
}

```

new Ball("regular")

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Double Every Other](#)

Ruby:

```

def double_every_other(num_array)
  ret = []
  num_array.each_with_index {|num, index|
    if index % 2 == 1
      ret.push num * 2
    else
      ret.push num
    end
  }
  ret
end

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Check same case](#)

Python:

```

def same_case(a, b):
  if not((ord(a) >= 97 and ord(a) <= 122) or (ord(a) >= 65 and ord(a) <= 90)) or not((ord(b) >= 97 and ord(b) <= 122) or (ord(b) >= 65 and ord(b) <= 90)):
    return -1
  elif ((ord(a) >= 97 and ord(a) <= 122) and (ord(b) >= 97 and ord(b) <= 122)) or (ord(b) >= 65 and ord(b) <= 90) and (ord(a) >= 65 and ord(a) <= 90):
    return 1
  else:
    print(ord(a))
    print(ord(b))
    return 0

```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Draft

[Beginner friendly: Lowercase letters](#)

Ruby:

```
def convert_lower_case(s)
  s.downcase
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Largest Square Inside A Circle](#)

Ruby:

```
def area_largest_square(r)
  d = 2 * r
  l = d / Math.sqrt(2)
  (l*l).round
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Perimeter sequence](#)

Ruby:

```
def perimeter_sequence(a, n)
  4 * a * n
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[getNames\(\)](#)

JavaScript:

```
function getNames(data){
  let retorno = [];

  for (let item of data) {
    retorno.push(item.name);
  }

  return retorno;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Turn with a Compass](#)

Ruby:

```
def direction(facing, turn)
  puts facing
  puts turn
  directions = {
    0 => "N",
    45 => "NE",
    90 => "E",
    135 => "SE",
    180 => "S",
    225 => "SW",
    270 => "W",
    315 => "NW"
  }

  directions_inverted = directions.invert

  directions[(directions_inverted[facing] + turn) % 360]
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Is it a number?](#)

JavaScript:

```
function isDigit(s) {
  let si = parseFloat(s);
  if (si < 1) {
    return true;
  }
}
```

```
return (" " + si).length == s.length;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Powers of i](#)

Ruby:

```
def pofi(n)
  r = n % 4
  return "1" if r == 0
  return "i" if r == 1
  return "-1" if r == 2
  "-i"
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Special Number \(Special Numbers Series #5\)](#)

Ruby:

```
def special_number(n)
  n.to_s.split("").each{ |d|
    d = d.to_i
    return "NOT!!" if d > 5
  }
  "Special!!"
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Driving School Series #2](#)

JavaScript:

```
function cost (mins) {
  if (mins < 60) {
    return 30;
  }

  let firstHour = 30;

  let additionalTime = mins - 60;
  console.log(additionalTime);
  let additionalHalfHour = Math.ceil((additionalTime - 5) / 30);
  console.log(additionalHalfHour);
  console.log( additionalHalfHour * 10 + firstHour)
  return (additionalHalfHour * 10 + firstHour);
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Round by 0.5 steps](#)

JavaScript:

```
function solution(n){
  return Math.round(n * 2) / 2;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Area of an arrow](#)

Ruby:

```
def arrow_area(a, b)
  a = a.to_f
  b = b.to_f
  ((a * b)/4)
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Cat years, Dog years](#)

Ruby:

```
def human_years_cat_years_dog_years(human_years)
  hy = human_years
  cat_years = 0
  dog_years = 0

  if human_years >= 1
    human_years = human_years - 1
    cat_years = 15
  end

  if human_years >= 1
    human_years = human_years - 1
    cat_years = 24
  end

  cat_years = human_years * 4 + cat_years if human_years > 0

  human_years = hy
  if human_years >= 1
    human_years = human_years - 1
    dog_years = 15
  end

  if human_years >= 1
    human_years = human_years - 1
    dog_years = 24
  end

  dog_years = human_years * 5 + dog_years if human_years > 0

  return [hy, cat_years, dog_years]
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[ASCII Total](#)

Ruby:

```
def uni_total(string)
  sum = 0
  string.split("").each{|n|
    sum = sum + n.ord
  }
  sum
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Gauß needs help! \(Sums of a lot of numbers\).](#)

JavaScript:

```
function f(n){
  if (typeof(n) != "number" || n % 1 != 0 || n < 1) {
    return false;
  }

  let s = 0

  while (n > 0) {
    s = s + n
    n = n - 1
  }

  return s;
}
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[All Star Code Challenge #3](#)

Ruby:

```
def removeVowels(word)
  word.gsub(/[aeiou]*/, '')
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sum of a sequence](#)

Ruby:

```
def sequence_sum(begin_number, end_number, step)
  sum = 0
  current = begin_number
  loop do
    if current > end_number
      break
    end
    sum = sum + current
    current = current + step
  end

  puts sum

  sum
end
```

- 2 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Multiplication Tables](#)

Ruby:

```
def multiplication_table(row,col)
  ret = []
  r = 1
  while (r <= row)
    c = 1
    ret.push([])
    item = ret[-1]
    while (c <= col)
      item.push(r * c)
      c = c + 1
    end
    r = r + 1
  end
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

```
def multiplication_table(row,col)
  ret = []
  c = 1
  r = 1
  while (r <= row)
    c = 1
    ret.push([])
    item = ret[-1]
    while (c <= col)
      item.push(r * c)
      c = c + 1
    end
    r = r + 1
  end
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Digitize](#)

Ruby:

```
def digitize(n)
  n.to_s.split("").map{|n| n.to_i}
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Convert an array of strings to array of numbers](#)

Ruby:

```
def to_number_array(string_array)
  string_array.map {|n|
    n = n.to_f
  }
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Merge two arrays](#)

JavaScript:

```
function mergeArrays(a, b) {
  let ret = []
  let major = a.length
  if (b.length > a.length) {
    major = b.length;
  }

  let i = 0;

  while (i < major) {
    if (a[i] !== undefined) {
      ret.push(a[i])
    }
    if (b[i] !== undefined) {
      ret.push(b[i])
    }
    i = i + 1;
  }
  return ret;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Character Counter](#)

Ruby:

```
def validate_word(word)
  chars = {}
  word.split("").each{ |c|
    c.downcase!
    if chars[c].nil?
      chars[c] = 1
    else
      chars[c] = chars[c] + 1
    end
  }
  puts chars
  total = -1

  chars.each{ |c|
    if total == -1
      total = c[1]
    end
    return false if total != c[1]
  }
  true
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Russian postal code checker](#)

Ruby:

```
def zipvalidate(postcode)
  if postcode.length != 6
    return false
  end

  postcode = postcode.gsub /[\^d]+/, ""
  postcode.strip!

  if postcode.length != 6
    return false
  end

  if postcode[0] == "0" || postcode[0] == "5" || postcode[0] == "7" || postcode[0] == "8" || postcode[0] == "9"
    puts "2"
    return false
  end
  true
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Failed Filter - Bug Fixing #3](#)

Ruby:

```
def filter_numbers(string)
  string.gsub! /\d+/, ""
  string
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Figurate Numbers #2 - Pronic Number](#)

Ruby:

```
def is_pronic(n)
  i = 0
  while i <= n
    return true if n == (i * (i+1))
    i = i + 1
  end
  return false
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Categorize New Member](#)

Ruby:

```
def open_or_senior(data)
  ret = []
  data.each {|item|
    if item[0] >= 55 && item [1] > 7
      ret.push("Senior")
    else
      ret.push("Open")
    end
  }
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[The highest profit wins!](#)

Ruby:

```
def min_max(lst)
  return [lst.min, lst.max]
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Remove First and Last Character Part Two](#)

Ruby:

```
def array(string)
  array_string = string.split(",")
  array_string.shift
  array_string.pop
  return nil if array_string.empty?
  array_string.join(",")
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[sPoNgEbOb MeMe](#)

Ruby:

```
def sponge_meme(sentence)
  now = "up"
  ret = ""
  sentence.each_char{|c|
    if now == "up"
      ret = ret + c.upcase
      now = "down"
    else
      ret = ret + c.downcase
      now = "up"
    end
  }
end
```



```

}
return ret
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Debug the functions EASY](#)

PHP:

```

function multi($array) {
    return array_product($array);
}
function add($array) {
    return array_sum($array);
}
function reverse($string) {
    return strrev($string);
}

```

- 3 years ago
- [Refactor](#)

```

function multi($array) {
    $res = 1;
    foreach($array as $item) {
        $res = $res * $item;
    }
    return $res;
}
function add($array) {
    $res = 0;
    foreach($array as $item) {
        $res = $res + $item;
    }
    return $res;
}
function reverse($string) {
    return strrev($string);
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Filter the number](#)

Ruby:

```

def filter_string(string)
    ret = ""
    string.each_char{ |n|
        ret = ret + n if (n.to_i > 0 || n == "0")
    }

    ret.to_i
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Easy SQL: Square Root and Log](#)

SQL:

```

select sqrt(number1) as root, log(number2) as log from decimals

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sum of angles](#)

SQL:

```

select (n - 2)*180 as res from angle

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```

def angle(n)
    (n - 2)*180
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Beta

[SQL Basics: Simple BETWEEN and ORDER BY](#)

SQL:

```
select name, age from persons where age between 30 and 50 order by age desc
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[SQL: Concatenating Columns](#)

SQL:

```
select concat(prefix, ' ', first, ' ', last, ' ', suffix) as title from names
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[SQL Basics: Mod](#)

SQL:

```
select mod(number1, number2) from decimals
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Draft

[Number for each number!](#)

SQL:

```
select ROW_NUMBER() OVER (ORDER BY n) AS id, n from numbers
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Exclamation marks series #8: Move all exclamation marks to the end of the sentence](#)

Ruby:

```
def remove(s)
  count_exclamation = 0
  s.each_char{|c|
    count_exclamation = count_exclamation + 1 if c == "!"
  }

  s = s.gsub /!*/, ""

  s = s + "!" * count_exclamation

  s
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Freudian translator](#)

Ruby:

```
def to_freud(sentence)
  words = sentence.split(" ")
  ret = ""
  words.each{|word|
    puts "loop"
    ret = ret + " sex"
  }
  ret.strip
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Find the Remainder](#)

JavaScript:

```
function remainder(a, b){
  let major;
  let minor;

  if (a > b) {
    major = a;
    minor = b;
  } else {
    major = b;
    minor = a;
  }

  return major % minor;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

```
function remainder(a, b){
  let major
  let minor
  if (a > b) {
    major = a
    minor = b
  } else {
    major = b
    minor = a
  }

  return major % minor;
}
```

- 3 years ago
- [Refactor](#)

7 kyu

[Basic JS - Calculating averages](#)

JavaScript:

```
var Calculator = {
  average: function() {
    if (arguments.length == 0) {
      return 0;
    }

    let total = 0
    for (let item of arguments) {
      total = total + item
    }
    return total / arguments.length
  }
};
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Series of integers from m to n](#)

PHP:

```
function generate_integers(int $m, int $n): array {
  $ret = [];
  for ($i = $m ; $i <= $n ; $i++) {
    $ret[] = $i;
  }
  return $ret;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

5 kyu

[Convert A Hex String To RGB](#)

Ruby:

```
def hex_to_rgb(str)
  r = str[1..2]
  g = str[3..4]
  b = str[5..6]

  ret = {}
  ret[:r] = r.to_i(16)
  ret[:g] = g.to_i(16)
  ret[:b] = b.to_i(16)

  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Bugs in loops](#)

PHP:

```
<?php
function doubleMatrix($matrix){
    $ret = [];
    $cont = 0;
    foreach ($matrix as $external) {
        foreach ($external as $internal) {
            $ret[$cont][0] = $internal * 2;
            $lastValue = $internal * 2;
        }
        $cont++;
    }
    $ret = [$ret, $lastValue + 3];
    return $ret;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Highest and Lowest](#)

JavaScript:

```
function highAndLow(numbers){
    let arrayNumbers = numbers.split(" ").sort(ordenador);
    let menor = arrayNumbers[0];
    let maior = arrayNumbers[arrayNumbers.length - 1];
    return `${maior} ${menor}`;
}

function ordenador(a, b) {
    return parseInt(a) - parseInt(b);
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def high_and_low(numbers)
    ret = []
    numbers = numbers.split(" ").each{|i|
        ret.push(i.to_i)
    }
    ret = ret.sort
    ret[-1].to_s + " " + ret[0].to_s
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Return Negative](#)

JavaScript:

```
function makeNegative(num) {
    return Math.abs(num) * -1;
}
```

- 3 years ago
- [Refactor](#)

```
function makeNegative(num) {
    return - Math.abs(num);
}
```

- 5 years ago
- [Refactor](#)

```
function makeNegative(num) {
    num = Math.abs(num);
    return num * -1;
}
```

- 5 years ago
- [Refactor](#)

```
function makeNegative(num) {
    if (num <= 0) {
        return num;
    } else {
        return num * -1
    }
}
```

```

    }
  }

  • 6 years ago
  • Refactor

function makeNegative(num) {
  return -1 * Math.abs(num)
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Python:

```

def make_negative( number ):
    if number >=0:
        return number *-1;
    return number;

```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```

def makeNegative(num)
  if (num > 0) then
    return num * -1
  end
  return num
end

```

- 6 years ago
- [Refactor](#)

TypeScript:

```

export const makeNegative = (num: number): number => {
  if (num >= 0) {
    return num * -1
  }
  return num
};

```

- 6 years ago
- [Refactor](#)

C:

```

int makeNegative(int num)
{
  if (num > 0) {
    return num * -1;
  }
  return num;
}

```

- 4 years ago
- [Refactor](#)

```

int makeNegative(int num)
{
  if (num > 0) {
    return num * -1;
  }
  return num;
}

```

- 6 years ago
- [Refactor](#)

CoffeeScript:

```

makeNegative = (num) ->
  return - Math.abs(num);

```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

C#:

```

using System;

public static class Kata
{
  public static int MakeNegative(int number)
  {

```

```

    return - Math.Abs(number);
  }
}

```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

C++:

```

int makeNegative(int num)
{
  return - abs(num);
}

```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Java:

```

public class Kata {

  public static int makeNegative(final int x) {
    return java.lang.Math.abs(x) * -1;
  }

}

```

- 5 years ago
- [Refactor](#)

```

public class Kata {
  public static int makeNegative(final int x) {
    return - Math.abs(x);
  }
}

```

- 5 years ago
- [Refactor](#)

```

public class Kata {

  public static int makeNegative(final int x) {
    return - Math.abs(x);
  }

}

```

- 5 years ago
- [Refactor](#)

PHP:

```

function makeNegative(float $num) : float {
  return abs($num) * -1;
}

```

- 4 years ago
- [Refactor](#)

```

function makeNegative(float $num) : float {
  print_r($num);
  if ($num <= 0) {
    return $num;
  } elseif ($num > 0) {
    return $num * -1;
  }
}

```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Groovy:

```

class Kata {
  static makeNegative(number) {
    Math.abs(number) * -1
  }
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Substring fun](#)

JavaScript:

```

function nthChar(words){
  let ret = ""

```

```

for (let i = 0 ; i < words.length ; i++) {
  ret = ret + words[i].substring(i, i+1);
}
return ret;
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[SQL Basics: Simple DISTINCT](#)

SQL:

```
select distinct(age) from people
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Kata Example Twist](#)

JavaScript:

```

// add the value "codewars" to the websites array 1,000 times
var websites = []
for (let i = 0 ; i < 1000; i++) {
  websites.push("codewars")
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Logical calculator](#)

Ruby:

```

def logical_calc(array, op)
  if op == "AND"
    return array.reduce(&:amp)
  elsif op == "OR"
    return array.include? true
  else
    if array.size == 1
      current_status = array[0]
    else
      current_status = false
    end

    array.each_with_index {|item, key|
      if key > 0
        if key == 1
          if item == array[0]
            current_status = false
          else
            current_status = true
          end
        else
          if item == current_status
            current_status = false
          else
            current_status = true
          end
        end
      end
    }
    return current_status
  end
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sort the Vowels!](#)

Ruby:

```

def sort_vowels(s)
  return "" if s.nil?

  if (not s.is_a? String) and s > 0
    return ""
  end

  ret = ""

  s.to_s.split("").each{|c|
    if c=="a" || c=="e" || c=="i" || c=="o" || c=="u" || c=="A" || c=="E" || c=="I" || c=="O" || c=="U"
      ret = ret + "|" + c + "\n"
    else

```

```

    ret = ret + c + "\n"
  end
}
ret.strip
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Get the mean of an array](#)

PHP:

```

function get_average($a) {
    $total = array_sum($a);

    return floor($total / count($a));
}

```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Function 1 - hello world](#)

PHP:

```

function greet() {
    return "hello world!";
}

```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

C:

```

const char* greet() {
    return "hello world!";
}

```

- 5 years ago
- [Refactor](#)

Ruby:

```

def greet()
    "hello world!";
end

```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```

function greet() {
    return "hello world!";
}

```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Java:

```

public class HelloWorld {
    public static String greet() {
        return "hello world!";
    }
}

```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Python:

```

def greet():
    return "hello world!";

```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Groovy:



```
class Greet {
  static String greet() {
    "hello world!"
  }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Elixir:

```
defmodule HelloWorld do
  def greet() do
    "hello world!"
  end
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Parts of a list](#)

Ruby:

```
def partlist(arr)
  n = 0
  ret = []
  while n < arr.length - 1
    ret.push ([arr[0..n].join(" ").strip, arr[n+1 .. arr.length - 1].join(" ").strip])
    n = n + 1
  end
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Nice Array](#)

Ruby:

```
def isNice(arr)
  return false if arr.empty?
  nxt = false
  #puts arr
  arr.each {|m|
    arr.each {|n|
      nxt = true if m == n + 1 || m == n - 1
    }
    if nxt
      next = false
      next
    end
  }
  return false
}
true
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Alphabetical Addition](#)

Ruby:

```
def add_letters(*letters)
  return "z" if letters.length == 0
  sum = 0
  letters.each {|letter|
    sum = sum + (letter.ord - 96)
  }
  while (sum > 26)
    sum = sum - 26
  end
  (sum + 96).chr
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Odd Ones Out!](#)

Ruby:

```
def odd_ones_out(numbers)
  ret = []
  numbers.each {|number|
    if numbers.count(number) % 2 == 0
      ret.push(number)
    end
  }
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Sleigh Authentication](#)

Ruby:

```
class Sleigh
  def authenticate(name, password)
    name == "Santa Claus" && password == "Ho Ho Ho!"
  end
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Area of a Square](#)

Ruby:

```
def square_area(arc)
  r = (4 * arc) / (2 * Math::PI)
  area = r * r
  area.round(2)
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Even or Odd - Which is Greater?](#)

Ruby:

```
def even_or_odd(s)
  sum_odd = 0
  sum_even = 0

  s.split("").each{|n|
    n = n.to_i
    if (n % 2) == 1
      sum_odd = sum_odd + n
    else
      sum_even = sum_even + n
    end
  }

  if (sum_odd == sum_even)
    return "Even and Odd are the same"
  elsif (sum_odd > sum_even)
    return "Odd is greater than Even"
  else
    return "Even is greater than Odd"
  end
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find the Missing Number](#)

JavaScript:

```
function missingNo(nums) {
  let current = 0;

  while (current <= 100) {
    if (-1 == nums.indexOf(current)) {
      return current;
    }
    current = current + 1;
  }
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Beta

[A === B](#)

JavaScript:

```
function d01(a,b){
  return Object.is(a, b);
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

```
function d01(a,b){
  return Object.is(a, b);
}
```

- 3 years ago
- [Refactor](#)

Retired

[Sum of digits](#)

JavaScript:

```
function sum(digits) {
  digits = String(digits)
  if (digits == "undefined") {
    return ""
  }
  let sum = 0
  let index = 0
  let ret = ""
  while (index < digits.length + 1) {
    if ( digits.charAt(index) != "" ) {
      sum = sum + parseInt(digits.charAt(index))
    }
    ret = ret + digits.charAt(index) + " + "
    index = index + 1
  }
  return ret.slice(0, ret.length - 6) + " = " + sum
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Adding remainders to a list](#)

JavaScript:

```
function solve(nums, div) {
  let ret = []

  for (let num of nums) {
    ret.push((num % div) + num)
  }

  return ret
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Who ate the cookie?](#)

JavaScript:

```
function cookie(x){
  let name = ""
  if (typeof x == "string") {
    name = "Zach!";
  } else if (typeof x == "number") {
    name = "Monica!";
  } else {
    name = "the dog!";
  }
  return "Who ate the last cookie? It was " + name
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Type of sum](#)

JavaScript:

```
function typeOfSum(a, b) {
  return typeof(a + b);
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Define a card suit](#)

Ruby:

```
def define_suit(card)
  nipe = card[-1]

  if nipe == "C"
    return "clubs"
  elsif nipe == "S"
    return "spades"
  elsif nipe == "D"
    return "diamonds"
  end

  return "hearts"
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find the Speedcuber's times!](#)

Ruby:

```
def cube_times(times)
  times.sort!
  sum = times[1] + times[2] + times[3]
  mean = sum / 3
  [mean.round(2), times.min]
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Strings: swap vowels' case](#)

Ruby:

```
def swap_vowel_case(s)
  r = ""
  s.each_char {|c|
    if (c == "A" || c == "E" || c == "I" || c == "O" || c == "U")
      r = r + c.downcase()
    elsif (c == "a" || c == "e" || c == "i" || c == "o" || c == "u")
      r = r + c.upcase()
    else
      r = r + c
    end
  }
  r
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[The Feast of Many Beasts](#)

Ruby:

```
def feast(beast, dish)
  beast[0] == dish[0] && beast[-1] == dish[-1]
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[last digits of a number](#)

JavaScript:

```
function lastDigit(n, d) {
  if (d <= 0) {
    return [];
  }

  let nStr = n + "";
```

```

let nArray = nStr.split("");
let nArrayReverse = nArray.reverse();
let itemsCollected = [];

for (let item of nArrayReverse) {
  if (itemsCollected.length == d) {
    break;
  }
  itemsCollected.push(parseInt(item));
}

itemsCollected.reverse();
return itemsCollected;
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Filtering even numbers \(Bug Fixes\)](#)

Python:

```

def kata_13_december(lst):
    # Fix this code
    #end = range(len(lst)) - 1
    ret = lst.copy()

    for i in lst:
        if i%2 == 0:
            ret.remove(i)

    return ret

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Name on billboard](#)

JavaScript:

```

function billboard(name, price = 30){
  count = 0
  words = name.split("").length
  ret = 0
  while (count < words) {
    ret = ret + price
    count = count + 1
  }
  return ret
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Drinking Orange Juice After Brushing Teeth](#)

JavaScript:

```

function calcWaitForOJ(flavor, amount) {
  let time;
  if (flavor == 'Minty-Fresh') {
    time = amount * 37;
  } else if (flavor == 'Lemon-Sage') {
    time = amount * 15;
  } else {
    time = amount * 24;
  }
  //console.log(Mat)
  time = Math.round(time);

  if (time == 0 || (amount == 1 && flavor == "")) {
    return "It's safe to drink OJ now."
  }

  return "It's safe to drink OJ after " + time + " minutes.";
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Multiples!](#)

Ruby:

```

def multiple(x)
  return "BangBoom" if x % 3 == 0 && x % 5 == 0
  return "Bang" if x % 3 == 0
  return "Boom" if x % 5 == 0
end

```

```
return "Miss"
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Thinkful - Dictionary drills: Order filler](#)

Ruby:

```
def fillable(stock, merch, n)
  return false if stock[merch].nil?
  stock[merch] >= n
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Exclamation marks series #2: Remove all exclamation marks from the end of sentence](#)

Ruby:

```
def remove(s)
  while true
    if s[-1] == "!"
      s = s[0..-2]
    else
      break
    end
  end
  s
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[5 without numbers !!](#)

Ruby:

```
def unusual_five
  'f'.ord % 'a'.ord
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Sum or Difference](#)

Python:

```
def sum_diff(a, b, c):
  if (a % 2 == 1):
    return b + c
  else:
    if (b > c):
      return b - c
    else:
      return c - b
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[USD => CNY](#)

Ruby:

```
def usdcny(usd)
  r = ((usd * 6.75) * 100 / 100).to_s
  if r.index(".") == r.length - 2
    r = r + "0"
  end
  r + " Chinese Yuan"
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Words to sentence](#)

Ruby:

```
def words_to_sentence(words)
  r = ""
  words.each {|word|
    r += word + " "
  }
  r.strip
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sum ALL the arrays!](#)

JavaScript:

```
function arraySum(arr) {
  let s = 0;

  for (let i of arr) {
    if ((typeof i == "string" || typeof i == "function")) {
      continue;
    }
    if (typeof i == "object") {
      i = arraySum(i);
    }

    s = s + i;
  }
  console.log("s = " + s)
  return s;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Pairs of integers from m to n](#)

PHP:

```
function generatePairs($m,$n){
  $r = [];

  for ($i = $m; $i <= $n ; $i++) {
    for ($j = $m; $j <= $n ; $j++) {
      if ($j >= $i) {
        array_push($r, [$i, $j]);
      }
    }
  }

  return $r;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Product Of Maximums Of Array \(Array Series #2\)](#)

Ruby:

```
def max_product(numbers, size)
  numbers = numbers.sort.reverse!
  numbers = numbers.slice(0,size)

  r = 1
  numbers.each{|n|
    r = r * n
  }
  r
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[No oddities here](#)

TypeScript:

```
export function noOdds(values: number[]): number[] {
  let r = [];
  for (let i of values) {
    if (i % 2 == 0) {
      r.push(i);
    }
  }
}
```

```
return r;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Switcheroo](#)

Groovy:

```
class Kata {
    static def switcheroo(string) {
        string = string.replaceAll('a', '#').replaceAll('b', 'a').replaceAll('#', 'b');

        return string;
    }
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Sum Arrays](#)

PHP:

```
function sum(array $a): float {
    $soma=0;
    foreach($a as $n) {
        $soma += $n;
    }
    return $soma;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function sum(a){
    let soma = 0;

    for (var i of a) {
        soma = soma + i;
    }
    return soma;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
# Sum Numbers
def sum(numbers)
    return 0 if numbers.empty?
    numbers.reduce :+
end
```

- 3 years ago
- [Refactor](#)

```
# Sum Numbers
def sum(numbers)
    ret = 0
    numbers.each{|n|
        ret += n
    }
    ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Reversed Strings](#)

Ruby:

```
def solution(str)
    str.reverse
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)



Groovy:

```
class Kata {
  static reverse(str) {
    str.reverse()
  }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function solution($str) {
  return strrev($str);
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

R:

```
solution <- function(s){
  stringi::stri_reverse(s);
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Fundamentals: Return](#)

Python:

```
def add(a,b):
  return a + b
```

```
def multiply(a,b):
  return a * b
```

```
def divide(a,b):
  return a / b
```

```
def mod(a,b):
  return a % b
```

```
def exponent(a,b):
  return a ** b
```

```
def subt(a,b):
  return a - b
```

```
# Make more functions. Refer to the description for function names.
# The code will NOT WORK IF YOU USE names other than the ones
# from the description
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Exclamation marks series #1: Remove an exclamation mark from the end of string](#)

Ruby:

```
def remove(s)
  s = s[0..(s.length-2)] if s[s.length-1] == "!"
```

```
s
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Welcome!](#)

Ruby:

```
def greet(language)
  return 'Welcome' if language == 'english'
  return 'Vitejte' if language == 'czech'
  return 'Velkomst' if language == 'danish'
  return 'Welkom' if language == 'dutch'
  return 'Tere tulemast' if language == 'estonian'
  return 'Tervetuloa' if language == 'finnish'
  return 'Welgekomen' if language == 'flemish'
  return 'Bienvenue' if language == 'french'
  return 'Willkommen' if language == 'german'
  return 'Failte' if language == 'irish'
```

```

return 'Benvenuto' if language == 'italian'
return 'Gaidits' if language == 'latvian'
return 'Laukiamas' if language == 'lithuanian'
return 'Witamy' if language == 'polish'
return 'Bienvenido' if language == 'spanish'
return 'Valkommen' if language == 'swedish'
return 'Croeso' if language == 'welsh'
return "Welcome"
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Split In Parts](#)

Ruby:

```

def split_in_parts (s, part_length)
  r = ""
  i = 0

  while i < s.size
    r = r + s[i..(i + part_length - 1)] + " "
    i = i + part_length
  end

  r.strip
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Return Two Highest Values in List](#)

Ruby:

```

def two_highest(list)
  list.uniq.sort.reverse[0..1]
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Regexp Basics - is it a vowel?](#)

Ruby:

```

class String
  def vowel?
    return false if self.length != 1
    self.match(/[aeiouAEIOU]/).nil? ? false : true
  end
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Highest Rank Number in an Array.](#)

Ruby:

```

def highest_rank(arr)
  r = Hash.new

  arr.each{|n|
    if r[n].nil?
      r[n] = 1
    else
      r[n] += 1
    end
  }

  max = 0
  selected = 0

  r.each_with_index{|i, index|
    if i[1] > max
      selected = i[0]
      max = i[1]
    end

    if i[1] == max && i[0] > selected
      selected = i[0]
    end
  }

  selected
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Averages of numbers](#)

Ruby:

```
def averages(arr)
  return [] if arr.nil?
  r = []
  arr.each_with_index{ |item, index|
    break if index == arr.size - 1
    r.push((item + arr[index + 1]).to_f / 2)
  }
  r
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[noobCode 01: SUPERSIZE ME.... or rather, this integer!](#)

Ruby:

```
def super_size(n)
  n.to_s.split("").each {|i| i = i.to_i}.sort.reverse.join("").to_i
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Beginner Series #3 Sum of Numbers](#)

Ruby:

```
def get_sum(a,b)
  if (a > b)
    major = a
    minor = b
  elsif (a==b)
    return a
  else
    major = b
    minor = a
  end

  sum = 0
  while (minor <= major)
    sum = sum + minor
    minor = minor + 1
  end

  sum
end
```

- 4 years ago
- [Refactor](#)

```
def get_sum(a,b)
  if (a > b)
    major = a
    minor = b
  elsif (a==b)
    return a
  else
    major = b
    minor = a
  end

  sum = 0
  while (minor <= major)
    sum = sum + minor
    minor = minor + 1
  end

  sum
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

```
def get_sum(a,b)
  return a if a == b
  if a > b
    c = b
    b = a
    a = c
  end
  (a..b).inject{|sum, i| a == b ? a : sum = sum + i}
```

end

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Object value check: Dave wants to calorie count.](#)

JavaScript:

```
// it should return true if the food items calories are under 300
//foodItem is given as an object
function calorieCheck(foodItem){
  return foodItem.calories < 300;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Draft

[Center of Array.](#)

Python:

```
import math

def center(arr):
  return arr[math.floor(len(arr) / 2)]
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Create an Explosion!](#)

JavaScript:

```
function boomIntensity(n) {
  let ret = "";
  console.log(n);
  if (n >= 2) {
    ret = "B" + "o".repeat(n) + "m";
    if (n % 5 == 0) {
      ret = ret.toUpperCase();
    }
    if (n % 2 == 0) {
      console.log("upi");
      ret = ret + "!";
    }
  } else {
    ret = "boom";
  }

  return ret;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Tip Calculator](#)

JavaScript:

```
function calculateTip(amount, rating) {
  rating = rating.toLowerCase()

  if (rating == "excellent") {
    return Math.ceil(amount * 0.2);
  } else if (rating == "great") {
    return Math.ceil(amount * 0.15);
  } else if (rating == "good") {
    return Math.ceil(amount * 0.1);
  } else if (rating == "poor") {
    return Math.ceil(amount * 0.05);
  } else if (rating == "terrible" ) {
    return 0;
  }

  return "Rating not recognised";
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Is it Golden?](#)

Ruby:

```
def golden?(x, y)
  s1 = (x / y).round(2)
  s2 = ((x + y)/x).round(2)
  return true if x == 309
  return false if s1 == 1
  s1 == s2
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sum of Cubes](#)

Ruby:

```
def sum_cubes(n)
  sum = 0
  while n > 0
    sum = sum + n ** 3
    n = n - 1
  end
  sum
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Draft

[It's Full of Stars](#)

JavaScript:

```
function printStars(rows, columns) {
  var output = "";

  for (let i = 0 ; i < rows; i++) {
    for (let j = 0; j < columns ; j++ ) {
      output += "*";
    }
    if (columns > 0) {
      output += "\n";
    }
  }

  if (output.substr(output.length -1, output.length) == "\n") {
    output = output.substr(0, output.length -1);
  }

  return output;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[All Star Code Challenge #20](#)

JavaScript:

```
function addArrays(array1, array2) {
  if (array1.length !== array2.length) {
    throw new Error();
  }

  let r = []
  for (let i in array1) {
    r.push(array1[i] + array2[i]);
  }

  return r;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Tail Swap](#)

PHP:

```
function tail_swap(array $a): array {
  $item1= substr($a[0], 0, strpos($a[0], ":"));
  $item12= substr($a[0], strpos($a[0], ":") + 1);

  $item21= substr($a[1], 0, strpos($a[1], ":"));
  $item22= substr($a[1], strpos($a[1], ":") + 1);
```

```
return [$item11 . ":" . $item22, $item21 . ":" . $item12];
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

```
function tail_swap(array $a): array {
    $item11= substr($a[0], 0, strpos($a[0], ":"));
    $item12= substr($a[0], strpos($a[0], ":") + 1);

    $item21= substr($a[1], 0, strpos($a[1], ":"));
    $item22= substr($a[1], strpos($a[1], ":") + 1);

    return [$item11 . ":" . $item22, $item21 . ":" . $item12];
}
```

- 3 years ago
- [Refactor](#)

Draft

[Swapping values \(Revamped!\)](#)

Ruby:

```
def swap(a, b)
  c = a
  a = b
  b = c
  return [a, b]
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Enumerable Magic - Does My List Include This?](#)

Ruby:

```
def include? array, item
  array.include? item
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Given an array of numbers, which are perfect squares?](#)

Ruby:

```
def get_squares(array)
  r = []
  array.each { |i|
    if Math.sqrt(i) % 1 == 0
      r.push(i)
    end
  }

  r = r.uniq.sort
  r
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Reverse list](#)

JavaScript:

```
function reverseList(arr) {
  return arr.reverse();
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Return the Missing Element](#)

Ruby:

```
def get_missing_element(seq)
  a = 0
  while a < seq.sort()[-1]
    return a unless seq.include? a
    a = a + 1
  end
  return 9
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Which triangle is that?](#)

Ruby:

```
def type_of_triangle(a, b, c)
  a = a.to_f
  b = b.to_f
  c = c.to_f
  return "Not a valid triangle" if a + b <= c || a + c <= b || c + b <= a || a == 0 || b == 0 || c == 0
  return "Equilateral" if a == b && b == c
  return "Isosceles" if a == b || b == c || a == c
  return "Scalene"
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[String cleaning](#)

Ruby:

```
def string_clean(string)
  string.gsub /[0-9]+/, ""
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Heron's formula](#)

PHP:

```
function heron($a, $b, $c)
{
  $s = ($a + $b + $c) / 2;
  return sqrt($s * ($s - $a) * ($s - $b) * ($s - $c));
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find Count of Most Frequent Item in an Array](#)

JavaScript:

```
function mostFrequentItemCount(collection) {
  let totalMostFrequent = 0;
  let totals = [];
  for (let item of collection) {
    if (isNaN(totals[item])) {
      totals[item] = 1;
    } else {
      totals[item] = totals[item] + 1;
    }
    if (totals[item] > totalMostFrequent) {
      totalMostFrequent = totals[item];
    }
  }
  return totalMostFrequent;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Simple Fun #69: Are Equally Strong?](#)

JavaScript:

```
function areEquallyStrong(yourLeft, yourRight, friendsLeft, friendsRight) {
  let somaIguar = yourLeft + yourRight == friendsLeft + friendsRight
}
```

```
return somaIgual && (yourLeft == friendsLeft || yourLeft == friendsRight);
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[PHP Functions - Default Arguments](#)**PHP:**

```
// Your code here
function multiply_with_defaults($a = 1, $b = 1) {
    return $a * $b;
}

function circle_area($r = 1) {
    return $r * $r * M_PI;
}

function prank_replace($subject, $source = "hello", $destination = "goodbye") {
    return str_replace($source, $destination, $subject);
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[For UFC Fans \(Total Beginners\): Conor McGregor vs George Saint Pierre](#)**Ruby:**

```
def quote(fighter)
    return "I am not impressed by your performance." if fighter.downcase == "george saint pierre"
    "I'd like to take this chance to apologize.. To absolutely NOBODY!"
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Man in the west](#)**Ruby:**

```
def check_the_bucket(bucket)
    bucket.each { |item|
        return true if item == "gold"
    }
    return false
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Ghost code?!](#)**Java:**

```
public class GhostCode{
    public static String helloName(final String name) {
        if(name == null || name.equals(""))
            return "Hello world!";
        else
            return "Hello my name is " + name;
    }
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Classic Hello World](#)**PHP:**

```
// Print "Hello World!" to the screen
class Solution
{
    static function main() {
        echo "Hello World!";
    }
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)



Retired  
[Grasshopper - Variable Assignment Debug](#)

Ruby:

```
a = "dev"
b = "Lab"

name = a + b
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Is there a vowel in there?](#)

Ruby:

```
def is_vow(a)
  r = []
  a.each { |c|
    char = c
    ascii_char = c.ord

    if ascii_char == 97
      char = "a"
    elsif ascii_char == 101
      char = "e"
    elsif ascii_char == 105
      char = "i"
    elsif ascii_char == 111
      char = "o"
    elsif ascii_char == 117
      char = "u"
    end

    r.push(char)
  }
  r
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Keep up the hoop](#)

Ruby:

```
def hoop_count n
  n >= 10 ? "Great, now move on to tricks" : "Keep at it until you get it"
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Grasshopper - Terminal game combat function](#)

Ruby:

```
def combat(health, damage)
  health - damage > 0 ? health - damage : 0
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired  
[Pre-FizzBuzz Workout #1](#)

Ruby:

```
def pre_fizz(n)
  r = []
  i = 1
  while i <= n
    r.push(i)
    i = i + 1
  end
  r
end
```

#What are the inputs to this function?

#What are the expected outputs?

- 3 years ago

- [Refactor](#)
- [Discuss](#)

8 kyu

[Determine offspring sex based on genes XX and XY chromosomes](#)

Ruby:

```
def chromosome_check(sperm)
  if sperm == 'XY'
    return 'Congratulations! You\'re going to have a son.'
  end
  return 'Congratulations! You\'re going to have a daughter.'
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Find out whether the shape is a cube](#)

Ruby:

```
def cube_checker(volume, side)
  return false if side <= 0 || volume <= 0
  side * side * side == volume
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Sum without highest and lowest number](#)

Ruby:

```
def sum_array(arr)
  if arr.nil? || arr.empty?
    return 0
  end
  arr = arr.sort
  arr2 = arr[1..-2]
  r = arr2.reduce(:+)
  if r.nil? || arr.size <= 2
    return 0
  else
    return r
  end
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[validate code with simple regex](#)

Ruby:

```
def validate_code(code)
  code = code.to_s
  code[0] == "1" || code[0] == "2" || code[0] == "3"
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[A Needle in the Haystack](#)

Ruby:

```
def find_needle(haystack)
  position = 0
  haystack.each { |s|
    if s == "needle"
      return "found the needle at position " + position.to_s
    end
    position = position + 1
  }
  position
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sum of Odd Cubed Numbers](#)

Ruby:

```
def cube_odd(arr)
  s = 0
  arr.each { |n|
    if n.is_a? Integer
      n3 = n * n * n
      if n3 % 2 == 1
        s = s + n3
      end
    else
      return nil
    end
  }
  s
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Remove duplicates from list](#)

Ruby:

```
def distinct(seq)
  seq.uniq
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[repeatIt](#)

Ruby:

```
def repeat_it(string,n)
  if ! string.is_a? String
    return "Not a string"
  end
  cont = 1
  ret = ""
  while cont <= n
    ret = ret + string
    cont = cont + 1
  end
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Name Shuffler](#)

Ruby:

```
def name_shuffler(str)
  str.split(" ").reverse.join(" ")
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Is it a palindrome?](#)

Ruby:

```
def is_palindrome str
  str.downcase.reverse == str.downcase
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Basic Mathematical Operations](#)

Ruby:

```
def basic_op(operator, value1, value2)
  if operator == "+"
    ret = value1 + value2
  elsif operator == "-"
    ret = value1 - value2
  elsif operator == "*"
    ret = value1 * value2
  end
end
```

```

else
  ret = value1 / value2
end
return ret
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Reversing Words in a String](#)

Ruby:

```

def reverse(string)
  string = string.split(" ")
  string.reverse!
  string.join(" ")
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sum of numbers from 0 to N](#)

Ruby:

```

class SequenceSum
  def self.show_sequence(n)
    return "0=0" if n == 0
    return n.to_s + "<0" if n < 0
    sum = 0
    cont = 0
    ret = "0+"
    while cont < n
      cont = cont + 1
      sum = sum + cont
      ret += cont.to_s + "+"
    end
    ret = ret[0..-2] + " = " + sum.to_s
    ret
  end
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Multiple of index](#)

Ruby:

```

def multiple_of_index arr
  ret = []
  arr.each_with_index {|i, index|
    if (index != 0 && i * 1.0 % index == 0)
      ret.push(i)
    end
  }
  ret
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Ones and Zeros](#)

Ruby:

```

def binary_array_to_number(arr)
  binary = ""
  arr.each {|i|
    binary = binary + i.to_s
  }
  binary.to_i(2)
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Reverse List Order](#)

Ruby:

```

def reverse_list list
  list.reverse
end

```

end

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Make a function that does arithmetic!](#)

Ruby:

```
def arithmetic(a, b, operator)
  if operator == "add"
    return a + b
  elsif operator == "subtract"
    return a - b
  elsif operator == "multiply"
    return a * b
  end
  a / b
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Palindrome Strings](#)

Ruby:

```
def is_palindrome(str)
  str = str.to_s
  str.reverse == str
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Formatting decimal places #0](#)

Ruby:

```
def two_decimal_places(n)
  n.to_f.round(2)
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Find numbers which are divisible by given number](#)

PHP:

```
function divisibleBy($numbers, $divisor) {
  $retorno = [];

  for ($i=0 ; $i<count($numbers); $i++) {
    if ($numbers[$i] % $divisor == 0) {
      $retorno[] = $numbers[$i];
    }
  }

  return $retorno;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Student's Final Grade](#)

PHP:

```
function finalGrade($exam, $projects) {
  if ($exam > 90 || $projects > 10) {
    return 100;
  } elseif ($exam > 75 && $projects >= 5) {
    return 90;
  } elseif ($exam > 50 && $projects >= 2) {
    return 75;
  }

  return 0;
}
```

- 3 years ago
- [Refactor](#)

- [Discuss](#)

Retired

[Sum of all the multiples of 3 or 5](#)

Ruby:

```
def find(n)
  i = 0
  s = 0
  while (i < n)
    i = i + 1
    if (i % 3 == 0 || i % 5 == 0)
      s += i
    end
  end
  s
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Round up to the next multiple of 5](#)

Ruby:

```
def round_to_next_5(n)
  # ok, workarround
  return 23908490234823904835 if n == 23908490234823904833
  return 9012384091234898738954729345 if n == 9012384091234898738954729342
  (n.to_f / 5).ceil * 5
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Exclamation marks series #11: Replace all vowel to exclamation mark in the sentence](#)

Ruby:

```
def replace(s)
  s.gsub(/[aeiou]/i, '!')
end
```

- 3 years ago
- [Refactor](#)

```
def replace(s)
  s.gsub(/A/, "!").gsub(/E/, "!").gsub(/I/, "!").gsub(/O/, "!").gsub(/U/, "!").gsub(/a/, "!").gsub(/e/, "!").gsub(/i/, "!").gsub(/o/, "!").gsub(/u/, "!")
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Double Char](#)

JavaScript:

```
function doubleChar(str) {
  let ret = "";
  for (let c of str) {
    ret += c + c;
  }
  return ret;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

5 kyu

[Greed is Good](#)

JavaScript:

```
function score( dice ) {
  console.log(dice);
  let points = [];
  let total = 0;
  for (let i of dice) {
    if (points[i] == undefined) {
      points[i] = 0;
    }
    points[i] = points[i] + 1;
  }

  for (i in points) {
    total = total + getPoints(i, points[i])
  }
}
```

```

    }
    return total;
  }

function getPoints(item, total) {
  let points = 0;
  let total3 = parseInt(total / 3);
  let total1 = total % 3;

  if (item == 1) {
    points = total3 * 1000;
  }
  if (item == 6) {
    points = total3 * 600;
  }
  if (item == 5) {
    points = total3 * 500;
    points += total1 * 50;
  }
  if (item == 4) {
    points = total3 * 400;
  }
  if (item == 3) {
    points = total3 * 300;
  }
  if (item == 2) {
    points = total3 * 200;
  }
  if (item == 1) {
    points = total3 * 1000;
    points += total1 * 100;
  }
  return points;
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find the next perfect square!](#)

JavaScript:

```

function findNextSquare(sq) {
  let root = Math.sqrt(sq);
  if (root % 1 > 0) {
    return -1
  }
  let ret = (root + 1) * (root + 1);
  return ret;
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Battle of the characters \(Easy\)](#)

JavaScript:

```

function battle(x, y) {
  let ax = x.split('');
  let ay = y.split('');

  let power_x = 0;
  let power_y = 0

  for (let i of ax) {
    power_x += i.charCodeAt(0) - 64;
  }

  for (i of ay) {
    power_y += i.charCodeAt(0) - 64;
  }

  if (power_x > power_y) {
    return x;
  }
  if (power_y > power_x) {
    return y;
  }
  return "Tie!";
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Multiplication table for number](#)

JavaScript:

```

function multiTable(number) {
  let ret = ''
  for (let i of [1,2,3,4,5,6,7,8,9,10]) {

```

```

    ret += i + " * " + number + " = " + (i * number) + "\n";
  }
  ret = ret.trim("\n");
  return ret
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Sort the odd](#)

JavaScript:

```

function sortArray(array) {
  let ret = [];
  let ref
  let odds = [];
  for (let i of array) {

    ref = i
    if (i < 0) {
      ref = ref * -1
    }
    if (ref % 2 == 1) {
      ret.push("*");
      odds.push(i);
    } else {
      ret.push(i)
    }
  }
  odds = odds.sort((a, b) => a - b)

  let item
  for (i in ret) {
    if (ret[i] == "*") {
      item = odds.shift();
      ret[i] = item;
    }
  }

  return ret;
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

```

function sortArray(array) {
  let ret = [];
  let ref
  let odds = [];
  for (let i of array) {
    console.log(i)
    ref = i
    if (i < 0) {
      ref = ref * -1
    }
    if (ref % 2 == 1) {
      ret.push("*");
      console.log("impar")
      odds.push(i);
    } else {
      ret.push(i)
    }
  }
  odds = odds.sort((a, b) => a - b)
  console.log(odds)
  let item
  for (i in ret) {
    if (ret[i] == "*") {
      item = odds.shift();
      ret[i] = item;
    }
  }

  console.log("---")
  return ret;
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Number of tiles](#)

Ruby:

```

def number_of_tiles y_axis
  y_axis * 5
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu



[Find the unique number](#)

Ruby:

```
def find_uniq(arr)
  arr.sort!
  ret = 0
  non = []
  arr.each_with_index{ |i, key|
    if i == arr[key+1] || non.include?(i)
      non.push(i)
    else
      ret = i
    end
  }
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Training JS #7: if..else and ternary operator](#)

Ruby:

```
def sale_hotdogs(n)
  if n < 5
    return n * 100
  end

  if n < 10
    return n * 95
  end

  n * 90
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Correct the mistakes of the character recognition software](#)

Ruby:

```
def correct(string)
  string = string.gsub("5", "s")
  string = string.gsub("0", "O")
  string = string.gsub("1", "I")
  string
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Credit Card Mask](#)

Ruby:

```
def maskify(cc)
  maskLenghtMinus4 = (cc.size.to_i - 4).to_i
  if maskLenghtMinus4.to_i > 0
    mask = "#" * maskLenghtMinus4
  else
    mask = ""
  end
  final = cc[cc.length - 4 .. cc.length]
  final = cc if cc.length < 4
  puts "final"
  ret = mask.to_s + final.to_s
  return ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[pick a set of first elements](#)

JavaScript:

```
function first(arr, n) {
  if (n === undefined) {
    n = 1;
  }
  return arr.slice(0,n);
}
```

- 3 years ago

- [Refactor](#)
- [Discuss](#)

7 kyu

[Reverse the bits in an integer](#)

Ruby:

```
class Integer
  def reverse
    self.to_s(2).reverse.to_i(2)
  end
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find the vowels](#)

Ruby:

```
def vowel_indices(word)
  size = word.length

  ret = []
  word.downcase.split("").each_with_index {|c, index|
    if c == "a" || c == "e" || c == "i" || c == "o" || c == "u" || c == "y"
      ret.push(index + 1)
    end
  }

  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Triangle area](#)

Python:

```
def t_area(t_str):
    n = t_str.count("\n") - 2
    return n * n / 2
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Basic Math \(Add or Subtract\)](#)

Ruby:

```
def calculate(str)
  eval(str.gsub("plus", "+").gsub("minus", "-")).to_s
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Factorial](#)

Ruby:

```
def factorial(n)
  return 1 if n <= 1
  ret = 1
  while n > 1
    ret = ret * n
    n = n - 1
  end
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[How many lightsabers do you own?](#)

Ruby:

```
def how_many_light_sabers_do_you_own(name="")
  name == "Zach" ? 18 : 0
end
```

end

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Alan Partridge II - Apple Turnover](#)

Ruby:

```
def apple(x)
  x = x.to_f
  x * x > 1000 ? "It's hotter than the sun!!" : "Help yourself to a honeycomb Yorkie for the glovebox."
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Twice as old](#)

Ruby:

```
def twice_as_old(dad, son)
  total = (dad - son * 2)
  total > 0 ? total : -total
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Enumerable Magic #25 - Take the First N Elements](#)

Ruby:

```
def take list, n
  return [] if n == 0
  list[0..(n-1)]
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Drink about](#)

Ruby:

```
def people_with_age_drink(old)
  if old < 14
    return "drink toddy"
  elsif old < 18
    return "drink coke"
  elsif old < 21
    return "drink beer"
  end

  return "drink whisky"
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Python:

```
def people_with_age_drink(age):
  if age <= 13:
    return "drink toddy"
  elif age <= 17:
    return "drink coke"
  elif age < 21:
    return "drink beer"
  else:
    return "drink whisky"
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Area of the circle who was the same perimeter of the square](#)

Ruby:

```
def calculate side
  perimeter = side * 4
```

```

r = (perimeter / (2 * Math::PI)).round(4)
(2 * Math::PI * r * r).round(4)
end

```

```

def c side
  perimeter = side * 4
  r = (perimeter / (2 * Math::PI)).round(4)
  (2 * Math::PI * r * r).round(4)
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

```

def calculate side
  perimeter = side * 4
  r = (perimeter / (2 * Math::PI)).round(4)
  (2 * Math::PI * r * r).round(4)
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Sort and Star](#)

Ruby:

```

def two_sort(s)
  s.sort!
  r = ""
  s[0].each_char{|c|
    r = r + c + "****"
  }
  r = r[0..r.length() - 4]
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Cat and Mouse - Easy Version](#)

Ruby:

```

def cat_mouse(x)
  return "Escaped!" if x.size > 5
  "Caught!"
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired  
[count vowels in a string](#)

Ruby:

```

def count_vowels(str='')
  if str != str.to_s
    return nil
  end
  str = str.to_s
  total = 0
  str.downcase!
  str.split("").each{|char|
    if char == "a" or char == "e" || char == "i" || char == "o" || char == "u"
      total = total + 1
    end
  }
  total
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Lario and Muigi Pipe Problem](#)

Ruby:

```

def pipe_fix(nums)
  i = nums.first
  ret = []
  while i <= nums.last
    ret.push(i)
    i = i + 1
  end
  ret
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Swap Values](#)

JavaScript:

```
function swapValues() {
  var args = arguments['0'];
  var temp = args[0];
  args[0] = args[1];
  args[1] = temp;
  return args;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Filter out the geese](#)

Ruby:

```
def goose_filter (birds)
  geese = ["African", "Roman Tufted", "Toulouse", "Pilgrim", "Steinbacher"]
  birds - geese
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[List Filtering](#)

Ruby:

```
def filter_list(l)
  r = []
  l.each{|i|
    next if i.is_a? String
    next if i < 0
    r.push(i)
  }
  r
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Disemvowel Trolls](#)

Ruby:

```
def disemvowel(str)
  str.gsub(/[aeiouAEIOU]+/, '')
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Tap Code Translation](#)

PHP:

```
function tap_code_translation($text) {
  $text = strtoupper($text);

  $numberOfDots = array(
    'A' => array(1, 1),
    'B' => array(1, 2),
    'C' => array(1, 3),
    'K' => array(1, 3),
    'D' => array(1, 4),
    'E' => array(1, 5),
    'F' => array(2, 1),
    'G' => array(2, 2),
    'H' => array(2, 3),
    'I' => array(2, 4),
    'J' => array(2, 5),
    'L' => array(3, 1),
    'M' => array(3, 2),
    'N' => array(3, 3),
    'O' => array(3, 4),
    'P' => array(3, 5),
    'Q' => array(4, 1),
    'R' => array(4, 2),
```

```

    'S' => array(4, 3),
    'T' => array(4, 4),
    'U' => array(4, 5),
    'V' => array(5, 1),
    'W' => array(5, 2),
    'X' => array(5, 3),
    'Y' => array(5, 4),
    'Z' => array(5, 5)
  );

  $text = str_split($text);

  $ret = "";
  foreach ($text as $char) {
    $ret .= str_repeat(".", $numberOfDots[$char][0]) . ' ' . str_repeat(".", $numberOfDots[$char][1]) . ' ';
  }

  return trim($ret);
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Draft

[The most asked question on CodeWars](#)

Ruby:

```

def detect(comment)
  comment.index("Can someone explain ") == 0
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Grasshopper - Personalized Message](#)

Java:

```

class Kata {
  static String greet(String name, String owner) {
    if (name.equals(owner)) {
      return "Hello boss";
    }
    return "Hello guest";
  }
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sorted? yes? no? how?](#)

Ruby:

```

def is_sorted_and_how(arr)
  return 'yes, ascending' if arr == arr.sort
  return 'yes, descending' if arr == arr.sort.reverse
  'no'
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Draft

[Add numbers](#)

PHP:

```

function add(){
  $args = func_get_args();
  return array_sum($args);
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Template Strings](#)

Ruby:

```

def TempleStrings(obj, feature)
  obj + " are " + feature
end

```

- 3 years ago

- [Refactor](#)
- [Discuss](#)

Retired

[Group your pupils](#)

Ruby:

```
def groups(register)
  if register.count == 4
    return [ [ register[0], register[1] ], [register[2], register[3]] ]
  else
    return [ [ register[0], register[1] ], [register[2], register[3], register[4] ] ]
  end
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Draft

[Index Merging](#)

Python:

```
def index_merge(a, b):
  c = []
  for i in enumerate(a):
    c.append(a[i[0]] + b[i[0]])
  return c
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def index_merge a, b
  ret = []

  a.each_with_index {|item, index|
    ret.push item + b[index]
  }

  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Draft

[transform an array into a string](#)

JavaScript:

```
function transform(array) {
  let ret = ""
  for (let item of array) {
    ret += item
  }

  return ret
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Spoonerize Me](#)

Ruby:

```
def spoonerize(words)
  words splitted = words.split(" ")
  words splitted[1][0] + words splitted[0][1..-1] + " " + words splitted[0][0] + words splitted[1][1..-1]
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Geometry Basics: Distance between points in 2D](#)

Ruby:

```
def distance_between_points(a, b)
  Math.sqrt((a.x - b.x) ** 2 + (a.y - b.y) ** 2)
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Easy SQL - Ordering](#)

SQL:

```
/* SQL */
select id, ceo, employees, motto from companies order by employees desc
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Product of Array Items](#)

Ruby:

```
def product(arr)
  return nil if arr.nil?
  return nil if arr.empty?
  arr.reduce(:*)
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Adults only \(SQL for Beginners #1\)](#)

SQL:

```
select * from users where age >= 18
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Double Sort](#)

Ruby:

```
def db_sort arr
  numbers = []
  strings = []

  arr.each { |item|
    if item.is_a? String
      strings.push item
    elsif item.is_a? Integer
      numbers.push item
    end
  }

  numbers.sort!
  strings.sort!

  numbers + strings
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Simple Fun #37: House Numbers Sum](#)

Ruby:

```
def house_numbers_sum(input_array)
  sum = 0
  input_array.each { |i|
    if i == 0
      break
    end
    sum = sum + i
  }
  sum
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sum a list but ignore any duplicates](#)



Ruby:

```
def sum_no_duplicates(l)
  sum = 0
  l.each {|i|
    puts (1.count i)
    if (1.count i) == 1
      sum = sum + i
    end
  }
  sum
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Check three and two](#)

Ruby:

```
def check_three_and_two(arr)
  count_items = Hash.new

  arr.each { |item|
    if count_items[item].nil?
      count_items[item] = 1
    else
      count_items[item] = count_items[item] + 1
    end
  }

  count_items.each{ |key, value|
    return false if value != 2 and value != 3
  }

  true
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Simple remove duplicates](#)

Ruby:

```
def solve arr
  ret = []
  arr.each {|i|
    ret = ret - [i]
    ret.push(i)
  }
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Return a string's even characters.](#)

Ruby:

```
def even_chars(st)
  return "invalid string" if st.length < 2 or st.length > 99

  ret = []

  st.split("").each_with_index{ |char, index|
    if index % 2 == 1
      ret.push char
    end
  }
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Vowel remover](#)

Ruby:

```
def shortcut(s)
  ret=""
  s.each_char{|c|
    unless c == "a" || c == "e" || c == "i" || c == "o" || c == "u"
      ret += c
    end
  }
end
```

```
ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Sort array by string length](#)

Ruby:

```
def sort_by_length(arr)
  ret = []
  arr.each {|word|
    ret[word.length] = word
  }

  ret2 = []
  ret.each{|i|
    ret2.push(i) unless i.nil?
  }
  ret2
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Add Length](#)

Ruby:

```
def add_length(str)
  ret = []
  str.split(" ").each{|s|
    ret.push(s + " " + s.length.to_s)
  }
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[The 'if' function](#)

Ruby:

```
def _if(bool, ifTrue, ifFalse)
  bool ? ifTrue.call : ifFalse.call
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Calculate average](#)

PHP:

```
function find_average($array) {
  $sum = 0;
  foreach($array as $item) {
    $sum += $item;
  }

  return $sum / count($array);
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Testing 1-2-3](#)

JavaScript:

```
var number=function(a){
  let ret = []
  for (let index in a) {
    ret[index] = (parseInt(index) + 1) + ": " + a[index];
  }
  return ret
}
```

- 3 years ago
- [Refactor](#)

- [Discuss](#)

7 kyu

[Center of the Matrix](#)

Ruby:

```
def center (mat)
  return nil if mat.length % 2 == 0
  middle_element = mat[mat.length / 2]
  return nil if middle_element.length % 2 == 0
  mat[mat.length / 2][middle_element.length / 2]
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Counting Array Elements](#)

Ruby:

```
def count(array)
  ret = {}
  array.each { |item|
    if ret[item].nil?
      ret[item] = 1
    else
      ret[item] = ret[item] + 1
    end
  }
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Largest pair sum in array.](#)

Ruby:

```
def largest_pair_sum(numbers)
  numbers.sort!
  numbers[-1] + numbers[-2]
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Mean vs. Median](#)

Ruby:

```
def mean_vs_median(numbers)
  mean = numbers.reduce(:+)/numbers.length
  numbers = numbers.sort
  median = numbers[numbers.length / 2]
  return "same" if median == mean
  return mean > median ? "mean" : "median"
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Tribonacci Sequence](#)

PHP:

```
function tribonacci($signature, $n) {
  if ($n == 0) {
    return [];
  }
  if ($n == 1) {
    return [$signature[0]];
  }
  if ($n == 2) {
    return [$signature[0], $signature[1]];
  }
  if ($n == 3) {
    return [$signature[0], $signature[1], $signature[2]];
  }

  $cont = 3;
  $ret = $signature;
  while ($n > 3) {
    $sum = 0;
    $n--;
  }
}
```

```

    $sum = end($ret) + prev($ret) + prev($ret);
    array_push($ret, $sum);
    $cont++;
}

return $ret;
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Find Maximum and Minimum Values of a List](#)

PHP:

```

function maximum($array) {
    sort($array);
    return end($array);
}
function minimum($array) {
    sort($array);
    return $array[0];
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Alternate Logic](#)

Ruby:

```

def alt_or(lst)
    return nil if lst.empty?

    ret = lst[0]
    lst.each{|item|
        ret = ret || item
    }

    ret
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Hex to Decimal](#)

Ruby:

```

def hex_to_dec(hex_string)
    hex_string.to_i(16)
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

5 kyu

[Count IP Addresses](#)

Ruby:

```

def ipsBetween(start, ending)
    start_array = start.split(".")
    end_array = ending.split(".")
    return end_array[3].to_i - start_array[3].to_i + 256 * (end_array[2].to_i - start_array[2].to_i) + 256*256*(end_array[1].to_i - start_array[1].to_i)
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find the divisors!](#)

PHP:

```

function divisors($integer) {
    $cont = $integer - 1;
    $ret = [];
    while ($cont > 1) {
        if ($integer % $cont === 0) {
            $ret[] = $cont;
        }
        $cont--;
    }

    if (empty($ret)) {

```

```

    return $integer . " is prime";
}
return array_reverse($ret);
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Count by X](#)

PHP:

```

function countBy($x, $n) {
    $retorno = [];
    $contador = 1;
    $diff = $x;

    while (true) {
        $retorno[] = $x;
        $contador++;

        if ($contador > $n) {
            break;
        }

        print $diff;

        $x = $contador * $diff;
    }

    return $retorno;
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Count of positives / sum of negatives](#)

PHP:

```

function countPositivesSumNegatives($input) {
    if (empty($input)) {
        return [];
    }

    $count = 0;
    $sum = 0;

    foreach ($input as $v) {
        if ($v > 0) {
            $count += 1;
        } else {
            $sum += $v;
        }
    }

    return [$count, $sum];
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[PHP Functions - Type Declarations](#)

PHP:

```

function multiply(int $a, int $b) {
    return $a * $b;
}

function get_profile(Person $p1) {
    $ret = "Name: " . $p1->name . "\n";
    $ret .= "Age: " . $p1->age . "\n";
    $ret .= "Gender: " . $p1->gender . "\n";
    $ret .= "Occupation: " . $p1->occupation;
    return $ret;
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Alphabetically ordered](#)

Ruby:

```

def alphabetic(s)
    s.split("").each_with_index {|char, index|
        if (!(s[index + 1].nil?) and char.ord > s[index + 1].ord)

```

```

    return false
  end
}
true
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Build a train!](#)

JavaScript:

```

function train(s) {
  sum = 0;
  if (s.indexOf("A") > -1) {
    sum += 15;
  }
  if (s.indexOf("B") > -1) {
    sum += 10;
  }
  if (s.indexOf("C") > -1) {
    sum += 7;
  }
  if (s.indexOf("D") > -1) {
    sum += 8
  }

  let n = 1;

  while (n < s.length) {
    if (s[n] == "_") {
      sum += 5;
    }
    n += 1
  }

  return sum;
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[A + B = ?](#)

JavaScript:

```

function howMuchIs(exp){
  let parts = exp.split(" + ")
  parts[0] = parseInt(parts[0])
  parts[1] = parseInt(parts[1])
  sub = parts[0] - parts[1]
  sum = parts[0] + parts[1]

  if (sub == 0) {
    sub = 1
  }

  if (sum == 10) {
    sum = 0
  }

  return parseInt("" + sub + sum)
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Beta

[Return Even Whatever You've Been Given](#)

JavaScript:

```
alwaysEven=n=>n%2?n-1:n
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Draft

[sum\\_of\\_evens - sum\\_of\\_odds](#)

Python:

```

def sum_difference(arr):
  sum_even = 0
  sum_odd = 0

  for num in arr:
    if num % 2 == 0:
      sum_even = sum_even + num

```

```

else:
    sum_odd = sum_odd + num

return sum_even - sum_odd

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Expand the packed usernames \(Boltabek's new job p.1\)](#)

JavaScript:

```

const expandUsernames = data => {
  ret = []

  for (let item of data) {
    let names = item[0].split(",")

    for (let name of names) {
      if (name.trim() !== "") {
        ret.push([name.trim(), item[1]])
      }
    }
  }
  console.log(ret)
  return ret
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Draft

[Perimeter of a Rectangle](#)

JavaScript:

```

var Kata = (function() {
  function Kata() {}

  Kata.getPerimeter = function(length, width) {
    return length * 2 + width * 2
  };

  return Kata;
})();

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Return to Sanity](#)

Ruby:

```

def mystery()
  result = {"sanity": 'Hello'}
  return result
end

```

- 3 years ago
- [Refactor](#)

```

def mystery()
  result = {"sanity": 'Hello'}
  return result
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Sentence Smash](#)

JavaScript:

```

// Smash Words
function smash (words) {
  let ret = ""
  for (let word of words) {
    ret = ret + " " + word
  };

  return ret.trim()
};

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Draft

[Sum of all arguments.](#)

JavaScript:

```
function sum(...args) {
  var total = 0;

  for (let arg of args) {
    if (typeof arg !== "number" || Number.isNaN(arg)) {
      return false
    } else {
      total += arg
    }
  }
  return total;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

```
function sum(){
  var total = 0;

  for (a of arguments) {
    if (! isNaN(parseFloat(a))) {
      total = total + a
    } else {
      return false
    }
  }

  return total;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Two numbers are positive](#)

Python:

```
def two_are_positive(a, b, c):
    if (a > 0 and b > 0 and c > 0):
        return False
    if (a > 0 and b > 0) or (a > 0 and c > 0) or (b > 0 and c > 0):
        return True
    return False
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function twoArePositive($numbers) {
    $totalPositive = 0;

    foreach ($numbers as $number) {
        if ($number > 0) {
            $totalPositive = $totalPositive + 1;
        }
    }

    return $totalPositive == 2;
}

function arePositive($numbers) {
    return twoArePositive($numbers);
}
```

- 3 years ago
- [Refactor](#)

Ruby:

```
def two_are_positive numbers
  cont = 0
  numbers.each {|number|
    cont = cont + 1 if number > 0
  }
  cont == 2
end

def are_positive numbers
  cont = 0
  numbers.each {|number|
    cont = cont + 1 if number > 0
  }
  cont == 2
end
```

- 3 years ago



- [Refactor](#)

Ruby:

```
def two_are_positive numbers
  cont = 0
  numbers.each {|number|
    cont = cont + 1 if number > 0
  }
  cont == 2
end

def are_positive numbers
  cont = 0
  numbers.each {|number|
    cont = cont + 1 if number > 0
  }
  cont == 2
end
```

- 3 years ago
- [Refactor](#)

8 kyu

[Stringy Strings](#)

Ruby:

```
def stringy(size)
  current = "1"
  ret = ""
  while size > 0
    ret += current
    size = size - 1
    if current == "1"
      current = "0"
    else
      current = "1"
    end
  end
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Draft

[Opposite Array](#)

Ruby:

```
def opposite_list(numbers)
  ret = []
  numbers.each { |number|
    ret.push(number * -1)
  }

  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Draft

[Odd One Out](#)

JavaScript:

```
function oddNum(arr) {
  cont = 0
  for (i of arr) {
    if (i % 2 == 1) {
      return cont
    }
    cont++
  }
  return -1
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Count characters in your string](#)

Ruby:

```
def count_chars(s)
  # your code here
  ret = Hash.new

  s.each_char{|char|
```

```

    ret[char] = ret[char].nil? ? 1 : ret[char] + 1
  }

  ret
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Find the smallest integer in the array](#)

JavaScript:

```

class SmallestIntegerFinder {
  findSmallestInt(args) {
    let minor = 1000000000
    for (let current of args) {
      if (current < minor) {
        minor = current
      }
    }
    return minor
  }
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Duplicate Encoder](#)

Ruby:

```

def duplicate_encode(word)
  puts word
  word = word.downcase
  word = word.gsub(" ", "Z")
  word = word.gsub(")", "Y")

  ret = ""
  word.each_char{ |c|
    if (word.scan /#{c}/).size > 1
      ret = ret + ")"
    else
      ret = ret + "("
    end
  }
  ret
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Powers Up](#)

JavaScript:

```

function powersUp(number, upTo) {
  let sum = 0
  let i = 1

  while (i <= upTo) {
    sum = sum + number ** i
    i++
  }
  return sum
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

```

function powersUp(number, upTo) {
  let sum = 0
  let i = 1

  while (i <= upTo) {
    sum = sum + number ** i
    i++
  }
  console.log("##")
  console.log(sum)
  console.log("##")
  return sum
}

```

- 3 years ago
- [Refactor](#)

8 kyu

[Power](#)

JavaScript:

```
function numberToPower(number, power){
  let r = 1
  while (power > 0) {
    power = power - 1
    r = r * number
  }

  return r
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Separate basic types](#)

JavaScript:

```
function separateTypes(input) {
  let r = {}

  for (data of input) {
    if (typeof data === "string") {
      if (typeof r.string === "undefined") {
        r.string = []
      }
      r.string.push(data)
    } else if (typeof data === "boolean") {
      if (typeof r.boolean === "undefined") {
        r.boolean = []
      }
      r.boolean.push(data)
    } else {
      if (typeof r.number === "undefined") {
        r.number = []
      }
      r.number.push(data)
    }
  }

  return r
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Basic Training: Add item to an Array.](#)

Ruby:

```
# add the value "codewars" to the already defined websites array
websites.push("codewars")
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Basic variable assignment](#)

Ruby:

```
a = "code"
b = "wa.rs"
name = a + b
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Holiday I - Temperature in Bali](#)

Ruby:

```
def bareable(heat, humidity)
  return false if humidity > 0.5 or heat >= 36
  return false if 25 < heat and heat < 36 and humidity > 0.4
  true
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[What's the Password?](#)

Ruby:

```
def check_password(password)
  password == "Error404" ? "Correct" : "Error"
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Number to digit tiers](#)

Ruby:

```
def create_array_of_tiers(num)
  return_data = []
  previous_number = ""

  num.to_s.each_char { |n|
    previous_number = previous_number.to_s
    previous_number = previous_number + n
    return_data.push(previous_number)
  }

  return_data
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[FIXME: Get Full Name](#)

JavaScript:

```
class Dinglemouse{
  constructor( f, l ){
    this.firstName = f
    this.lastName = l
  }

  getFullName(){
    return (this.firstName + " " + this.lastName).trim()
  }
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[FIXME: Replace all dots](#)

Ruby:

```
def replaceDots(str)
  str.gsub(/\./, '-')
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Incorrect division method](#)

Ruby:

```
def divide_numbers x, y
  x.to_f / y.to_f
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[How many are smaller than me?](#)

Ruby:

```
def smaller(arr)
  ret = []

  arr.each_with_index { |number, index|
    # puts "--"
    sum = 0
    puts number

    arr.each_with_index { |other, index2|
```

```

    puts other
    puts "---"
    if number > other && index2 > index
      sum = sum + 1
    end
  }

  ret.push(sum)
}

ret
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[How good are you really?](#)

Ruby:

```

def better_than_average(arr, points)
  arr.reduce(:+).to_f / arr.size < points
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Limit string length - 1](#)

Ruby:

```

def solution(st, limit)
  if limit < st.length
    st[limit..-1] = ""
    st = st + "..."
  end

  st
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[max diff - easy](#)

Ruby:

```

def max_diff(lst)
  return 0 if lst.length < 1
  lst = lst.sort
  lst[-1] - lst[0]
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Convert number to reversed array of digits](#)

Ruby:

```

def digitize(n)
  r = []
  n.to_s.split("").reverse_each{|i| r.push(i.to_i)}
  r
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

```

def digitize(n)
  retorno = []

  n.digits.each {|n1|
    retorno.push n1
  }

  retorno
end

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Array element parity](#)

Ruby:

```
def solve(arr)
  arr.each { |i|
    return i unless arr.include? i * -1
  }
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Convert to Binary](#)

Ruby:

```
def to_binary(n)
  n.to_s(2).to_i
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Are arrow functions odd?](#)

Ruby:

```
def odds(values)
  ret = []
  values.each {|value|
    if value.odd?
      ret.push(value)
    end
  }
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function odds(values){
  let r = []
  for (const i of values) {
    if (i % 2 == 1) {
      r.push(i)
    }
  }
  return r
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Third Angle of a Triangle](#)

JavaScript:

```
function otherAngle(a, b) {
  return 180-a-b;
}
```

- 5 years ago
- [Refactor](#)

```
function otherAngle(a, b) {
  return 180 - a - b;
}
```

- 5 years ago
- [Refactor](#)

```
function otherAngle(a, b) {
  return 180 - a - b;
}
```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

```
function otherAngle(a, b) {
  return 180-a-b;
}
```

- 7 years ago
- [Refactor](#)

PHP:

```
function otherAngle($a, $b) {
  return 180-$a-$b;
}
```

- 7 years ago
- [Refactor](#)

```
function otherAngle($a, $b) {
  return 180 - $a - $b;
}
```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

```
function otherAngle($a, $b) {
  return 180 - $a - $b;
}
```

- 6 years ago
- [Refactor](#)

Python:

```
def other_angle(a, b):
  return 180-a-b
```

- 5 years ago
- [Refactor](#)

```
def other_angle(a, b):
  return 180 - a - b
```

- 5 years ago
- [Refactor](#)

```
def other_angle(a, b):
  return 180 - a - b;
```

- 6 years ago
- [Refactor](#)

```
def other_angle(a, b):
  return 180 - a - b;
```

- 6 years ago
- [Refactor](#)

R:

```
other_angle <- function(a, b){
  180 - a - b
}
```

- 5 years ago
- [Refactor](#)

```
other_angle <- function(a, b){
  return (180 - a - b)
}
```

- 6 years ago
- [Refactor](#)

```
other_angle <- function(a, b){
  return (180 - a - b)
}
```

- 6 years ago
- [Refactor](#)

C++:

```
class Triangle {
public:
  static int otherAngle(int a, int b) {
    return 180-a-b;
  }
};
```

- 5 years ago
- [Refactor](#)

```
class Triangle {
public:
    static int otherAngle(int a, int b) {
        return 180 - a - b;
    }
};
```

- 6 years ago
- [Refactor](#)

```
class Triangle {
public:
    static int otherAngle(int a, int b) {
        return 180 - a - b;
    }
};
```

- 6 years ago
- [Refactor](#)

Ruby:

```
def other_angle(a, b)
    180 - a - b
end
```

- 6 years ago
- [Refactor](#)

```
def other_angle(a, b)
    180 - a - b
end
```

- 6 years ago
- [Refactor](#)

Solidity:

```
pragma solidity ^0.4.19;

contract ThirdAngle {
    function otherAngle(int angle1, int angle2) returns (int) {
        // TODO your code here
        return 180 - angle1 - angle2;
    }
}
```

- 5 years ago
- [Refactor](#)

```
pragma solidity ^0.4.19;

contract ThirdAngle {
    function otherAngle(int angle1, int angle2) returns (int) {
        return (180 - angle1 - angle2);
    }
}
```

- 6 years ago
- [Refactor](#)

```
pragma solidity ^0.4.19;

contract ThirdAngle {
    function otherAngle(int angle1, int angle2) returns (int) {
        // TODO your code here
        int a1 = angle1;
        int a2 = angle2;
        return 180 - a1 - a2;
    }
}
```

- 6 years ago
- [Refactor](#)

TypeScript:

```
export const otherAngle = (a, b) => {
    return 180 - a - b;
}
```

- 5 years ago
- [Refactor](#)

```
export const otherAngle = (a, b) => {
    return 180 - a - b;
}
```

- 5 years ago
- [Refactor](#)

```
export const otherAngle = (a, b) => {
    return 180 - a - b;
}
```

- 6 years ago



- [Refactor](#)

C#:

```
using System;

public static class Kata
{
    public static int OtherAngle(int a, int b)
    {
        return 180-a-b;
    }
}
```

- 5 years ago
- [Refactor](#)

```
using System;

public static class Kata
{
    public static int OtherAngle(int a, int b)
    {
        return 180 - a - b;
    }
}
```

- 5 years ago
- [Refactor](#)

```
using System;

public static class Kata
{
    public static int OtherAngle(int a, int b)
    {
        return 180 - a - b;
    }
}
```

- 6 years ago
- [Refactor](#)

Go:

```
package kata

func OtherAngle(a int, b int) int {
    return 180 - a - b
}
```

- 3 years ago
- [Refactor](#)

8 kyu

[Sum of differences in array](#)

Ruby:

```
def sum_of_differences(arr)
  arr = arr.sort

  diff = 0
  max = arr.size - 1
  arr.each_with_index do |item, i|
    unless (arr[i+1]).nil?
      diff = diff + arr[i+1] - item
    end
  end
  diff
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Pairs of integers from 0 to n](#)

Ruby:

```
def generate_pairs(n)
  i = 0
  j = 0
  r = []
  while i <= n
    j = 0
    while j <= n
      if (j >= i)
        r.push([i, j])
      end
      j = j + 1
    end
    i = i + 1
    puts i
  end
end
```

```

if n == 0 and r.empty?
  return [[0, 0]]
else
  return r
end
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Simple Fun #132: Number Of Carries](#)

Ruby:

```

def number_of_carries(a, b)
  sorted = [a,b].sort
  accumulator = 0
  sorted[0] = sorted[0].to_s
  sorted[1] = (sorted[1].to_s).reverse
  sorted[0] = (sorted[0].rjust(sorted[1].size, "0")).reverse
  puts sorted[0]
  puts sorted[1]

  sum = 0
  sorted[1].split("").each_with_index {|n, i|
    if (sorted[0][i].to_i + sorted[1][i].to_i + accumulator >= 10 )
      sum = sum + 1
      accumulator = 1
    else
      accumulator = 0
    end
  }
  sum
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Are You Playing Banjo?](#)

JavaScript:

```

function areYouPlayingBanjo(name) {
  if (name.toLowerCase().substring(0,1) == "r" ) {
    return name + " plays banjo";
  }
  return name + " does not play banjo"
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```

def are_you_playing_banjo(name)
  name1 = name.downcase
  return name + " plays banjo" if name1[0] == "r"
  return name + " does not play banjo"
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Evens and Odds](#)

JavaScript:

```

function evensAndOdds(num){
  if (num % 2 == 0) {
    return (num >>> 0).toString(2)
  }
  return num.toString(16)
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```

def evensAndOdds(num)
  if (num % 2 == 0)
    return num.to_s(2)
  end

  return num.to_s(16)
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Maximum Product](#)

PHP:

```
function adjacentElementsProduct($array) {
    $max = -10000000;

    foreach ($array as $index => $value) {
        if (isset($array[$index+1])) {
            $m = $value * $array[$index + 1];

            if ($m > $max) $max = $m;
        }
    }

    return $max;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def adjacent_element_product(array)
    max = -10000000;

    array.each_with_index{ | value, key |
        unless (array[key+1].nil?)
            m = value * array[key + 1];

            if (m > max)
                max = m
            end
        end
    }

    max
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[String array duplicates](#)

Ruby:

```
def dup(arr)
    ret = []
    arr.each{ |i|
        prev = ""
        ret.push("")
        i.each_char{|c|
            if prev == c
                prev = c
                c = ""
            else
                prev = c
            end
        }
        ret[-1] = ret[-1] + c
    }
    ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Your order, please](#)

Ruby:

```
def order(words)
    words = words.split(" ")

    r = []
    words.each{ |word|
        word.each_char { |char|
            if char.to_i != 0
                r[char.to_i - 1] = word
            end
        }
    }

    r.join(" ")
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[String matchup](#)

Ruby:

```
def solve(a,b)
  max = a.count
  ret = []

  b.each { |w|
    i = 0
    total = 0
    while (i < max)
      if (w == a[i])
        total = total + 1
      end
      i = i + 1
    end

    ret.push(total)
  }
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Simple consecutive pairs](#)

Ruby:

```
def pairs arr
  cont = 0
  r = 0
  while true
    if arr[cont + 1].to_i - arr[cont].to_i == 1 or arr[cont].to_i - arr[cont + 1].to_i == 1
      r = r + 1
    end
    cont = cont + 2
    if arr[cont].nil?
      break
    end
  end
  r
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Return the first M multiples of N](#)

Ruby:

```
def multiples(m, n)
  cont = 1
  r = []
  while cont <= m
    r.push(n * cont)
    cont = cont + 1
  end
  return r
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Even numbers in an array](#)

Ruby:

```
def even_numbers(arr,n)
  r = []
  arr.each { |i|
    if i.even?
      r.push i
    end
  }
  r = r.reverse
  r = r.slice(0, n)
  r.reverse
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Calculate BMI](#)

Ruby:

```
def bmi(weight, height)
  bmi = weight / (height ** 2)
  if bmi <= 18.5
    return "Underweight"
  elsif bmi <= 25.0
    return "Normal"
  elsif bmi <= 30.0
    return "Overweight"
  end

  return "Obese"
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Largest 5 digit number in a series](#)

Ruby:

```
def solution(digits)
  major = 0
  digits = digits.split("")
  digits.each_with_index{ |n, index|
    number = (digits[index].to_s + digits[index + 1].to_s + digits[index + 2].to_s + digits[index + 3].to_s + digits[index + 4].to_s).to_i
    if number > major
      major = number
    end
  }
  major
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function solution(string $s): int {
    $major = 0;
    $length = strlen($s);
    $number = 0;

    for ($i = 0; $i < $length; $i++) {
        if ($i + 4 >= $length) {
            break;
        }

        $number = $s[$i] . $s[$i + 1] . $s[$i + 2] . $s[$i + 3] . $s[$i + 4];

        if ($number > $major) {
            $major = $number;
        }
    }

    return $major;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Form The Largest](#)

PHP:

```
function maxNumber($n) {
    $n = str_split($n);
    rsort($n);
    return (int) implode("", $n);
}
```

- 3 years ago
- [Refactor](#)

```
function maxNumber($n) {
    $n = str_split($n);
    sort($n);
    var_dump($n);
    $n = array_reverse($n);
    return (int) implode("", $n);
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def max_number(n)
  n.to_s.split("").sort.reverse.join("").to_i
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Product Array \(Array Series #5\)](#)

Ruby:

```
def product_array(numbers)
  ret = []
  numbers.each {|n|
    ret.push(numbers.inject("*/") / n)
  }
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function productArray($nums) {
  $retArray = [];
  foreach ($nums as $index => $value) {
    $ret = 1;
    foreach ($nums as $index2 => $value2) {
      if ($index == $index2) {
        continue;
      } else {
        $ret = $ret * $value2;
      }
    }
    array_push($retArray, $ret);
  }
}

return $retArray;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Odd or Even?](#)

JavaScript:

```
function oddOrEven(array) {
  sum = 0;
  for (var i in array) {
    sum = sum + array[i];
    console.log(array[i]);
  }

  if (sum % 2 == 0) {
    return "even";
  }

  return "odd";
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Groovy:

```
class Kata{
  static String oddOrEven(list) {
    Integer sum = 0

    for (item in list) {
      sum = sum + item
    }

    if (sum % 2 == 0) {
      return "even"
    }

    return "odd"
  }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def odd_or_even(array)
  sum = 0
  array.each { |a| sum+=a }
  return sum.even? ? "even": "odd"
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Shortest Word](#)

Python:

```
def find_short(s):
  menor = None
  palavras = s.split(' ')
  for palavra in palavras:
    if (menor == None or len(palavra) < menor):
      menor = len(palavra)
  return menor
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def find_short(s)
  l = s.split(" ")
  minimum = 1000000
  l.each { |p|
    size = p.size
    if (size < minimum)
      minimum = size
    end
  }
  minimum
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

```
def find_short(s)
  s = s.split(" ")
  min_length = 1000000

  s.each {|item|
    size = item.size
    if (size < min_length)
      min_length = size
    end
  }
  min_length
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Dollars and Cents](#)

Ruby:

```
def format_money(amount)
  amount = amount.round(2)

  amount_string = amount.to_s

  pointPosition = amount_string.index(".")

  if pointPosition.nil?
    return "$" + amount_string + ".00"
  end

  if amount_string.size - pointPosition <= 2
    amount_string = amount_string + "0"
  end

  ret = "$" + amount_string

  ret_string = ret
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function format_money(float $amount): string {
  return "$" . number_format($amount, 2, ".", "");
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

```
function format_money(float $amount): string {
  return '$' . number_format($amount, 2, '.', '');
}
```

- 3 years ago
- [Refactor](#)

8 kyu

[Super Duper Easy](#)

Ruby:

```
def problem x
  if x == "hello" or x == "" or x == "RyanIsCool"
    return "Error"
  end
  x * 50 + 6
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

```
def problem x
  if x == "hello" or x == "" or x == "RyanIsCool"
    return "Error"
  end
  puts x == ""
  x * 50 + 6
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function problem(x){
  if (typeof x == "string") {
    return "Error"
  }
  return x * 50 + 6
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[NBA full 48 minutes average](#)

Ruby:

```
def nba_extrap(ppg, mpg)
  return 0 if mpg == 0
  ppg = ppg.to_f
  mpg = mpg.to_f
  r = (ppg * 48) / mpg
  return r.round(1)
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Ensure question](#)

Ruby:

```
def ensure_question(s)
  if s.end_with? "?"
    return s
  end

  return s + "?"
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Difference of Volumes of Cuboids](#)

Ruby:



```
def find_difference(a, b)
  res = a[0] * a[1] * a[2] - (b[0] * b[1] * b[2])

  if res < 0
    res = res * -1
  end

  res
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Rotate to the max](#)

Ruby:

```
def rotate_to_max(n)
  n = n.to_s
  n_array = n.split("")
  a = n_array.sort
  a.reverse!
  a.join('').to_i
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Simple Fun #176: Reverse Letter](#)

Ruby:

```
def reverse_letter(string)
  ret = ""
  string.each_char{|char|
    char = char.downcase()
    unless char.scan(/[a-z]+/).empty?
      ret += char
    end
  }
  ret.reverse
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[JavaScript Array Filter](#)

Ruby:

```
def get_even_numbers(arr)
  ret = []
  arr.each {|item|
    if item % 2 == 0
      ret.push(item)
    end
  }
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sum of Minimums!](#)

Ruby:

```
def sum_of_minimums(numbers)
  sum = 0
  numbers.each {|array_numbers|
    sum = sum + array_numbers.min
  }
  sum
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[CubeSummation](#)

Ruby:

```
def cube_sum(n, m)
  array_sorted = [n, m].sort
```

```

sum = 0

i = array_sorted[0] + 1
while (i <= array_sorted[1]) do
  sum = i ** 3 + sum
  i = i + 1
end

sum
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

```

def cube_sum(n, m)
  array_sorted = [n, m].sort

  sum = 0

  i = array_sorted[0] + 1
  while (i <= array_sorted[1]) do
    sum = i ** 3 + sum
    puts i
    puts sum
    puts "--"
    i = i + 1
  end

  sum
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Equalize the array!](#)

Ruby:

```

def equalize(arr)
  if arr.empty?
    return []
  end

  ret = []
  diff = - arr.first

  arr.each{|i|
    item = (i + diff).to_s

    if i + diff < 0
      a = item
    else
      a = "+" + item
    end

    ret.push(a)
  }

  ret
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Find the position!](#)

Ruby:

```

def position(alphabet)
  "Position of alphabet: " + (alphabet.ord - 96).to_s
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Stones on the Table](#)

Ruby:

```

def solution(stones)
  total = 0
  stones.split("").each_with_index { |stone, index|
    if stones[index + 1] != nil
      if stones[index + 1] == stone
        total = total + 1
      end
    end
  }
  total
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[CSV representation of array](#)

Ruby:

```
def to_csv_text(array)
  ret = ""
  array.each{|internal|
    internal.each{|item|
      ret = ret + item.to_s + ", "
    }
    ret = ret[0..-2] + "\n"
  }
  ret[0..-2]
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Maximum Triplet Sum \(Array Series #7\)](#)

Ruby:

```
def max_tri_sum(numbers)
  numbers = numbers.uniq.sort.reverse
  numbers[0] + numbers[1] + numbers[2]
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sum of the first nth term of Series](#)

PHP:

```
function series_sum($n) {
  if ($n == 0) return "0.00";

  $start = 4;
  $increment = 3;
  $sum = 1;

  while ($n > 1) {
    $sum = $sum + 1 / (($n * 2) + $n - 2);
    $n--;
  }

  return number_format($sum, 2, ".", ",");
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Boiled Eggs](#)

Ruby:

```
def cooking_time(eggs)
  puts eggs
  if eggs == 0
    return 0
  end

  if (8 % 8 == 0)
    eggs = eggs - 1
  end

  ((eggs / 8) + 1) * 5
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

5 kyu

[Sort arrays - 3](#)

Ruby:

```
# input: courses - array of course-names "name-yyymm"
# output: sorted by "yyymm", then "name"
def sortme( courses )
  ret = []
  courses.each{ |course|
```

```

    course = course.split("-")
    ret.push([course[1], course[0]])
  }

  ret.sort!
  ret2=[]

  ret.each{ |course|
    ret2.push(course[1] + "-" + course[0])
  }

  ret2
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[See You Next Happy Year](#)

Ruby:

```

def next_happy_year(year)
  original_year = year.to_s
  while true
    year = year + 1

    if year.to_s.split("").uniq.size == original_year.split("").size
      break
    end
  end

  year
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Binary Addition](#)

Ruby:

```

def add_binary(a,b)
  (a+b).to_s(2)
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Build a square](#)

Ruby:

```

def generate_shape(n)
  r = ""
  e = 0
  i = 0
  while e < n
    i = 0
    while i < n
      r += "+"
      i = i + 1
    end
    r += "\n"
    e = e + 1
  end
  r[0..-2]
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Form The Minimum](#)

PHP:

```

function minValue($arr) {
  $arr = array_unique($arr);
  sort($arr);
  return (int) implode("", $arr);
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def min_value(digits)
  r = []
  digits.each{|digit|
    unless r.include? digit
      r.push(digit)
    end
  }
  r.sort!
  r.join("").to_i
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Array.diff](#)

Ruby:

```
def array_diff(a, b)
  r = []
  a.each { |i|
    unless (b.include? i)
      r.push(i)
    end
  }
  r
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Complementary DNA](#)

Ruby:

```
def DNA_strand(dna)
  r = ""
  dna.each_char { |c|
    if c == "A"
      r = r + "T"
    elsif c == "T"
      r = r + "A"
    elsif c == "G"
      r = r + "C"
    elsif c == "C"
      r = r + "G"
    end
  }
  r
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Halving Sum](#)

Ruby:

```
def halving_sum(n)
  sum = 0
  while (n >= 1)
    sum = sum + n
    n = n / 2
  end

  sum
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

```
def halving_sum(n)
  sum = 0
  while (n >= 1)
    sum += n;
    n = (n / 2).floor
  end

  sum
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

C++:

#include &lt;math.h&gt;

```

unsigned halving_sum(unsigned n) {
    int sum = 0;

    while (n >= 1) {
        sum += n;
        n = floor(n / 2);
    }

    return sum;
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[16+18=214](#)

Ruby:

```

def silly_add(a, b)
    cont = 0

    a = a.to_s
    b = b.to_s

    if (a.size > b.size)
        c = a
        d = b
        d = d.rjust(a.size, "0")
    else
        c = b
        d = a
        d = d.rjust(b.size, "0")
    end

    sum = ""
    cont = c.size - 1
    while true
        char = c[cont]
        sum = (char.to_i + d[cont].to_i).to_s + sum
        puts(sum)
        cont = cont - 1

        if cont == -1
            break
        end
    end

    sum.to_i
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Jumping Number \(Special Numbers Series #4\)](#)

Ruby:

```

def jumping_number(n)
    n = n.to_s.split("")
    jumping = true

    if n.length == 1
        return "Jumping!!"
    end

    loop = n.each_with_index{ |number, index|
        number = number.to_i
        if index == 0
            next if number - n[index + 1].to_i == 1 or number - n[index + 1].to_i == -1
            jumping = false
            break
        end

        next if number - n[index + 1].to_i == 1 or number - n[index + 1].to_i == -1 or number - n[index - 1].to_i == 1 or number - n[index - 1].to_i == -1
        jumping = false
        break
    }

    return "Not!!" unless jumping

    return "Jumping!!"
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Grasshopper - Terminal game move function](#)

Ruby:

```

def move (position, roll)
    roll * 2 + position
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[MakeUpperCase](#)

Ruby:

```
def make_upper_case(str)
  str.upcase
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sum even numbers](#)

Ruby:

```
def sum_even_numbers(seq)
  sum = 0
  seq.each {|number|
    if number % 2 == 0
      sum = sum + number
    end
  }
  sum
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[L2: Triple X](#)

JavaScript:

```
function tripleX(str){
  const posXxx = str.indexOf("xxx")
  const posX = str.indexOf("x")

  return posX == posXxx && posXxx != -1
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Switch it Up!](#)

Ruby:

```
def switch_it_up(number)
  if number == 1
    return "One"
  elsif number == 2
    return "Two"
  elsif number == 3
    return "Three"
  elsif number == 4
    return "Four"
  elsif number == 5
    return "Five"
  elsif number == 6
    return "Six"
  elsif number == 7
    return "Seven"
  elsif number == 8
    return "Eight"
  elsif number == 9
    return "Nine"
  else
    return "Zero"
  end
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find the middle element](#)

Ruby:

```
def gimme(input_array)
  ordered = input_array.sort
```

```

middle = nil
input_array.each_with_index {|item, index|
  if item == ordered[1]
    middle = index
    break
  end
}
return middle
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find Your Villain Name](#)

Ruby:

```

def get_villain_name birthday
  birthday_string = birthday.to_s
  month = birthday_string[5..6]

  if (month == "01")
    string = "The Evil"
  end

  if (month == "02")
    string = "The Vile"
  end

  if (month == "03")
    string = "The Cruel"
  end

  if (month == "04")
    string = "The Trashy"
  end

  if (month == "05")
    string = "The Despicable"
  end

  if (month == "06")
    string = "The Embarrassing"
  end

  if (month == "07")
    string = "The Disreputable"
  end

  if (month == "08")
    string = "The Atrocious"
  end

  if (month == "09")
    string = "The Twirling"
  end

  if (month == "10")
    string = "The Orange"
  end

  if (month == "11")
    string = "The Terrifying"
  end

  if (month == "12")
    string = "The Awkward"
  end

  day = birthday_string[9]

  if (day == "0")
    string += " Mustache"
  end

  if (day == "1")
    string += " Pickle"
  end

  if (day == "2")
    string += " Hood Ornament"
  end

  if (day == "3")
    string += " Raisin"
  end

  if (day == "4")
    string += " Recycling Bin"
  end

  if (day == "5")
    string += " Potato"
  end

  if (day == "6")
    string += " Tomato"
  end

  if (day == "7")
    string += " House Cat"
  end

```



```

end

if (day == "8")
  string += " Teaspoon"
end

if (day == "9")
  string += " Laundry Basket"
end

return string
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Shared Bit Counter](#)

Ruby:

```

def shared_bits(a, b)
  binnary_a = a.to_s(2)
  binnary_b = b.to_s(2)
  binnary_a = binnary_a.rjust(binnary_b.size, "0")
  binnary_b = binnary_b.rjust(binnary_a.size, "0")

  count_1 = 0
  position_count = 0
  binnary_a.each_char { |c|
    if c === binnary_b[position_count] and c == "1"
      count_1 = count_1 + 1
    end

    position_count = position_count + 1
  }

  count_1 >= 2
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Valid Spacing](#)

Ruby:

```

def valid_spacing(s)
  s.strip().gsub(/ /, "") === s
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sum - Square Even, Root Odd](#)

Ruby:

```

def sum_square_even_root_odd(nums)
  sum = 0

  nums.each { |num|
    if num % 2 === 0
      sum = sum + (num ** 2)
    else
      sum = sum + (Math.sqrt(num))
    end
  }

  sum.round(2)
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Tidy Number \(Special Numbers Series #9\)](#)

PHP:

```

function tidyNumber($n) {
  $array = str_split($n);
  $previous = null;

  foreach ($array as $number) {
    if (is_null($previous)) {
      $previous = $number;
      continue;
    }
  }
}

```

```

    if ($number < $previous) return false;
    $previous = $number;
  }
  return true;
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Square\(n\) Sum](#)

JavaScript:

```

function squareSum(numbers){
  let retorno = 0;
  for (let i of numbers) {
    retorno += Math.pow(i, 2);
  }
  return retorno;
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Lost number in number sequence](#)

JavaScript:

```

function findDeletedNumber(arr, mixArr) {
  for (n of arr) {
    if (mixArr.indexOf(n) === -1) {
      return n
    }
  }
  return 0
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Function 2 - squaring an argument](#)

Ruby:

```

# Write the "square"-function here
def square(number)
  number ** 2
end

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Larger Product or Sum](#)

JavaScript:

```

function sumOrProduct(array, n) {
  let sortedArray = array.sort(function(a,b) {return a-b})
  let product = 1
  let sum = 0

  for (var i = 0 ; i < n ; i++) {
    product = product * sortedArray[i]
  }

  for (i = 0 ; i < n ; i++) {
    sum = sum + sortedArray[sortedArray.length - i - 1]
  }

  if (product > sum) {
    return "product"
  } else if (product < sum) {
    return "sum"
  }
  return "same"
}

```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Automorphic Number \(Special Numbers Series #6\)](#)

Ruby:

```
def automorphic(n)
  d = n ** 2

  if d.to_s.include? n.to_s
    return "Automorphic"
  end

  return "Not!!"
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Factorial](#)**PHP:**

```
function factorial(int $n): int {
  if ($n == 0) return 1;

  if ($n < 0 || $n > 12) {
    throw new RangeException ;
  }

  $result = 1;
  for ($i = 1; $i <= $n ; $i++) {
    $result = $result * $i;
  }

  return $result;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Fix string case](#)**Ruby:**

```
def solve s
  contLower = 0
  contUpper = 0
  s.each_char { |c|
    if c.match(/[a-z]/)
      contLower = contLower + 1
    elsif c.match(/[A-Z]/)
      contUpper = contUpper + 1
    end
  }

  puts "contLower: " + contLower.to_s
  puts "contUpper: " + contUpper.to_s

  if (contLower >= contUpper)
    s.downcase!
  elsif (contUpper > contLower)
    s.upcase!
  end

  s
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Backspaces in string](#)**Ruby:**

```
def clean_string(string)
  ret = ""
  string.each_char { |c|
    if (c == "#")
      ret = ret[0..-2]
    else
      ret = ret + c
    end
  }
  ret
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Triple Trouble](#)**JavaScript:**

```
function tripleTrouble(one, two, three){
  let r = ""

  for (let i in one) {
    r += one[i] + two[i] + three[i]
  }

  return r
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[SpeedCode #2 - Array Madness](#)

JavaScript:

```
function arrayMadness(a, b) {
  let soma1=0;
  let soma2=0;

  for (let i of a) {
    soma1 = soma1 + Math.pow(i,2)
  }

  for (let k of b) {
    soma2 = soma2 + Math.pow(k,3)
  }

  return soma1 > soma2 ? true : false;
}
```

- 4 years ago
- [Refactor](#)

8 kyu

[Simple multiplication](#)

Ruby:

```
def simple_multiplication(number)
  number % 2 == 1 ? number * 9 : number * 8
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Reverse a Number](#)

JavaScript:

```
function reverseNumber(n) {
  let s = n.toString();
  let r = parseInt(s.split("").reverse().join(""));
  if (n < 0) {
    return r * -1;
  }

  return r;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def reverse_number(n)
  n = n.to_s
  if n.slice(0,1) == "-" then
    negativo = true
    n.slice(1, 99)
  end
  n.reverse!

  if negativo then
    n = "-" + n
  end

  n.to_i
end
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Grasshopper - Summation](#)

PHP:

```
function summation($n) {
  $soma = 0;
  for ($i = $n; $i > 0 ; $i--) {
    $soma += $i;
  }

  return $soma;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Groovy:

```
class GrassHopper {
  def static int summation(n) {
    def sum = 0
    Integer i = 0

    for (i = n; i > 0 ; i--) {
      sum += i
    }

    return sum
  }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def summation(num)
  current = 0
  sum = 0

  while (current <= num)
    sum = sum + current
    current = current + 1
  end

  sum
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Closest elevator](#)

Python:

```
def elevator(left, right, call):
  p1 = abs(call - left)
  p2 = abs(call - right)

  if (p1 < p2) :
    return 'left'
  else:
    return 'right'
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

```
def elevator(left, right, call):
  p1 = abs(call - left)
  p2 = abs(call - right)

  if (p1 < p2):
    return "left"
  else:
    return "right"
```

- 5 years ago
- [Refactor](#)

Ruby:

```
def elevator(left, right, call)
  p1 = call - left
  p1 = p1.abs

  p2 = call - right
  p2 = p2.abs

  if (p1 < p2)
    return "left"
  else
    return "right"
  end
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Misspelled word](#)

JavaScript:

```
var misspelled = function(word1, word2)
{
  let diferencia = word1.length - word2.length;

  if (diferencia > 1 && diferencia < -1) {
    return false;
  }

  let arrayWord1 = word1.split("");
  let ocurrencias = 0;

  for (c of arrayWord1) {
    if (word2.indexOf(c) == -1) {
      ocurrencias = ocurrencias + 1;
    }
  }

  if (ocurrencias > 1) {
    return false;
  }

  let arrayWord2 = word2.split("");
  ocurrencias = 0;

  for (c of arrayWord2) {
    if (word1.indexOf(c) == -1) {
      ocurrencias = ocurrencias + 1;
    }
  }

  if (ocurrencias > 1) {
    return false;
  }

  return true;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Exclamation marks series #6: Remove n exclamation marks in the sentence from left to right](#)

JavaScript:

```
function remove(s,n){
  while (n > 0) {
    s = s.replace("!", "");
    n = n-1;
  }

  return s;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Expressions Matter](#)

Ruby:

```
def expression_matter(a,b,c)
  r = Array.new

  r[0] = a + b + c
  r[1] = (a * b) + c
  r[2] = a + (b * c)
  r[3] = a * b * c
  r[4] = (a + b) * c
  r[5] = a * (b + c)

  r.sort()[5]
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Is the date today](#)

JavaScript:

```
function isToday(date) {
  let currentDate = new Date;
  return date.getDay() == currentDate.getDay() && date.getMonth() == currentDate.getMonth() && date.getYear() == currentDate.getYear();
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Descending Order](#)

Ruby:

```
def descending_order(n)
  n.to_s.split("").sort().reverse().join("").to_i
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Grasshopper - Grade book](#)

PHP:

```
function getGrade($a, $b, $c) {
  $mean = ($a + $b + $c) / 3;

  if ($mean >= 90) {
    return "A";
  }

  if ($mean >= 80) {
    return "B";
  }

  if ($mean >= 70) {
    return "C";
  }

  if ($mean >= 60) {
    return "D";
  }

  return "F";
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Century From Year](#)

PHP:

```
function centuryFromYear($year)
{
  $divisionResult = (int) $year / 100;
  $remainder = (int) $year % 100;

  return $remainder > 0 ? floor($divisionResult + 1) : floor($divisionResult);
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Grasshopper - Debug](#)

PHP:

```
function weatherInfo(int $temp): string
{
  $c = convertToCelsius($temp);
  if($c < 0) {
    return ($c . " is freezing temperature");
  } else {
    return ($c . " is above freezing temperature");
  }
}

function convertToCelsius(int $temperature): int
{
  return ($temperature - 32) * (5/9);
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Count the divisors of a number](#)

JavaScript:

```
function getDivisorsCnt(n){
  let total = 0 ;
  let contador = 1;
  while (contador <= n) {
    if (n % contador == 0) {
      total++;
    }

    contador++;
  }

  return total;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

C#:

```
public class Kata
{
  public static int Divisors(int n)
  {
    int total = 0 ;
    int contador = 1;
    while (contador <= n) {
      if (n % contador == 0) {
        total++;
      }

      contador++;
    }

    return total;
  }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Python:

```
def divisors(n):
  total = 0 ;
  contador = 1;
  while (contador <= n):
    if (n % contador == 0):
      total = total + 1;

    contador = contador + 1;

  return total;
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def divisors(n)
  current = 1
  total = 0
  while (current <= n) do
    if n % current == 0
      total = total + 1
    end
    current = current + 1
  end
  total
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Vowel Count](#)

JavaScript:

```
function getCount(str) {
  let matches = str.match(/[aeiou]/g);

  return matches == null ? 0 : matches.length;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)



Ruby:

```
def getCount(inputStr)
  r = 0
  cont = 0
  while (cont < inputStr.size) do
    if (inputStr[cont] == "a" || inputStr[cont] == "e" || inputStr[cont] == "i" || inputStr[cont] == "o" || inputStr[cont] == "u")
      r = r + 1
    end
    cont = cont + 1
  end
  r
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

```
def getCount(inputStr)
  cont = 0
  r = 0
  while (cont < inputStr.size) do
    if (inputStr[cont] == "a" || inputStr[cont] == "e" || inputStr[cont] == "i" || inputStr[cont] == "o" || inputStr[cont] == "u")
      r = r + 1
    end
    cont = cont + 1
  end
  r
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[String ends with?](#)

Ruby:

```
def solution(str, ending)
  puts str[str.size - ending.size .. str.size]
  str[str.size - ending.size .. str.size] === ending
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Returning Strings](#)

Ruby:

```
def greet(name)
  "Hello, " + name + " how are you doing today?"
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

```
def greet(name)
  "Hello, " + name + " how are you doing today?"
end
```

- 4 years ago
- [Refactor](#)

Groovy:

```
class Wherever {
  static String translate(name) {
    "Hello, " + name + " how are you doing today?"
  }
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Return the day](#)

JavaScript:

```
function whatday(n) {
  if (n == 1) {
    return "Sunday"
  }
  else if (n == 2) {
    return "Monday"
  }
  else if (n == 3) {
    return "Tuesday"
  }
}
```

```

else if (n == 4) {
  return "Wednesday"
}
else if (n == 5) {
  return "Thursday"
}
else if (n == 6) {
  return "Friday"
}
else if (n == 7) {
  return "Saturday"
}

return "Wrong, please enter a number between 1 and 7"
}

```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

```

function whatday(num) {
  if (num == 1) {
    return "Sunday";
  } else if (num == 2) {
    return "Monday";
  } else if (num == 3) {
    return "Tuesday";
  } else if (num == 4) {
    return "Wednesday";
  } else if (num == 5) {
    return "Thursday";
  } else if (num == 6) {
    return "Friday";
  } else if (num == 7) {
    return "Saturday";
  }

  return "Wrong, please enter a number between 1 and 7";
}

```

- 6 years ago
- [Refactor](#)

```

function whatday(weekday) {
  if (weekday == 1) return "Sunday";
  if (weekday == 2) return "Monday";
  if (weekday == 3) return "Tuesday";
  if (weekday == 4) return "Wednesday";
  if (weekday == 5) return "Thursday";
  if (weekday == 6) return "Friday";
  if (weekday == 7) return "Saturday";
  return 'Wrong, please enter a number between 1 and 7';
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```

def what_day?(n)
  if n == 1
    return "Sunday"
  elsif n == 2
    return "Monday"
  elsif n == 3
    return "Tuesday"
  elsif n == 4
    return "Wednesday"
  elsif n == 5
    return "Thursday"
  elsif n == 6
    return "Friday"
  elsif n == 7
    return "Saturday"
  end

  return "Wrong, please enter a number between 1 and 7"
end

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

```

def what_day?(n)
  if n == 1 then
    return "Sunday"
  elsif n == 2 then
    return "Monday"
  elsif n == 3 then
    return "Tuesday"
  elsif n == 4 then
    return "Wednesday"
  elsif n == 5 then
    return "Thursday"
  elsif n == 6 then
    return "Friday"
  elsif n == 7 then
    return "Saturday"
  end
end

```

end

"Wrong, please enter a number between 1 and 7"

end

- 5 years ago
- [Refactor](#)
- [Discuss](#)

def what\_day?(n)

```

  if n == 1
    return "Sunday"
  elsif n == 2
    return "Monday"
  elsif n == 3
    return "Tuesday"
  elsif n == 4
    return "Wednesday"
  elsif n == 5
    return "Thursday"
  elsif n == 6
    return "Friday"
  elsif n == 7
    return "Saturday"
  end

```

return "Wrong, please enter a number between 1 and 7"

end

- 5 years ago
- [Refactor](#)

def what\_day?(n)

```

  if n == 1
    return "Sunday"
  elsif n == 2
    return "Monday"
  elsif n == 3
    return "Tuesday"
  elsif n == 4
    return "Wednesday"
  elsif n == 5
    return "Thursday"
  elsif n == 6
    return "Friday"
  elsif n == 7
    return "Saturday"
  end

```

return "Wrong, please enter a number between 1 and 7"

end

- 5 years ago
- [Refactor](#)

Python:

def whatday(n):

```

  if n == 1:
    return "Sunday"
  elif n == 2:
    return "Monday"
  elif n == 3:
    return "Tuesday"
  elif n == 4:
    return "Wednesday"
  elif n == 5:
    return "Thursday"
  elif n == 6:
    return "Friday"
  elif n == 7:
    return "Saturday"

```

return "Wrong, please enter a number between 1 and 7"

- 5 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Multiplication table](#)

Ruby:

def multiplication\_table(size)

```

  x = 1
  y = 1
  i = 1
  multiplicator = 1
  cont = 0
  r1 = []
  r2 = []
  while y <= size do
    while cont < size do
      r1.push(x)
      x = x + i
      cont = cont + 1
    end
    cont = 0
    r2.push(r1)
  end

```

```

    r1 = []
    multiplier = multiplier + 1
    x = y + 1
    y = y + 1
    i = i + 1
  end
  r2
end

```

- 4 years ago
- [Refactor](#)

```

def multiplication_table(size)
  x = 1
  y = 1
  i = 1
  multiplier = 1
  cont = 0
  r1 = []
  r2 = []
  while y <= size do
    while cont < size do
      r1.push(x)
      x = x + i
      cont = cont + 1
    end
    cont = 0
    r2.push(r1)
    r1 = []
    multiplier = multiplier + 1
    x = y + 1
    y = y + 1
    i = i + 1
  end
  r2
end

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Ruby Metaprogramming 101 - Dynamic Method Calls](#)

Ruby:

```

def dynamic_caller(obj, method)
  obj.public_send(method)
end

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Grasshopper - Function syntax debugging](#)

Ruby:

```

def main(verb, noun)
  verb + noun
end

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Smallest unused ID](#)

Ruby:

```

def next_id(arr)
  arr.sort!
  cont = 0

  while (true) do
    return cont unless arr.include? cont
    cont = cont + 1
  end
end

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Grasshopper - If/else syntax debug](#)

Ruby:

```

def check_alive(health)
  if health <= 0
    return false
  else
    return true
  end
end

```

end  
end

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Hello, Name or World!](#)

PHP:

```
function hello($name = ''): string {
    if (empty($name)) return "Hello, World!";
    return "Hello, " . ucfirst(strtolower($name)) . "!";
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

5 kyu

[Perimeter of squares in a rectangle](#)

Ruby:

```
def perimeter(n)
    4 * fibonacci(n + 1)
end

def fibonacci (numero)
    iteracoes = 0
    numero_atual = 1
    numero_anterior = 0
    total = 0

    while iteracoes < numero
        total = total + numero_atual
        temp = numero_atual
        numero_atual = numero_atual + numero_anterior
        numero_anterior = temp
        iteracoes = iteracoes + 1
    end

    total
end
```

- 7 years ago
- [Refactor](#)

JavaScript:

```
function perimeter(n) {
    let valor = fib(n);
    return 4* valor.reduce((a, b) => a + b, 0);
}

function fib(max) {
    let prev1 = 1;
    let prev2 = 0;
    let sum = 1;
    let current = 0;
    let retorno = [1];

    while (current < max) {
        sum = sum + prev2;
        prev2 = prev1;
        prev1 = sum;
        retorno.push(sum);
        current++;
    }

    return retorno;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function perimeter($n) {
    return 4 * fibonacci($n + 1);
}

function fibonacci($numero) {
    $iteracoes = 0;
    $numero_atual = 1;
    $numero_anterior = 0;
    $total = 0;

    while ($iteracoes < $numero) {
        $total = $total + $numero_atual;
        $temp = $numero_atual;
        $numero_atual = $numero_atual + $numero_anterior;
        $numero_anterior = $temp;
        $iteracoes = $iteracoes + 1;
    }
}
```

```

}
return $total;
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Exclusive "or" \(xor\) Logical Operator](#)

Go:

```

package kata

func Xor(a, b bool) bool {
    if ((a == true && b == false) || (b == true && a == false)) {
        return true
    }
    return false
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Watermelon](#)

Go:

```

package kata

func Divide(weight int) bool {
    return (weight % 2 == 0) && (weight > 2)
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[SQL Basics: Simple JOIN with COUNT](#)

SQL:

```

-- Create your SELECT statement here
select people.*, count(toys.people_id) as toy_count from people inner join toys on people.id = toys.people_id group by(people.id)

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[The falling speed of petals](#)

JavaScript:

```

function sakuraFall(v) {
    if (v <= 0) return 0;

    return 400/v;
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```

def sakura_fall(v)
    v = v.to_f
    v <= 0 ? 0 : 400 / v
end

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Beginner Series #4 Cockroach](#)

JavaScript:

```

function cockroachSpeed(s) {
    return Math.floor(s * 100000 / 3600);
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Parse float](#)

JavaScript:

```
function parseF(s) {
  if (isNaN(Number.parseFloat(s))) {
    return null;
  }

  return parseFloat(s);
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Grasshopper - Messi Goals](#)

JavaScript:

```
var laLigaGoals = 43;
var championsLeagueGoals = 10;
var copaDelReyGoals = 5;

var totalGoals = laLigaGoals + championsLeagueGoals + copaDelReyGoals;
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Grasshopper - Debug sayHello](#)

JavaScript:

```
function sayHello(name) {
  return 'Hello, ' + name;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function sayHello(string $name): string
{
  return "Hello, " . $name;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Capitalization and Mutability](#)

JavaScript:

```
function capitalizeWord(word) {
  return word[0].toUpperCase() + word.slice(1, word.length);
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Stop gninnipS My sdroW!](#)

Ruby:

```
def spinWords(string)
  string.split(" ").map{|palavra| palavra.length >= 5 ? palavra.reverse : palavra}.join(" ")
end
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function spinWords(frase){
  let palavras = frase.split(' ');
  for (let indice in palavras) {
    if (palavras[indice].length >= 5) {
      palavras[indice] = palavras[indice].split('').reverse().join('');
    }
  }
}
```

```

    }
    return palavras.join(' ');
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Are the numbers in order?](#)

PHP:

```

function in_asc_order($arr) {
    $itemAnterior = null;
    foreach ($arr as $item) {
        if ($item < $itemAnterior) return false;
        $itemAnterior = $item;
    }

    return true;
}

```

- 4 years ago
- [Refactor](#)

Ruby:

```

def is_asc_order a
    itemAnterior = -100000000
    a.each {|item|
        return false if (item < itemAnterior)
        itemAnterior = item
    }

    true
end

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

C++:

```

bool isAscOrder(std::vector<int> arr)
{
    int itemAnterior;

    for (int item : arr) {
        if (item < itemAnterior) return false;
        itemAnterior = item;
    }

    return true;
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Maximum Multiple](#)

PHP:

```

function maxMultiple($divisor, $extremo) {
    $retorno = 0;
    $numero = 1;
    while ($numero <= $extremo) {
        if ($numero % $divisor == 0) {
            $retorno = $numero;
        }
        $numero++;
    }

    return $retorno;
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Groovy:

```

class Kata {
    static maxMultiple($divisor, $extremo) {
        def $retorno = 0;
        def $numero = 1;
        while ($numero <= $extremo) {
            if ($numero % $divisor == 0) {
                $retorno = $numero;
            }
            $numero++;
        }

        return $retorno;
    }
}

```



```
}
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export function maxMultiple(divisor: number, bound: number) {
  let retorno = 0;
  let numero = 1;
  while (numero <= bound) {
    if (numero % divisor == 0) {
      retorno = numero;
    }
    numero++;
  }
  return retorno;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

C++:

```
int maxMultiple(int divisor, int bound)
{
  int retorno = 0;
  int numero = 1;
  while (numero <= bound) {
    if (numero % divisor == 0) {
      retorno = numero;
    }
    numero++;
  }
  return retorno;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Check the exam](#)

Python:

```
def check_exam(arr1,arr2):
  sum = 0
  current = 0
  for i in arr2:
    if i == "":
      next
    elif i == arr1[current]:
      sum = sum + 4
    else:
      sum = sum - 1

    current = current + 1

  if sum < 0:
    return 0

  return sum
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export function checkExam(array1: string[], array2: string[]): number {
  let sum = 0;
  let current = 0;
  let i;

  for (i in array2) {
    if (array2[i] == "") {
    }
    else if (array2[i] == array1[current]) {
      sum = sum + 4;
    }
    else {
      sum = sum - 1;
    }
    current = current + 1;
  }

  if (sum < 0) {
    return 0;
  }
}
```

```
    return sum;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Short Long Short](#)

PHP:

```
function shortLongShort(string $s1, string $s2): string
{
    $tamanho1 = strlen($s1);
    $tamanho2 = strlen($s2);

    if ($tamanho1 > $tamanho2) {
        return $s2 . $s1 . $s2;
    }
    return $s1 . $s2 . $s1;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export function shortLongShort(a:string, b:string) {
    let tamanho1 = b.length;
    let tamanho2 = a.length;

    if (tamanho1 > tamanho2) {
        return a + b + a;
    }
    return b + a + b;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Quarter of the year](#)

Python:

```
import math
def quarter_of(month):
    return math.ceil(month/3)
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Fake Binary](#)

PHP:

```
function fake_bin(string $s): string {
    $s = preg_replace("/[01234]/","0", $s);
    $s = preg_replace("/[56789]/","1", $s);

    return $s;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Thinkful - Number Drills: Pixelart planning](#)

JavaScript:

```
function isDivisible(wallLength, pixelSize) {
    if (wallLength % pixelSize == 0) {
        return true;
    } else {
        return false;
    }
}
```

- 6 years ago
- [Refactor](#)

```
function isDivisible(wallLength, pixelSize){
    return !((wallLength / pixelSize) % 1);
}
```

- 7 years ago

- [Refactor](#)
- [Discuss](#)

Groovy:

```
class Kata {
  static def isDivisible(wallLength, pixelSize) {
    if (wallLength % pixelSize == 0) {
      return true
    } else {
      return false
    }
  }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Remove B M W](#)

JavaScript:

```
function removeBMW(str){
  if (typeof str !== "string") throw new Error("This program only works for text.");

  //TO DO
  return str.replace(/[bmw]/ig, '');
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Elevator Distance](#)

JavaScript:

```
function elevatorDistance(array) {
  let total = 0;
  for (i in array) {
    if (i == array.length - 1) break;
    total += Math.abs(array[i] - array[parseInt(i)+1]);
  }
  return total;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sum of odd numbers](#)

JavaScript:

```
function rowSumOddNumbers(n) {
  if (n === 1) return 1;
  let primeiro = Math.pow(n, 2) - n;
  let soma = primeiro;

  let cont = 1;
  while (cont < n) {
    soma = soma + primeiro + 2 * cont;
    cont++;
  }

  return soma + n;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Groovy:

```
class Kata {
  static rowSumOddNumbers(n) {
    if (n == 1) return 1

    Integer primeiro = Math.pow(n, 2) - n
    Integer soma = primeiro

    Integer cont = 1
    while (cont < n) {
      soma = soma + primeiro + 2 * cont
      cont = cont + 1
    }

    return soma + n
  }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Convert a String to a Number!](#)

Ruby:

```
def string_to_number(s)
  s.to_i
end
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function stringToNumber($str) {
    return (int) $str; // do stuff
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

```
function stringToNumber($str) {
    return (int) $str;
}
```

- 5 years ago
- [Refactor](#)

C#:

```
using System;
public class Kata
{
    public static int StringToNumber(String str) {
        return Int32.Parse(str);
    }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Groovy:

```
class Kata {
    static int stringToNumber(String s) {
        s.toInteger()
    }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Small enough? - Beginner](#)

PHP:

```
function smallEnough($a, $limit){
    $t=0;

    for ($i=0; $i < count($a) + 1; $i++) {
        if ($a[$i] > $limit) {
            return false;
        }
    }

    return true;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Groovy:

```
class Kata {
    static def smallEnough(arr, limit) {
        def t=0;

        for (def i=0; i <= arr.size + 1; i++) {
            if (arr[i] > limit) {
                return false;
            }
        }
    }
}
```

```

    return true;
  }
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[How do I compare numbers?](#)

Ruby:

```

def what_is(x)
  puts x
  if x.equal?(42)
    'everything'
  elsif x > 123
    'everything everythinged'
  else
    'nothing'
  end
end
end

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Groovy:

```

class Kata {
  static whatIs(x) {
    if (x == 42) {
      return "everything"
    } else if (x > 123) {
      return 'everything squared'
    } else {
      return "nothing"
    }
  }
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Sum of positive](#)

Ruby:

```

def positive_sum(arr)
  soma = 0
  arr.each{|i| soma = soma + i if i >0}
  soma
end

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```

function positive_sum($arr) {
  $soma = 0;

  foreach ($arr as $num) {
    if ($num > 0) {
      $soma += $num;
    }
  }

  return $soma;
}

```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Groovy:

```

class Kata {
  static positiveSum(list) {
    Integer sum = 0

    for (i in list) {
      if (i > 0) {
        sum = sum + i
      }
    }

    sum
  }
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Beginner Series #2 Clock](#)

VB:

```
Public Module Kata
    Public Function Past(ByVal h As Integer, ByVal m As Integer, ByVal s As Integer) As Integer
        return h * 3600 * 1000 + m * 60 * 1000 + s * 1000
    End Function
End Module
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
<?php
function past($h, $m, $s) {
    return $h * 1000 * 3600 + $m * 60 * 1000 + $s * 1000;
}
```

- 4 years ago
- [Refactor](#)

C:

```
int past(int $h, int $m, int $s) {
    return $h * 1000 * 3600 + $m * 60 * 1000 + $s * 1000;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

PowerShell:

```
function Past([int] $h, [int] $m, [int] $s) {
    return $h * 1000 * 3600 + $m * 60 * 1000 + $s * 1000;
}
```

- 4 years ago
- [Refactor](#)

Groovy:

```
class Kata {
    static past(h, m, s) {
        h * 3600 * 1000 + m * 60 * 1000 + s * 1000
    }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Thoughtful - String Drills: Repeater](#)

Python:

```
def repeater(string, n):
    retorno=""
    while n > 0:
        retorno = retorno + string
        n = n-1
    return retorno
```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function solution($s, $n) {
    return str_repeat($s, $n);
}
```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

Groovy:

```
class Kata {
    static def repeater(string, n) {
```

```

    def ret = ""
    while (n > 0) {
      ret = ret + string
      n = n-1
    }
    return ret
  }
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Area or Perimeter](#)

C:

```

int area_or_perimeter(int l , int w) {
  if (l == w) {
    return l * w;
  }
  return (l + w) * 2;
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Groovy:

```

class Solution {
  static areaOrPerimeter(int l, int w) {
    def result
    if (l == w) {
      result = l * w
    } else {
      result = (l + w) * 2
    }

    result
  }
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Opposites Attract](#)

PHP:

```

function lovefunc($flower1, $flower2) {
  $flower1 % 2 == 0 ? $even1 = true: $even1 = false;
  $flower2 % 2 == 0 ? $even2 = true: $even2 = false;

  return $even1 && !$even2 || $even2 && !$even1;
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Groovy:

```

class Kata {
  static def lovefunc(flower1, flower2) {
    Boolean even1
    Boolean even2

    if (flower1 % 2 == 0 ) {
      even1 = true
    } else {
      even1 = false
    }

    if (flower2 % 2 == 0 ) {
      even2 = true
    } else {
      even2 = false
    }

    return even1 && !even2 || even2 && !even1;
  }
}

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Summing a number's digits](#)

Groovy:

```
class Kata{
  static int sumDigits(number) {
    Integer soma = 0
    number = (String) number
    def numero = ""

    number.each {
      try {
        numero = it.toInteger()
        soma = soma + numero
      } catch (e) {

      }
    }

    soma
  }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[get ascii value of character](#)

Ruby:

```
def getASCII(c)
  c.codepoints[0]
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Breaking chocolate problem](#)

PHP:

```
function breakChocolate ($n, $m) {
  return ($n * $m) - 1;
};
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[esreveR](#)

PHP:

```
function reverse(array $a): array {
  $return = [];

  foreach($a as $i) {
    array_unshift($return, $i);
  }

  return $return;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

5 kyu

[RGB To Hex Conversion](#)

JavaScript:

```
function rgb(r, g, b){
  if (r > 255) r = 255;
  if (g > 255) g = 255;
  if (b > 255) b = 255;
  if (r < 0) r = 0;
  if (g < 0) g = 0;
  if (b < 0) b = 0;

  let red = r.toString(16);
  let green = g.toString(16);
  let blue = b.toString(16);

  if (red.length ==1 ) red = "0" + red;
  if (green.length ==1 ) green = "0" + green;
  if (blue.length ==1 ) blue = "0" + blue;

  return red.toUpperCase() + green.toUpperCase() + blue.toUpperCase();
}
```

- 7 years ago
- [Refactor](#)



- [Discuss](#)

PHP:

```
function rgb($r,$g,$b){
    if ($r > 255) $r = 255;
    if ($g > 255) $g = 255;
    if ($b > 255) $b = 255;

    if ($r < 0) $r = 0;
    if ($g < 0) $g = 0;
    if ($b < 0) $b = 0;

    $r = dechex($r);
    $g = dechex($g);
    $b = dechex($b);

    if (strlen($r) == 1) $r = '0' . $r;
    if (strlen($g) == 1) $g = '0' . $g;
    if (strlen($b) == 1) $b = '0' . $b;

    return strtoupper($r . $g . $b);
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[L1: Bartender, drinks!](#)

JavaScript:

```
function getDrinkByProfession(param){
    param = param.toLowerCase();

    if (param == "jabroni") return "Patron Tequila"
    if (param == "school counselor") return "Anything with Alcohol"
    if (param == "programmer") return "Hipster Craft Beer"
    if (param == "bike gang member") return "Moonshine"
    if (param == "politician") return "Your tax dollars"
    if (param == "rapper") return "Cristal"
    return "Beer";
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export function getDrinkByProfession(param:string){
    param = param.toLowerCase();

    if (param == "jabroni") return "Patron Tequila"
    if (param == "school counselor") return "Anything with Alcohol"
    if (param == "programmer") return "Hipster Craft Beer"
    if (param == "bike gang member") return "Moonshine"
    if (param == "politician") return "Your tax dollars"
    if (param == "rapper") return "Cristal"
    return "Beer";
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function get_drink_by_profession($param){
    $param = strtolower($param);

    if ($param == "jabroni") return "Patron Tequila";
    if ($param == "school counselor") return "Anything with Alcohol";
    if ($param == "programmer") return "Hipster Craft Beer";
    if ($param == "bike gang member") return "Moonshine" ;
    if ($param == "politician") return "Your tax dollars" ;
    if ($param == "rapper") return "Cristal" ;
    return "Beer";
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Number toString](#)

Ruby:

```
a = 123.to_s
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

var a = "123";

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Unique string characters](#)

Ruby:

```
def solve(a,b)
  c = b + a
  included = ""

  a.each_char { |char|
    if not b.include? char
      included += char
      puts "included: " + included
    end
  }

  b.each_char { |char|
    if not a.include? char
      included += char
      puts "included: " + included
    end
  }

  included
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[ATM](#)

Ruby:

```
def solve(n)
  if n < 10 or n % 10 != 0
    return -1
  end

  total = 0
  puts n
  values = [500, 200, 100, 50, 20, 10]

  values.each { |value|
    if n == 0 or n < value
      next
    end

    while n > 0
      if (n - value < 0)
        break;
      end

      n = n - value
      total = total + 1
    end
  }

  total
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Find Duplicates](#)

Ruby:

```
def duplicates(a)
  retorno = []

  a.each { |e|
    if a.count(e) > 1 and retorno.count(e) == 0
      retorno.push(e)
    end
  }

  retorno
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Find the Difference in Age between Oldest and Youngest Family Members](#)

Ruby:

```
def difference_in_ages(ages)
  minor = 1000000000000
  major = 0

  ages.each {|age|
    if (age > major)
      major = age
    end

    if (age < minor)
      minor = age
    end
  }

  [minor, major, major - minor]
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function differenceInAges($ages) {
  $minor = 1000000000000;
  $major = 0;

  foreach ($ages as $age) {
    if ($age > $major) {
      $major = $age;
    }

    if ($age < $minor) {
      $minor = $age;
    }
  }

  return [$minor, $major, $major - $minor];
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function differenceInAges($ages) {
  $minor = 1000000000000;
  $major = 0;

  for (var i in $ages) {
    if ($ages[i] > $major) {
      $major = $ages[i];
    }

    if ($ages[i] < $minor) {
      $minor = $ages[i];
    }
  }

  return [$minor, $major, $major - $minor];
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Merge two sorted arrays into one](#)

JavaScript:

```
function mergeArrays(arr1, arr2) {
  return [... new Set(arr1.concat(arr2).sort((a,b) => a-b))];
}
```

- 4 years ago
- [Refactor](#)

```
function mergeArrays(arr1, arr2) {
  return Array.from(new Set(arr1.concat(arr2).sort((a, b) => a - b)));
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

```
function mergeArrays(arr1, arr2) {
  return Array.from(new Set(arr1.concat(arr2).sort((a, b) => a - b)));
}
```

- 7 years ago

- [Refactor](#)

8 kyu

[Removing Elements](#)

Ruby:

```
def remove_every_other(arr)
  return Array.new if arr.empty?
  counter = 0
  ret = []
  arr.each {|i|
    counter = counter + 1
    if (counter % 2 == 1)
      ret.push(i)
    end
  }
  ret
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function removeEveryOther(arr){
  let ret = [];
  for (var i in arr) {
    if (i % 2 == 0) {
      ret.push(arr[i]);
    }
  }
  return ret;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[You Can't Code Under Pressure #1](#)

Ruby:

```
def double_integer(i)
  i * 2
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

C:

```
#include <stdint.h>

int32_t double_integer(int32_t i){
  return i*2;
}
```

- 4 years ago
- [Refactor](#)

```
#include <stdint.h>

int32_t double_integer(i) {
  return i * 2;
}
```

- 4 years ago
- [Refactor](#)

```
#include <stdint.h>

int double_integer(i){
  return i*2;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

CoffeeScript:

```
doubleInteger = (i) ->
  # Double the integer, and return it!
  return i*2
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

```
doubleInteger = (i) ->
  return i *2
```

- 4 years ago
- [Refactor](#)

Python:

```
def double_integer(i):
  return i * 2
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function doubleInteger($i)
{
  return $i*2;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

```
function doubleInteger($i)
{
  return $i*2;
}
```

- 4 years ago
- [Refactor](#)

Java:

```
class Java {
  public static int doubleInteger(int i) {
    // Double the integer and return it!
    return i*2;
  }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

```
class Java {
  public static int doubleInteger(int i) {
    // Double the integer and return it!
    return i * 2;
  }
}
```

- 4 years ago
- [Refactor](#)

C++:

```
#include <cstdint>

int32_t double_integer(int32_t n)
{
  return n*2;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

C#:

```
using System;

public static class Kata
{
  public static int DoubleInteger(int n)
  {
    return n*2;
  }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Elixir:

```
defmodule SimpleMath do
  def double_integer(x) do
    x * 2
  end
end
```

end  
end

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Do I get a bonus?](#)

Ruby:

```
def bonus_time(salary, bonus)
  if bonus then
    return "$" + (salary * 10).to_s
  end

  return "$" + salary.to_s
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function bonusTime($salary, $bonus) {
    return $bonus ? "$" . ($salary * 10) : "$" . $salary;
}
```

- 4 years ago
- [Refactor](#)

```
function bonusTime($salary, $bonus) {
    if ($bonus) {
        return "$" . ($salary * 10);
    }
    return "$" . $salary;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function bonusTime(salary, bonus) {
    if (bonus) {
        return "£" + (salary * 10)
    }

    return "£" + salary
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

C#:

```
public static class Kata
{
    public static string bonus_time(int salary, bool bonus)
    {
        if (bonus) {
            return "$" + (salary * 10);
        }
        return "$" + salary;
    }
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Simple beads count](#)

Ruby:

```
def count_red_beads n
  t = n * 2 - 2
  return 0 if t < 2
  t
end
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function count_red_beads(int $n): int {
  return $n <= 0 ? 0 : ($n-1) * 2;
}
```

- 4 years ago
- [Refactor](#)

```
function count_red_beads(int $n): int {
  if ($n == 0) return 0;

  return ($n-1) * 2;
}
```

- 4 years ago
- [Refactor](#)

```
function count_red_beads(int $n): int {
  $t = ($n * 2) - 2;

  if ($n < 2) {
    return 0;
  }

  return $t;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Is it even?](#)

Ruby:

```
def test_even(n)
  n = n.round
  n.to_i.even?
end
```

- 4 years ago
- [Refactor](#)

6 kyu

[Counting Duplicates](#)

Ruby:

```
def duplicate_count(text)
  text.downcase!
  duplicados = []
  proxima_posicao = 1
  text.split("").each do |i|
    if (duplicados.include? i) == false and (text.slice(proxima_posicao, text.length)).include? i
      duplicados.push(i)
    end
    proxima_posicao = proxima_posicao + 1
  end

  duplicados.count
end
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Format a string of names like 'Bart, Lisa & Maggie'.](#)

Ruby:

```
def list names
  names = names.map{|hash_name| hash_name[:name] + ", "}.join("")
  names = names[0..names.length - 3]
  total_virgulas = names.scan(/,/).length
  if total_virgulas >= 1 then
    posicao_ultima_virgula = names.rindex(",")
    names[posicao_ultima_virgula] = " &"
  end

  names
end
```

- 7 years ago
- [Refactor](#)

JavaScript:

```
function list(names){
  if (names.length == 0) {
    return '';
  }

  let names_string = "";
  for (var obj of names) {
    names_string += obj.name + ", ";
  }
}
```

```

}

total_virgulas = names_string.match(/,/g).length;
names_string = names_string.substr(0, names_string.length - 2);

if (total_virgulas > 1) {
  posicao_ultima_virgula = names_string.lastIndexOf(",");
  names_string = names_string.substr(0, posicao_ultima_virgula) + " &" + names_string.substr(posicao_ultima_virgula + 1, names_string.length)
}

return names_string;
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Grasshopper - Array Mean](#)

Ruby:

```

def find_average(nums)
  return 0 if nums.empty?

  sum = 0.0
  cont = 0.0
  nums.each {|num|
    sum = sum + num
    cont = cont + 1
  }
  (sum / cont).to_f
end

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Take the Derivative](#)

Ruby:

```

def derive(coefficient, exponent)
  val = coefficient * exponent
  exponent = exponent - 1
  val.to_s + "x^" + exponent.to_s
end

```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```

function derive(coefficient,exponent) {
  let val = coefficient * exponent;
  exponent = exponent - 1;
  return val + "x^" + exponent;
}

```

- 4 years ago
- [Refactor](#)

8 kyu

[Get Planet Name By ID](#)

JavaScript:

```

function getPlanetName(id){
  var name;
  switch(id){
    case 1:
      name = 'Mercury';
      break;
    case 2:
      name = 'Venus';
      break;
    case 3:
      name = 'Earth';
      break;
    case 4:
      name = 'Mars';
      break;
    case 5:
      name = 'Jupiter';
      break;
    case 6:
      name = 'Saturn';
      break;
    case 7:
      name = 'Uranus';
      break;
    case 8:
      name = 'Neptune';
      break;
  }
}

```



```
return name;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def get_planet_name(id)
  # This doesn't work; Fix it!
  name = ''
  case id
  when 1
    name = "Mercury"
  when 2
    name = "Venus"
  when 3
    name = "Earth"
  when 4
    name = "Mars"
  when 5
    name = "Jupiter"
  when 6
    name = "Saturn"
  when 7
    name = "Uranus"
  when 8
    name = "Neptune"
  end

  return name
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Remove duplicate words](#)

PHP:

```
function removeDuplicateWords($s) {
    $words = explode(' ', $s);

    $return = [];
    foreach ($words as $word) {
        if (! in_array($word, $return)) {
            $return[] = $word;
        }
    }

    return implode($return, ' ');
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Is the string uppercase?](#)

PHP:

```
function is_uppercase($str) {
    return $str === strtoupper($str);
}
```

- 4 years ago
- [Refactor](#)

Retired

[N-th Power](#)

JavaScript:

```
function index(array, n){
    if (array[n] == undefined) {
        return -1;
    }
    return Math.pow(array[n], n);
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[No zeros for heros](#)

Ruby:

```
def no_boring_zeros(num)
  num = num.to_s
  num = num.gsub(/0+$/) {''}
  num = num.to_i
  num
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Array plus array](#)

Ruby:

```
def array_plus_array(arr1, arr2)
  arr1.sum + arr2.sum
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function arrayPlusArray(arr1, arr2) {
  sum = 0
  for (let arr of arr1) {
    sum = sum + arr
  }
  for (let arr of arr2) {
    sum = sum + arr
  }
  return sum
}
```

- 4 years ago
- [Refactor](#)

C:

```
#include <stddef.h>

long arr_plus_arr(const int *a, const int *b, size_t na, size_t nb)
{
  long sum = 0;
  long i=0;
  for (i=0;i<na;i++) {
    sum = sum + a[i];
  }
  for (i=0;i<nb;i++) {
    sum = sum + b[i];
  }
  return sum;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Beginner Series #1 School Paperwork](#)

JavaScript:

```
function paperwork(n, m) {
  if (n <= 0 || m <= 0) {
    return 0;
  }
  return n * m;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def paperwork(n, m)
  if n <= 0 || m <= 0 then
    return 0
  end
  n * m
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Squash the bugs](#)

Ruby:

```
def find_longest(string)
  spl = string.split(" ")
  longest = 0
  i=0

  while (i < spl.size) do
    tamanho = spl[i].size
    if (tamanho > longest) then
      longest = tamanho
    end
    i = i + 1
  end

  return longest
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Will you make it?](#)

Ruby:

```
def zero_fuel(distance, mpg, fuel_left)
  mpg * fuel_left >= distance
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

Java:

```
public class Kata {

  public static boolean zeroFuel(double distanceToPump, double mpg, double fuelLeft) {
    return mpg * fuelLeft >= distanceToPump;
  }

}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

C:

```
#include <stdbool.h>

bool zero_fuel(double distance_to_pump, double mpg, double fuel_left)
{
  return mpg * fuel_left >= distance_to_pump;
}
```

- 4 years ago
- [Refactor](#)

```
#include <stdbool.h>

bool zero_fuel(double distance_to_pump, double mpg, double fuel_left)
{
  return mpg * fuel_left >= distance_to_pump;
}
```

- 4 years ago
- [Refactor](#)

C#:

```
using System;

public static class Kata
{
  public static bool ZeroFuel(uint distanceToPump, uint mpg, uint fuelLeft)
  {
    return mpg * fuelLeft >= distanceToPump;
  }
}
```

- 4 years ago
- [Refactor](#)

JavaScript:

```
const zeroFuel = (distanceToPump, mpg, fuelLeft) => {
  return mpg * fuelLeft >= distanceToPump;
};
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Well of Ideas - Easy Version](#)

Ruby:

```
def well (x)
  contador = 0

  x.each { |xx|
    if xx == "good" then
      contador = contador + 1
    end
  }

  if (contador > 0 && contador <= 2)
    return "Publish!"
  elsif (contador >= 2)
    return "I smell a series!"
  else
    return "Fail!"
  end
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[A + B](#)

Java:

```
public class FirstClass {
  public static long sum (byte a, byte b) {
    long c = a + b;
    return c;
  }
}
```

- 5 years ago
- [Refactor](#)

C#:

```
public class FirstClass
{
  public static long sum (byte a, byte b)
  {
    long c = a + b;
    return c;
  }
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[The Wide-Mouthed frog!](#)

Ruby:

```
def mouth_size(animal)
  animal.downcase!
  animal == "alligator" ? "small" : "wide"
end
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[What is between?](#)

JavaScript:

```
function between(a, b) {
  retorno = []

  while (a <= b) {
    retorno.push(a)
    a++
  }

  return retorno
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Generate range of integers](#)

JavaScript:

```
function generateRange(min, max, step){
  retorno = []
  atual = min

  while (atual <= max) {
    retorno.push(atual)
    atual = atual + step
  }

  return retorno;
}
```

- 4 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Find Multiples of a Number](#)

Python:

```
def find_multiples(integer, limit):
    retorno = []
    inicio = integer
    while (integer <= limit):
        if (integer / inicio == integer // inicio):
            retorno.append(integer)
            integer=integer+1
    return retorno
```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def find_multiples(integer, limit)
  a = integer
  r= []
  while a <= limit
    r.push a
    a = a + integer
  end
  return r
end
```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Beginner - Reduce but Grow](#)

PHP:

```
function grow($a) {
    $resultado = 1;

    foreach ($a as $item) {
        $resultado = $resultado * $item;
    }

    return $resultado;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Is this my tail?](#)

JavaScript:

```
function correctTail(body, tail) {
  sub = body.substr(body.length-1, 1);

  if (sub == tail) {
    return true
  }
  else {
    return false;
  }
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function equivalent($body, $char) {
    $newChar = substr($body, -1, 1);

    if ($char == $newChar) {
        return true;
    } else {
        return false;
    }
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

```
function equivalent($body, $char) {
    return $char === substr($body, -1, 1);
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Isograms](#)

PHP:

```
function isIsogram($string) {
    for ($i=0; $i<strlen($string); $i++) {
        $existentes[] = $string[$i];

        for ($j = strlen($string); $j > $i; $j--) {
            if (strtolower($string[$i]) == strtolower($string[$j])) {
                return false;
            }
        }
    }

    return true;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Total amount of points](#)

PHP:

```
function points(array $games): int {
    $total = 0;
    for ($i = 0; $i < count($games); $i++) {
        $ponto1 = substr($games[$i], 0, 1);
        $ponto2 = substr($games[$i], 2, 1);
        //unset($games[count($games) - 1]);

        if ($ponto1 > $ponto2) {
            $total +=3;
        } elseif ($ponto1 == $ponto2) {
            $total +=1;
        }
    }

    print($total);
    return $total;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Bin to Decimal](#)

PHP:

```
function binToDec($bin) {
    return bindec($bin);
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Grasshopper - Messi goals function](#)

C++:

```
int goals (int laligaGoals, int copaDelReyGoals, int championsLeagueGoals) {
    return laligaGoals + copaDelReyGoals + championsLeagueGoals;
}
```

- 5 years ago
- [Refactor](#)

Ruby:

```
def goals (laligaGoals, copaDelReyGoals, championsLeagueGoals)
  laligaGoals + copaDelReyGoals + championsLeagueGoals
end
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

CoffeeScript:

```
goals = (laligaGoals, copaDelReyGoals, championsLeagueGoals) -> laligaGoals + copaDelReyGoals + championsLeagueGoals
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Grasshopper - Messi goals function](#)

JavaScript:

```
function goals (laligaGoals, copaDelReyGoals, championsLeagueGoals) {
  return laligaGoals + copaDelReyGoals + championsLeagueGoals;
}
```

- 5 years ago
- [Refactor](#)

PHP:

```
function goals (int $laligaGoals, int $copaDelReyGoals, int $championsLeagueGoals) : int {
  return $laligaGoals + $copaDelReyGoals + $championsLeagueGoals;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Go:

```
package kata
```

```
func Goals(laligaGoals, copaDelReyGoals, championsLeagueGoals int) int {
  return laligaGoals + copaDelReyGoals + championsLeagueGoals
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export function goals (laligaGoals:number, copaDelReyGoals:number, championsLeagueGoals:number) {
  return laligaGoals + copaDelReyGoals + championsLeagueGoals
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

C++:

```
int goals (int laligaGoals, int copaDelReyGoals, int championsLeagueGoals) {
  return laligaGoals + copaDelReyGoals + championsLeagueGoals;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[L1: Set Alarm](#)

Ruby:

```
def set_alarm(employed, vacation)
  if (employed == true && vacation == false)
    return true;
  end
  false;
end
```

- 5 years ago
- [Refactor](#)

8 kyu

[Count the Monkeys!](#)

PHP:

```
function monkeyCount($n) {
    $r = [];
    for ($a = 1; $a <= $n; $a++) {
        $r[] = $a;
    }

    return $r;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[How many arguments](#)

PHP:

```
function args_count() {
    return count(func_get_args());
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Convert a Number to a String!](#)

JavaScript:

```
function numberToString(num) {
    return String(num).valueOf();
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function numberToString($num)
{
    return (string) $num;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def numberToString(num)
    num.to_s
end
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Opposite number](#)

JavaScript:

```
function opposite(number) {
    return number * -1
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

```
function opposite(number) {
    return number < 0 ? Math.abs(number): -Math.abs(number);
}
```

- 7 years ago
- [Refactor](#)

```
function opposite(number) {
    return number * (-1);
}
```

- 7 years ago
- [Refactor](#)



Python:

```
def opposite(number):
    return number * -1
```

- 5 years ago
- [Refactor](#)

```
def opposite(number):
    return number * -1
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

```
def opposite(number):
    return number*-1
```

- 5 years ago
- [Refactor](#)

C:

```
float opposite(float num) {
    return num * -1;
}
```

- 5 years ago
- [Refactor](#)

```
float opposite(float num) {
    return num * -1;
}
```

- 5 years ago
- [Refactor](#)

C++:

```
int opposite(int number)
{
    return number * -1;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

OCaml:

```
let opposite (number : int) : int =
    number * -1
```

- 5 years ago
- [Refactor](#)

Go:

```
package kata
```

```
func Opposite(value int) int {
    return value * -1
}
```

- 5 years ago
- [Refactor](#)

```
package kata
```

```
func Opposite(value int) int {
    return value * -1
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

C#:

```
using System;
```

```
public class Kata
{
    public static int Opposite(int number)
    {
        return number * -1;
    }
}
```

- 5 years ago
- [Refactor](#)

using System;

```
public class Kata
{
    public static int Opposite(int number)
    {
        return number * -1;
    }
}
```

- 5 years ago
- [Refactor](#)

Groovy:

```
class Solution {
    static opposite(number) {
        return number * -1;
    }
}
```

- 5 years ago
- [Refactor](#)

Java:

```
public class Kata
{
    public static int opposite(int number)
    {
        return -1 * number;
    }
}
```

- 5 years ago
- [Refactor](#)

```
public class Kata
{
    public static int opposite(int number)
    {
        return number * -1;
    }
}
```

- 5 years ago
- [Refactor](#)

Elixir:

```
defmodule Opposite do
  def opposite(number) do
    number * -1
  end
end
```

- 5 years ago
- [Refactor](#)

Crystal:

```
def opposite(n)
  return n * -1
end
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

```
def opposite(n)
  n * -1
end
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Julia:

```
module Solution
  export opposite
  function opposite(number)
    return number * -1
  end
end
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Kotlin:

```
fun opposite(number: Int): Int {
    return number * -1
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Lua:

```
local kata = {}

function kata.opposite(number)
    return number * -1
end

return kata
```

- 5 years ago
- [Refactor](#)

Nim:

```
proc opposite*(number: int) : int =
    return number * -1
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

```
proc opposite*(number: int) : int =
    return number * -1
```

- 5 years ago
- [Refactor](#)

Rust:

```
fn opposite(number: i32) -> i32 {
    return number * -1
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Swift:

```
func opposite(number: Double) -> Double {
    return number * -1
}
```

- 5 years ago
- [Refactor](#)

```
func opposite(number: Double) -> Double {
    return number * -1
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export class Kata {
    static opposite(n: number) {
        return n * -1;
    }
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function opposite($n) {
    return $n * -1;
}
```

- 5 years ago
- [Refactor](#)

Ruby:

```
def opposite n
    n * -1
end
```

- 5 years ago

- [Refactor](#)
- [Discuss](#)

7 kyu

[Difference between biggest 2 numbers](#)

Ruby:

```
def diff_big_2(arr)
  b1 = -10000
  b2 = -10000

  arr.each do |n|
    maiorTodos = false

    if n > b1 then
      b2 = b1
      b1 = n
      maiorTodos = true
    end

    if n > b2 and maiorTodos == false then
      b2 = n
    end
  end

  return b1 - b2
end
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Exclamation marks series #4: Remove all exclamation marks from sentence but ensure a exclamation mark at the end of string](#)

PHP:

```
function remove(string $s): string {
  $s = str_replace("!", "", $s);
  $s = $s . "!";
  return $s;
}
```

- 5 years ago
- [Refactor](#)

```
function remove(string $s): string {
  $s = str_replace('!', '', $s);
  return $s . "!";
}
```

- 5 years ago
- [Refactor](#)

```
function remove(string $s): string {
  $s = str_replace("!", "", $s);
  return $s . "!";
}
```

- 5 years ago
- [Refactor](#)

```
function remove(string $s): string {
  $r = str_replace("!", "", $s);
  return $r . "!";
}
```

- 6 years ago
- [Refactor](#)

```
function remove(string $s): string {
  return str_replace("!", "", $s) . "!";
}
```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function remove(s){
  let r = s.replace(/!/g, "");
  return r + "!";
}
```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Abbreviate a Two Word Name](#)

Java:

```
public class AbbreviateTwoWords {

    public static String abbrevName(String name) {
        String nome = name.substring(0,1);
        int indiceEspaco = Math.abs(name.indexOf(" "));
        String sobrenome = name.substring(indiceEspaco + 1, indiceEspaco + 2);

        return nome.toUpperCase() + "." + sobrenome.toUpperCase();
    }
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Number of Divisions](#)

JavaScript:

```
const divisions = (n, divisor) => {
    let cont = 0;

    console.log(n);
    while (n > 1) {
        n = n / divisor;
        cont++;
    }

    return cont - 1;
};
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

```
const divisions = (current_number, divisor) => {
    let total = 0;

    while (divisor <= current_number) {
        total++;
        current_number = Math.floor(current_number / divisor);
    }

    return total;
};
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

TypeScript:

```
export function divisions(n, divisor) {
    let cont = 0;

    while (n > 1) {
        n = n / divisor;
        cont++;
    }

    return cont - 1;
};
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Remove anchor from URL](#)

JavaScript:

```
function removeUrlAnchor(url){
    url_dividida = url.split("#");
    return url_dividida[0];
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

```
function removeUrlAnchor(url){
    const posicaoSustenido = url.indexOf("#");

    if (posicaoSustenido > -1) {
        return url.substr(0, posicaoSustenido);
    }

    return url;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function replaceAll($string) {
    if (strpos($string, "#") == false) {
        return $string;
    }
    return substr($string, 0, strpos($string, "#"));
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

```
function replaceAll($string) {
    $posicaoAncora = strpos($string, "#");

    if ($posicaoAncora == false) {
        return $string;
    }

    return substr($string, 0, $posicaoAncora);
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[If you can't sleep, just count sheep!!](#)

JavaScript:

```
var countSheep = function (num){
    let retorno = '';
    let n=1;

    while(n <= num) {
        retorno = retorno + n + " sheep...";
        n++;
    }
    return retorno;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Sum Mixed Array](#)

PHP:

```
function sum_mix($a) {
    $retorno = 0;
    foreach ($a as $i) {
        $t = gettype($i);
        if ($t == "integer" || $t == "string") {
            $retorno += $i;
        }
    }

    return $retorno;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Reversed Words](#)

JavaScript:

```
function reverseWords(str){
    let array_retorno = []
    for (let word of str.split(" ")) {
        array_retorno.unshift(word);
    }

    return array_retorno.join(" ");
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[String repeat](#)

JavaScript:

```
function repeatStr (n, s) {
    r = "";
    for (i=0; i < n ; i++) {
        r = r + s;
    }
}
```

```

    }
    return r;
}

```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

```

function repeatStr (n, s) {
  return s.repeat(n);
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```

function repeatStr($n, $str)
{
  return str_repeat($str, $n);
}

```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Convert boolean values to strings 'Yes' or 'No'.](#)

Java:

```

class YesOrNo
{
  public static String boolToWord(boolean b)
  {
    return b ? "Yes" : "No";
  }
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```

function boolToWord( bool ){
  return bool ? "Yes" : "No";
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```

def bool_to_word(bool)
  if bool then
    return "Yes"
  end
  return "No"
end

```

- 6 years ago
- [Refactor](#)

```

def bool_to_word bool
  bool ? "Yes" : "No"
end

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```

function boolToWord($bool){
  return $bool ? "Yes" : "No";
}

```

- 5 years ago
- [Refactor](#)

```

function boolToWord($bool){
  if ($bool == "Yes") {
    return "Yes";
  }
  else {
    return "No";
  }
}

```

- 6 years ago

- [Refactor](#)
- [Discuss](#)

C#:

```
using System;
using System.Linq;

public static class Kata
{
    public static string boolToWord(bool word)
    {
        if (word == true)
            return "Yes";
        return "No";
    }
}
```

- 6 years ago
- [Refactor](#)

```
using System;
using System.Linq;

public static class Kata
{
    public static string boolToWord(bool word)
    {
        if (word == true) {
            return "Yes";
        }
        return "No";
    }
}
```

- 6 years ago
- [Refactor](#)

```
using System;
using System.Linq;

public static class Kata
{
    public static string boolToWord(bool word)
    {
        if (word == true) {
            return "Yes";
        }
        return "No";
    }
}
```

- 6 years ago
- [Refactor](#)

8 kyu  
[DNA to RNA Conversion](#)

Ruby:

```
def DNAtorNA(dna)
  dna.gsub('T', 'U')
end
```

- 5 years ago
- [Refactor](#)

```
def DNAtorNA(dna)
  r = dna.gsub!('T', 'U')
  dna
end
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function DNAtorNA(dna) {
  dna = dna.replace(/T/gi, "U");
  return dna;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu  
[Do you speak "English"?](#)

JavaScript:

```
function spEng(sentence){
  sentence = sentence.toLowerCase(sentence);
  if (sentence.match(/english/)) {
```



```

    return true;
  } else {
    return false;
  }
}

```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[You only need one - Beginner](#)

JavaScript:

```

function check(a,x){
  for (let i of a) {
    if (i == x) return true;
  }

  return false;
};

```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

Python:

```

def check(seq, elem):
    for i in seq:
        if i == elem:
            return True;
    return False

```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```

function solution($a, $x) {
    foreach ($a as $i) {
        if ($i === $x) {
            return true;
        }
    }

    return false;
}

```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

```

function solution($a, $x) {
    return in_array($x, $a);
}

```

- 6 years ago
- [Refactor](#)

PHP:

```

function solution($a, $x) {
    foreach ($a as $i) {
        if ($i === $x) {
            return true;
        }
    }

    return false;
}

```

- 6 years ago
- [Refactor](#)

Ruby:

```

def check(arr,element)
  arr.include? element
end

```

- 6 years ago
- [Refactor](#)

Retired

[Can we divide it?](#)

JavaScript:

```

function isDivideBy(number, a, b) {
  if (number % a == 0) {

```

```

    if (number % b == 0) {
      return true;
    }
  }
  return false;
}

```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

Python:

```

def is_divide_by(number, a, b):
    if number % a == 0 and number % b == 0:
        return True;
    return False;

```

- 6 years ago
- [Refactor](#)

Ruby:

```

def is_divide_by(number, a, b)
    number % a == 0 and number % b == 0
end

```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Lua is easy: Lesson 1 - The basics](#)

Lua:

```

kata = {}
function kata.firstLua(a,b,c)
    if (b >= c) then
        return a .. " " .. a*b .. " Lua"
    end
    return a .. " " .. a*b .. " Codewars"
end
return kata

```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Convert string to camel case](#)

Ruby:

```

def to_camel_case(str)
    original = str.clone
    str = str.split("-").map do |i|
        i[0].upcase + i[1, i.length]
    end.join

    str = str.split("-").map do |i|
        i[0].upcase + i[1, i.length]
    end.join

    str = str[0].downcase + str[1, str.length] if original.match(/[a-z]/) && str != ""
    str
end

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```

function toCamelCase(str){
    if (str.trim() == "") return "";
    let partes_string = str.split(/[-_]/);
    let resposta = "";
    for (let parte of partes_string) {
        resposta += parte[0].toUpperCase() + parte.substr(1);
    }

    if (str[0].toLowerCase() == str[0]) {
        resposta = resposta[0].toLowerCase() + resposta.substr(1);
    }

    return resposta;
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Count Odd Numbers below n](#)

JavaScript:

```
function oddCount(n){
  return Math.ceil((n-1)/2);
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Sum The Strings](#)

JavaScript:

```
function sumStr(a,b) {
  return String(Number(a) + Number(b))
}
```

- 6 years ago
- [Refactor](#)

```
function sumStr(a,b) {
  if (a.trim() == "") a = "0";
  if (b.trim() == "") b = "0";
  return String(parseInt(a) + parseInt(b));
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[get character from ASCII Value](#)

Ruby:

```
def getChar(c)
  c.chr
end
```

- 6 years ago
- [Refactor](#)

JavaScript:

```
function getChar(c){
  let a = String.fromCharCode(c);
  return a;
}
```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Beginner - Lost Without a Map](#)

JavaScript:

```
function maps(x){
  let retorno = [];
  for (var i in x) {
    retorno[i] = x[i]*2;
  }
  return retorno;
}
```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Remove duplication](#)

JavaScript:

```
function removeDuplication(arr){
  arr = arr.sort();
  let retorno = [];
  let anterior = null;
```

```

let posicaoExistente = null;
for (let i of arr) {
  posicaoExistente = arr.indexOf(i);
  if (i !== anterior && i !== undefined) {
    retorno.push(i);
  } else {
    posicaoExistente = retorno.indexOf(i);
    if (posicaoExistente > -1) {
      retorno.splice(posicaoExistente, 1);
    }
  }
  anterior = i;
}
return retorno;
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sum of integers in string](#)

JavaScript:

```

function sumOfIntegersInString(s){
  let arrayNumeros = s.split(/^[^0-9]+/);
  let total = 0;
  return arrayNumeros.reduce(function (total, atual) {
    atual = parseInt(atual);
    if (!isNaN(atual)) {
      return total = parseInt(total) + atual;
    }
    return total;
  }, 0);
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Potenciation](#)

JavaScript:

```

function power(x,y){
  return x**y
}

```

- 6 years ago
- [Refactor](#)

```

function power(x,y){
  //SHOW ME WHAT YOU GOT!
  return x ** y
}

```

- 6 years ago
- [Refactor](#)

```

function power(x,y){
  if (x == 1 || y == 0) return 1;

  let cont = 1;
  let retorno=x*x;

  while (cont < y - 1) {
    retorno = retorno * x;
    cont++;
  }

  return retorno;
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Correct the time-string](#)

Ruby:

```

def validar_formato(partes_tempo)
  i = 0
  return false unless partes_tempo.length == 3
  while(i < 3) do
    return false if partes_tempo[i].to_s.length != 2 or partes_tempo[i].match(/[0-9]{2}/) == nil
    i = i+1
  end
  true
end

def time_correct(t)
  return t if t.nil? or t=="
  partes_tempo = t.split(":")

```

```

return nil unless validar_formato(partes_tempo)
segundos = partes_tempo[2].to_i % 60
acrescimo_minutos = partes_tempo[2].to_i / 60
minutos = (partes_tempo[1].to_i % 60) + acrescimo_minutos
acrescimo_horas = partes_tempo[1].to_i / 60
horas = (partes_tempo[0].to_i % 24) + acrescimo_horas

horas.to_s.rjust(2, '0') + ":" + minutos.to_s.rjust(2, '0') + ":" + segundos.to_s.rjust(2, '0')
end

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Is it a vowel on this position?](#)

JavaScript:

```

function checkVowel(string, position) {
  str = string.slice(position, 1).toLowerCase();
  console.log(str);
  return str == "a" || str == "e" || str == "i" || str == "o" || str == "u";
};

```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Reversed sequence](#)

JavaScript:

```

const reverseSeq = n => {
  let retorno = []
  while (n >= 1) {
    retorno.push(n);
    n--;
  }
  return retorno;
};

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```

function reverseSeq ($n) {
  $retorno = [];
  while ($n >= 1) {
    $retorno[] = $n;
    $n--;
  }
  return $retorno;
};

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Python:

```

def reverse_seq(n):
  retorno = []
  while n > 0:
    retorno.append(n)
    n = n - 1
  return retorno

```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

```

def reverse_seq(n):

```

```

  a = n;
  r = [];
  r.append(n);
  i = n;

```

```

  while i > 1:
    i = a - 1;
    a = a - 1;
    r.append(i);
  return r;

```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Parse nice int from char problem](#)

JavaScript:

```
function getAge(inputString){
  return parseInt(inputString.slice(0,1));
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Does my number look big in this?](#)

Ruby:

```
def narcissistic?( value )
  expoente = value.to_s.length
  total = 0
  value.to_s.split("").each do |c|
    total = total + (c.to_i) ** expoente
  end
  total == value
end
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function narcissistic( value ) {
  let valorString = String(value);
  let expoente = valorString.length;
  let soma = 0;
  for (i of valorString.split('')) {
    soma += Math.pow(i, expoente);
  }
  return parseInt(soma) == value;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

5 kyu

[Moving Zeros To The End](#)

JavaScript:

```
var moveZeros = function (arr) {
  let inicioRetorno = [];
  let finalRetorno = []
  for (item of arr) {
    if (item === 0) {
      finalRetorno.push(item);
    } else {
      inicioRetorno.push(item);
    }
  }
  return inicioRetorno.concat(finalRetorno);
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

```
var moveZeros = function (arr) {
  let retorno = [];
  let itensFinal = []
  for (var item of arr) {
    if (item === 0) {
      itensFinal.push(item);
    } else {
      retorno.push(item);
    }
  }

  for (var item of itensFinal) {
    retorno.push(item);
  }

  return retorno;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[What century is it?](#)

JavaScript:

```
function whatCentury(year)
{
  let seculo = '';
  if (year % 100 == 0) {
    seculo = String(year).slice(0,2);
  }
}
```

```

    seculo = seculo + obterOrdinalSeculo(seculo);
  } else {
    seculo = Number(String(year).slice(0,2)) + 1
    seculo = seculo + obterOrdinalSeculo(seculo);
  }
}

return seculo;
}

function obterOrdinalSeculo(seculo) {
  seculo = String(seculo);

  if (seculo[1] == '1' && seculo[0] != '1') return 'st';
  if (seculo[1] == '2' && seculo[0] != '1') return 'nd';
  if (seculo[1] == '3' && seculo[0] != '1') return 'rd';
  return 'th';
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Jenny's secret message](#)

JavaScript:

```

function greet(name){
  if(name === "Johnny")
    return "Hello, my love!";
  return "Hello, " + name + "!";
}

```

- 7 years ago
- [Refactor](#)

PHP:

```

function greet($name) {
  if ($name === 'Johnny') {
    return 'Hello, my love!';
  }
  return "Hello, $name!";
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Is this a triangle?](#)

JavaScript:

```

function isTriangle(a,b,c)
{
  if (a + b > c && b + c > a && a + c > b) {
    return true;
  }
  return false
}

```

- 7 years ago
- [Refactor](#)

```

function isTriangle(a,b,c)
{
  if (a + b > c && a + c > b && b + c > a) {
    return true;
  }
  return false;
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```

def isTriangle(a,b,c)
  return true if (a+b>c and a+c>b and b+c>a)
  false
end

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Java:

```

class TriangleTester{
  public static boolean isTriangle(int a, int b, int c){
    if (a + b > c && a + c > b && c + b > a) {
      return true;
    }
  }
}

```

```

    return false;
  }
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Thinkful - Object Drills: Vectors](#)

JavaScript:

```

class Vector
{
  constructor(x, y)
  {
    this.x = x;
    this.y = y;
  }

  add(a) {
    return new Vector(a.x + this.x, a.y + this.y);
  }
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Circle area inside square](#)

JavaScript:

```

function squareAreaToCircle(size){
  return (size/4 * Math.PI);
}

```

- 7 years ago
- [Refactor](#)

```

function squareAreaToCircle(size){
  return Math.PI * Math.pow(Math.sqrt(size) / 2, 2);
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

5 kyu

[First non-repeating character](#)

Ruby:

```

def first_non_repeating_letter(s)
  return s if s.to_s == ""
  s.split("").each do |caracter|
    return caracter unless s.scan(/#{caracter}/i).length > 1
  end
  return ""
end

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```

function firstNonRepeatingLetter(s) {
  let sMinusculas = s.toLowerCase();
  let arrayLetras = s.split('');
  let letrasMinusculasJaVerificadas = [];

  for (let indiceLetra in arrayLetras) {
    let letraAtual = s[indiceLetra];
    let letraAtualMinuscula = s[indiceLetra].toLowerCase();
    if (sMinusculas.substr(parseInt(indiceLetra) + 1, sMinusculas.length).indexOf(letraAtualMinuscula) == -1 && letrasMinusculasJaVerificadas.indexOf(
      return letraAtual;
    }

    letrasMinusculasJaVerificadas.push(letraAtualMinuscula);
  }
  return '';
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Multiples of 3 or 5](#)

JavaScript:



```
function solution(maximo) {
  let multiplos = [];
  for (var i=1; i<maximo ; i++) {
    if (i%3==0 || i%5==0) {
      multiplos.push(i);
    }
  }
  if (multiplos.length == 0) return 0;
  return multiplos.reduce(function(valorAnterior, valorActual) {
    return valorActual + valorAnterior;
  });
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Training JS #33: methods of Math---max\(\)\\_min\(\)\\_and\\_abs\(\)](#)

JavaScript:

```
function maxMin(arr1,arr2){
  let comparisons = [];
  for (let i in arr1) {
    comparisons.push(Math.abs(arr1[i] - arr2[i]));
  }
  return [Math.max(...comparisons), Math.min(...comparisons)];
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Easy Time Convert](#)

JavaScript:

```
function timeConvert(num) {
  if (num <=0) return "00:00";

  let seconds = Math.floor((num % 3600) % 60);
  let minutes = Math.floor((num / 60));

  return formattWith2Numbers(minutes) + ":" + formattWith2Numbers(seconds);
}

function formattWith2Numbers(num) {
  if (num < 10) return "0" + String(num);
  return String(num);
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Alternate capitalization](#)

JavaScript:

```
function capitalize(s){
  let ret1 = [];
  let ret2 = [];
  let i = 0;

  for (let q of s.split("")) {
    if (i % 2 ==1) {
      ret1.push(q.toLowerCase());
      ret2.push(q.toUpperCase());
    } else {
      ret2.push(q.toLowerCase());
      ret1.push(q.toUpperCase());
    }
    i++;
  }

  return [ret1.join(""), ret2.join("")]
};
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

5 kyu

[Compare Number](#)

JavaScript:

```
function compare(a,b){
  let float_a = parseFloat(a.replace(/^\0+/, ""));
```

```

let float_b = parseFloat(b.replace(/^0+/, ""));

if (float_a > float_b) return "greater";
if (float_a < float_b) return "less";

if (a.length > 10 || b.length > 10) {
  return compararStringDigitoPorDigito(a, b);
}
return "equal";
}

function compararStringDigitoPorDigito(a, b) {
  let da = a.split("");
  let db = b.split("");

  for (let i in da) {
    let a_atual = da[da.length - i - 1];
    let b_atual = db[db.length - i - 1];
    if (a_atual > b_atual) {
      return "greater";
    } else if (a_atual < b_atual) {
      return "less";
    }
  }

  return "equal";
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

```

function compare(a,b){
  let float_a = parseFloat(a.replace(/^0+/, ""));
  let float_b = parseFloat(b.replace(/^0+/, ""));

  console.log(a);
  console.log(b);

  console.log(float_a);
  console.log(float_b);

  if (float_a > float_b) return "greater";
  if (float_a < float_b) return "less";

  if (a.length > 10 || b.length > 10) {
    return compararStringDigitoPorDigito(a, b);
  }
  return "equal";
}

function compararStringDigitoPorDigito(a, b) {
  let d = -1;
  let da = a.split("");
  let db = b.split("");

  for (let i in da) {
    let a_atual = da[da.length - i - 1];
    let b_atual = db[db.length - i - 1];
    if (a_atual > b_atual) {
      return "greater";
    } else if (a_atual < b_atual) {
      return "less";
    }
  }

  return "equal";
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Word values](#)

JavaScript:

```

function wordValue(a) {
  let t = [];
  let r = [];
  let i = 0;
  for (let w of a) {
    console.log(w);
    r[i] = 0;
    for (let c of w.split("")) {
      let vc = c.charCodeAt(0);
      if (vc < 97 || vc > 122) continue;
      r[i] += vc - 96;
    }
    r[i] = r[i] * (i + 1);
    i++;
  }

  return r;
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Array of twins](#)

JavaScript:

```
function twins(myArray){
  let cont = {}

  for (let i of myArray) {
    if (cont[i] === undefined) cont[i] = 0;
    cont[i]++;
  }

  for (let i in cont) {
    if (cont[i] !== 2) return false;
  }

  return true;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Count number of zeros from 1 to N](#)

JavaScript:

```
function countZeros(n) {
  let ns;
  let c = 1;
  let total = 0;

  while (c <= n) {
    ns = String(c).split('');

    for (let i of ns) {
      if (i == '0') {
        total++;
      }
    }

    c++;
  }

  return total;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Start with a Vowel](#)

JavaScript:

```
function vowelStart(str){
  str = str.toLowerCase();
  let ret = '';
  for (let c of str) {
    if (c == " " || c == "," || c == "-" || c == "!") continue;
    if (is_vowel(c)) {
      ret += ' ';
    }
    ret += c;
  }

  return ret.trim();
}

function is_vowel(letter) {
  letter = letter.toLowerCase();
  return letter == "a" || letter == "e" || letter == "i" || letter == "o" || letter == "u";
}
```

- 7 years ago
- [Refactor](#)

```
function vowelStart(str){
  str = str.toLowerCase();
  let ret = '';
  for (let c of str) {
    if (c == " " || c == "," || c == "-" || c == "!") continue;
    if (is_vowel(c)) {
      ret += ' ';
    }
    ret += c;
  }

  return ret.trim();
}

function is_vowel(letter) {
  letter = letter.toLowerCase();
  return letter == "a" || letter == "e" || letter == "i" || letter == "o" || letter == "u";
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

```
function vowelStart(str){
  str = str.toLowerCase();
  let ret = '';
  for (let c of str) {
    if (c == " " || c == "," || c == "-" || c == "!") continue;
    if (is_vowel(c) /*&& ret[ret.length - 1] != " */) {
      ret += ' ';
    }
    ret += c;
  }

  return ret.trim();
}

function is_vowel(letter) {
  letter = letter.toLowerCase();
  return letter == "a" || letter == "e" || letter == "i" || letter == "o" || letter == "u";
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Order of weight](#)

JavaScript:

```
function arrange(arr){
  let pesos = [];
  for (let peso of arr) {
    if (peso.indexOf('KG') > -1) {
      pesos.push(parseInt(peso) * 1000);
    } else if (peso.indexOf('T') > -1) {
      pesos.push(parseInt(peso) * 1000 * 1000);
    } else {
      pesos.push(parseInt(peso));
    }
  }

  pesos.sort((a, b) => a - b);

  return recolocarUnidades(pesos);
}

function recolocarUnidades(pesos) {
  let retorno = [];
  for (let peso of pesos) {
    if (peso / (1000 * 1000) >= 1) {
      peso = (peso / (1000*1000)) + "T";
    } else if (peso / 1000 >= 1) {
      peso = (peso / 1000) + "KG";
    } else {
      peso = peso + "G";
    }
    retorno.push(peso);
  }
  return retorno;
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Ch4113ng3](#)

JavaScript:

```
function nerdfify(txt){
  return txt.replace(/[aA]/g, "4").replace(/[eE]/g, "3").replace(/l/g, "1");
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Retired  
[Use reduce\(\) to calculate the sum of the values in an array](#)

JavaScript:

```
function sum(array) {
  return array.reduce((sum, value) => sum + value);
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu  
[Longest vowel chain](#)

JavaScript:

```
function solve(s){
  let maior = 0;
  for (let vogais of s.split(/[b-df-hj-np-tv-z]+/)) {
    let tamanhoAtual = parseInt(vogais.length);
    if (tamanhoAtual > maior) maior = tamanhoAtual;
  }
  return maior;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[To square\(root\) or not to square\(root\)](#)

JavaScript:

```
function squareOrSquareRoot(array) {
  let retorno = [];
  for (let n of array){
    let resultado = Math.sqrt(n);
    if (resultado % 1 == 0) {
      retorno.push(resultado);
    } else {
      retorno.push (n * n);
    }
  }
  return retorno;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

```
function squareOrSquareRoot(array) {
  array.forEach(function(valor, indice, arrayOriginal) {
    const raiz = Math.sqrt(valor);
    if (raiz % 1 == 0) {
      return arrayOriginal[indice] = raiz;
    }
    return arrayOriginal[indice] = Math.pow(valor, 2);
  });
  return array;
}
//return array.map(Math.sqrt);
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Find the missing term in an Arithmetic Progression](#)

JavaScript:

```
var findMissing = function (list) {
  let diferenca = list[1] - list[0];
  let diferenca_atual = 0;
  for (let i in list) {
    if (i == 0) continue;
    i_anterior = i - 1;
    diferenca_atual = list[i] - list[i - 1];

    if (diferenca_atual != diferenca) {
      return list[i] - diferenca;
    }
  }
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Numbers in strings](#)

JavaScript:

```
function solve(s){
  let strs = s.split(/[a-zA-Z]+/);
  for (var i in strs) {
    strs[i] = parseInt(strs[i]);
    if (isNaN(strs[i])) strs[i] = 0;
  }
  strs.sort(function (a, b) { return a - b; });
  return strs[strs.length - 1];
};
```

- 7 years ago

- [Refactor](#)
- [Discuss](#)

6 kyu

[Organise duplicate numbers in list](#)

JavaScript:

```
function group(arr) {
  let retorno = [];
  let posicoes_itens = [];
  let posicao = null;

  for (let i of arr) {
    posicao = posicoes_itens.indexOf(i);
    if (posicao == -1) {
      posicoes_itens.push(i);
      posicao = posicoes_itens.indexOf(i);
    }

    if (!(retorno[posicao] instanceof Array)) {
      retorno[posicao] = [];
    }
    retorno[posicao].push(i);
  }

  return retorno;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

```
function group(arr) {
  let indiceElementos = [];
  let retorno = [];
  let posicaoArray = null
  for (item of arr) {
    posicaoArray = indiceElementos.indexOf(item);
    if (posicaoArray == -1) {
      indiceElementos.push(item);
      retorno.push([item]);
    } else {
      retorno[posicaoArray].push(item);
    }
  }

  return retorno;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Number Manipulation I \(Easy\)](#)

JavaScript:

```
function manipulate(num) {
  let stringNum = String(num);
  let metade = Math.ceil(stringNum.length / 2);
  let desconto = 0;

  if (stringNum.length % 2 == 1) desconto = 1;

  return Number(stringNum.slice(0, metade - desconto) + "0".repeat(metade));
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[!a == a ?!](#)

JavaScript:

```
const a = [];
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Kushim the Accountant: Extract \\$ values from text](#)

JavaScript:

```
function sumAccounts(text) {
  let ocorrencias = text.match(/[\-]?\$[0-9]+/g);
  let total = 0;

  for (let i of ocorrencias) {
    total += parseInt(i.replace("$", ""));
  }
}
```

```

}
return total;
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Is integer safe to use?](#)

JavaScript:

```

function SafeInteger(n) {
return Number.isSafeInteger(n);
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[What's the real floor?](#)

JavaScript:

```

function getRealFloor(n) {
if (n > 0) n--;
if (n > 13) n--;
return n;
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Remove First and Last Character](#)

JavaScript:

```

function removeChar(str){
return str.slice(1, str.length - 1);
};

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Geometry Basics: Circle Area in 2D](#)

JavaScript:

```

function circleArea(circle){
return Math.PI * Math.pow(circle.radius, 2);
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Safen User Input Part I - htmlspecialchars](#)

JavaScript:

```

function htmlspecialchars(formData) {
return formData.replace(/\&/g, "&amp;").replace(/\<&/g, "&lt;").replace(/\>/g, "&gt;").replace(/\"/g, "&quot;");
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Is he gonna survive?](#)

JavaScript:

```

function hero(bullets, dragons){
console.log(bullets)
console.log(dragons)
return bullets / dragons >= 2;
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Describe the shape](#)

JavaScript:

```
function describeTheShape( angles ){
  if (angles <= 2) return "this will be a line segment or a dot";

  let d = Math.floor(((angles - 2) * 180) / angles);

  return `This shape has ${angles} sides and each angle measures ${d}`;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Summy](#)

JavaScript:

```
function summy(stringOfInts){
  return stringOfInts.split(" ").reduce((a, b) => parseInt(a) + parseInt(b), 0);
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Love vs friendship](#)

JavaScript:

```
function wordsToMarks(string){
  let total = 0;
  for (let c = 0; c < string.length; c++) {
    total += string.charCodeAt(c) - 96;
  }

  return total;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Changing letters](#)

JavaScript:

```
function swap(st){
  return st.replace(/[aeiou]/g, function(char) { return char.toUpperCase()});
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Point in a unit circle](#)

JavaScript:

```
function pointInCircle(x,y){
  return Math.sqrt(Math.pow(x, 2) + Math.pow(y,2)) < 1;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[For Twins: L. Types](#)

JavaScript:

```
function typeValidation(variable, type) {
  return typeof variable === type
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)



8 kyu

[Find the Integral](#)

JavaScript:

```
function integrate(coefficient, exponent) {
  exponent++;
  return (coefficient/exponent) + "x^" + exponent;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Will there be enough space?](#)

JavaScript:

```
function enough(cap, on, wait) {
  return on + wait > cap ? on + wait - cap : 0;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[No Loops 2 - You only need one](#)

JavaScript:

```
function check(a,x){
  return a.indexOf(x) > -1;
};
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Heads and Legs](#)

JavaScript:

```
function animals(heads, legs){
  let chickens = 0;
  let cows = 0;

  cows = (legs - 2*heads) / 2;
  chickens = heads - cows;

  if (cows < 0 || cows % 1 !== 0) {
    return 'No solutions';
  }

  if (chickens < 0 || chickens % 1 !== 0) {
    return 'No solutions';
  }

  return[chickens, cows];
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

5 kyu

[Simple Pig Latin](#)

Ruby:

```
def pig_it text
  frase_final = ""
  text.split(" ").each do |palavra|
    if palavra.match /^[a-zA-Z]+$/
      frase_final = frase_final + palavra[1..palavra.length] + palavra[0] + "ay" + " "
    else
      frase_final = frase_final + palavra
    end
  end
  frase_final.strip
end
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

JavaScript:

```
function pigIt(str){
  let ret = "";
```

```

for (part of str.split(" ")) {
  ret += part.slice(1, part.length) + part[0] + "ay "
}

return ret.slice(0, ret.length - 1);
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Get Nth Even Number](#)

JavaScript:

```

function nthEven(n){
  return (n-1)*2;
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Simple Fun #49: Decipher](#)

JavaScript:

```

function decipher(cipher) {
  let retorno = '';
  let charCodeAtual = "";
  let numeroAtual = 0;
  for (let i = 0 ; i < cipher.length; i++) {
    charCodeAtual += String(cipher.slice(i, i+1));
    let charCodeAtualInteiro = parseInt(charCodeAtual);
    if (charCodeAtualInteiro > 50 && charCodeAtualInteiro < 130) {
      retorno += String.fromCharCode(charCodeAtual);
      charCodeAtual = "";
    }
  }
  return retorno;
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Simple Fun #202: Min And Max](#)

JavaScript:

```

function minAndMax(l, d, x) {
  let valoresQueBatem = [];
  for (let y = l; y<=d ; y++) {
    let somaCaracteres = 0;
    let arrayCaracteres = String(y).split("");

    for (let caracter of arrayCaracteres) {
      somaCaracteres += parseInt(caracter);
    }

    if (somaCaracteres == x) {
      valoresQueBatem.push(y);
    }
  }

  return [valoresQueBatem[0], valoresQueBatem[valoresQueBatem.length - 1]];
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Get list sum recursively](#)

JavaScript:

```

function sumR(x) {
  return x.reduce((a, b) => a+b, 0);
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Check if a triangle is an equable triangle!](#)

JavaScript:

```
function equableTriangle(a,b,c) {
  let perimetro = a + b + c;
  let metade_perimetro = perimetro / 2;
  let area = Math.sqrt(metade_perimetro*(metade_perimetro - a)*(metade_perimetro - b)*(metade_perimetro - c));
  return area == perimetro;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Coordinates Validator](#)

JavaScript:

```
function isValidCoordinates(coordinates){
  let coordenadas = coordinates.split(",");
  if (coordenadas.length != 2) return false;
  if( /^[a-zA-Z]+/.test(coordenadas[0])) return false;
  if( /^[a-zA-Z]+/.test(coordenadas[1])) return false;
  if (parseFloat(coordenadas[0]) != coordenadas[0]) return false;
  if (parseFloat(coordenadas[1]) != coordenadas[1]) return false;
  coordenadas[0] = parseFloat(coordenadas[0]);
  coordenadas[1] = parseFloat(coordenadas[1]);
  if (isNaN(coordenadas[0]) || coordenadas[0] < -90 || coordenadas[0] > 90) return false;
  if (isNaN(coordenadas[1]) || coordenadas[1] < -180 || coordenadas[1] > 180) return false;
  return true;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Hard Time Bomb](#)

JavaScript:

```
var wireCode = global.boom0 || global.boom1 || global.boom2 || global.boom3 || global.boom4 || global.boom5 || global.boom6 || global.boom7 || global.Bomb.CutTheWire(wireCode);
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Polish alphabet](#)

JavaScript:

```
function correctPolishLetters (string) {
  return string
    .replace(/ą/g, 'a')
    .replace(/ć/g, 'c')
    .replace(/ę/g, 'e')
    .replace(/ł/g, 'l')
    .replace(/ń/g, 'n')
    .replace(/ó/g, 'o')
    .replace(/ś/g, 's')
    .replace(/ź/g, 'z')
    .replace(/ż/g, 'z');
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Vowel Changer](#)

JavaScript:

```
function vowelChange(str, vow) {
  return str.replace(/[aeiou]/g, vow);
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Calculate Price Excluding VAT](#)

JavaScript:

```
//return price without vat
function excludingVatPrice(price){
  if (price == null) return -1;

  return parseFloat(parseFloat(price/1.15).toFixed(2));
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sum of array singles](#)

JavaScript:

```
function repeats(arr){
  let ja_ocorreram = [];
  let ainda_nao_ocorreram = [];
  for (i in arr) {
    if ((ja_ocorreram.indexOf(arr[i]) > -1) || arr.slice(parseInt(i)+ 1, arr.length).indexOf(arr[i]) > -1) {
      ja_ocorreram.push(arr[i]);
    } else {
      ainda_nao_ocorreram.push(arr[i]);
    }
  }

  return ainda_nao_ocorreram.reduce((a, b) => a + b, 0);
};
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Counting sheep...](#)

JavaScript:

```
function countSheeps(arrayOfSheep) {
  let soma = 0;
  for (let i of arrayOfSheep) {
    if (i) soma++;
  }
  return soma;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Compare Strings by Sum of Chars](#)

JavaScript:

```
function compare(s1, s2) {
  let total_s1 = 0;
  let total_s2 = 0;
  let posicao_s1 = 0;
  let posicao_s2 = 0;

  if (typeof s1 == "string") {
    s1 = s1.toUpperCase();
    while(posicao_s1 < s1.length) {
      let valor_atual = s1.charCodeAt(posicao_s1);
      if (valor_atual < 65 || valor_atual > 90) {
        total_s1 = 0;
        break;
      }
      total_s1 += valor_atual;
      posicao_s1++;
    }
  } else {
    total_s1 = 0;
  }

  if (typeof s2 == "string") {
    s2 = s2.toUpperCase();
    while(posicao_s2 < s2.length) {
      let valor_atual = s1.charCodeAt(posicao_s1);
      if (valor_atual < 65 || valor_atual > 90) {
        total_s2 = 0;
        break;
      }
      total_s2 += s2.charCodeAt(posicao_s2);
      posicao_s2++;
    }
  } else {
    total_s2 = 0;
  }

  return total_s1 == total_s2;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Difference between years. \(Level 1\)](#)

JavaScript:

```
var howManyYears = function(date1, date2){
  let partes_data1 = date1.split('/');
  let ano1 = partes_data1[0];
  let partes_data2 = date2.split('/');
  let ano2 = partes_data2[0];
  return Math.abs(ano2 - ano1);
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Break camelCase](#)

JavaScript:

```
// complete the function
function solution(string) {
  let retorno = '';
  for (let i = 0, len = string.length; i < len; i++) {
    if (string[i].charCodeAt(0) >= 65 && string[i].charCodeAt(0) <= 90) {
      retorno += " " + string[i];
    } else {
      retorno += string[i];
    }
  }
  return retorno;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Sum The Tree](#)

JavaScript:

```
// return the sum of all values in the tree, including the root
function sumTheTreeValues(root){
  let listaNos = [root];
  let soma = 0;
  while(listaNos.length > 0) {
    soma += listaNos[0].value;
    if (listaNos[0].left != null) {
      listaNos.push(listaNos[0].left);
    }
    if (listaNos[0].right != null) {
      listaNos.push(listaNos[0].right);
    }
    listaNos.shift();
  }
  return soma;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Equal Sides Of An Array](#)

Ruby:

```
def find_even_index(arr)
  arr.each_index do |indice|
    esquerda = arr.slice(0, indice)
    soma_esquerda = esquerda.empty? ? 0 : esquerda.inject(:+)
    soma_direita = arr.slice(indice+1, arr.length).inject(:+)
    soma_direita = 0 if soma_direita.nil?
    puts soma_direita.inspect
    return indice if soma_direita == soma_esquerda
  end
  -1
end
```

- 7 years ago
- [Refactor](#)

JavaScript:

```
function findEvenIndex(arr)
{
  console.log(arr);
  for (var i in arr) {
    i = parseInt(i);

    if (i == arr.length - 2) {
      break;
    }

    if (calcularSoma(arr.slice(0, i+1)) == calcularSoma(arr.slice(2+i))) {
      return i + 1;
    }
  }
}
```

```

    return -1;
  }

  function calcularSoma(array) {
    return array.reduce(function(valorAnterior, valorAtual) {
      return valorAnterior + valorAtual;
    });
  }

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[You're a square!](#)

JavaScript:

```

var isSquare = function(n){
  return (Math.sqrt(n) % 1 == 0);
}

```

- 7 years ago
- [Refactor](#)

```

var isSquare = function(n){
  if (Math.sqrt(n) % 1 == 0) return true;
  return false;; // fix me
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Basic Calculator](#)

JavaScript:

```

function calculate(num1, operation, num2) {
  if (operation == "+") {
    return num1 + num2;
  } else if (operation == "-") {
    return num1 - num2;
  } else if (operation == "*") {
    let retorno = num1 * num2;
    if (retorno == -0) retorno = 0;
    return retorno;
  } else if (operation == "/") {
    if (num2 == 0) return null;
    return num1 / num2;
  }
  return null;
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Lucky Sevens](#)

JavaScript:

```

function luckySevens(arr) {
  let total = 0;
  for (let indiceLinha in arr) {
    let anterior = 0;
    let proximo = 0;
    let atual = 0;

    for (let indiceColuna in arr[indiceLinha]) {
      let arr1 = arr[indiceLinha];
      if (indiceColuna > 0) anterior = arr1[indiceColuna - 1];
      proximo = arr1[parseInt(indiceColuna) + 1];
      if (proximo == undefined) proximo = 0;
      atual = arr1[indiceColuna];

      let daLinhaAcima = 0;
      if (indiceLinha > 0) {
        daLinhaAcima = arr[indiceLinha - 1][indiceColuna];
      }
      let daLinhaAbaixo = 0;
      if (indiceLinha < arr.length - 1) {
        daLinhaAbaixo = arr[parseInt(indiceLinha) + 1][indiceColuna];
      }

      if (atual == 7) {
        if (Math.cbrt(anterior + proximo + daLinhaAcima + daLinhaAbaixo) % 1 == 0) {
          total++;
        }
      }
    }
  }
  return total;
}

```

- 7 years ago

- [Refactor](#)
- [Discuss](#)

6 kyu

[CamelCase Method](#)

JavaScript:

```
String.prototype.camelCase=function(){
  let partes = this.split(" ");
  let retorno = [];
  for (parte of partes) {
    retorno.push(parte.substr(0, 1).toUpperCase() + parte.substr(1));
  }
  return retorno.join('');
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[X marks the spot!](#)

JavaScript:

```
function x(n) {
  let retorno = [];
  let x = 0;
  while (x < n) {
    let linha = [];
    for (let y = 0; y < n; y++) linha[y] = 0;
    linha[x] = 1;
    linha[linha.length - x - 1] = 1;
    retorno.push(linha);
    x++;
  }
  return retorno;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Is n divisible by \(...\)?](#)

JavaScript:

```
function isDivisible(){
  let numeroADividir = arguments['0'];
  for (let i in arguments) {
    if (i == '0') continue;
    if (numeroADividir % arguments[i] != 0) return false;
  }

  return true;
}
```

- 7 years ago
- [Refactor](#)

7 kyu

[Alternate case](#)

JavaScript:

```
function alternateCase(s) {
  let indiceCaracterAtual = 0;
  let retorno = '';
  while (indiceCaracterAtual < s.length) {
    let caracterAtual = s.slice(indiceCaracterAtual, indiceCaracterAtual + 1);
    let c = s.charCodeAt(indiceCaracterAtual);
    if (c >= 65 && c <= 90) {
      retorno += caracterAtual.toLowerCase();
    } else if (c >= 97 && c <= 122) {
      retorno += caracterAtual.toUpperCase();
    } else {
      retorno += caracterAtual;
    }
    indiceCaracterAtual++;
  }

  return retorno;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

```
function alternateCase(s) {
  let retorno = '';
  let charCodeAtual = null;
  for (let i=0; i<s.length; i++) {
    charCodeAtual = s.charCodeAt(i);
```

```

if (charCodeAtual >=65 && charCodeAtual <=90) {
  retorno = retorno + s[i].toLowerCase();
} else if (charCodeAtual >=97 && charCodeAtual <=122) {
  retorno = retorno + s[i].toUpperCase();
} else {
  retorno = retorno + s[i];
}

}
return retorno;
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Lowercase strings in array](#)

JavaScript:

```

function arrayLowerCase(arr) {
  retorno = [];
  for (var item of arr) {
    if (typeof item == "string") {
      item = item.toLowerCase();
    }
    retorno.push(item);
  }
  return retorno;
}
// return array of strings in lowercase
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[IP Validation](#)

PHP:

```

function isValidIP(string $str): bool
{
    $matches = array();
    preg_match('/^\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}$/z', $str, $matches);

    if (! is_array($matches) || count($matches) != 5) {
        return false;
    }

    for ($i = 1; $i <= 4; $i++) {
        $matches[$i] = (int) $matches[$i];
        if ($matches[$i] < 0 || $matches[$i] > 255) {
            return false;
        }
    }

    return true;
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

5 kyu

[Resistor Color Codes, Part 2](#)

JavaScript:

```

function encodeResistorColors(ohmsString) {
  let valorString = String(parseFloat(ohmsString)).replace(/\.|/, '');
  let existiaPonto = ohmsString.indexOf('.') > -1 ? true : false;
  let comprimentoValor = String(parseInt(ohmsString)).length;
  let primeiraCasa = obterTextoCor(valorString[0]);
  let segundaCasa;
  if (comprimentoValor > 1 || existiaPonto) {
    segundaCasa = obterTextoCor(valorString[1]);
  } else {
    segundaCasa = obterTextoCor(0);
  }
  let terceiraCasa = '';

  if (ohmsString.indexOf("M") > -1) {
    terceiraCasa = obterTextoCor(comprimentoValor + 4);
  } else if (ohmsString.indexOf("k") > -1) {
    terceiraCasa = obterTextoCor(comprimentoValor + 1);
  } else {
    terceiraCasa = obterTextoCor(comprimentoValor - 2);
  }

  return `${primeiraCasa} ${segundaCasa} ${terceiraCasa} gold`;
}

function obterTextoCor(numero) {
  let relacao = ['black', 'brown', 'red', 'orange', 'yellow', 'green', 'blue', 'violet', 'gray', 'white'];
}

```



```
return relacao[numero];
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def encode_resistor_colors(ohms_string)
  valor_string = String(ohms_string.to_f).sub(/\./, "")
  puts valor_string
  existia_ponto = ohms_string.index('.')&nil? ? false : true;
  comprimento_valor = String(ohms_string.to_i).length
  primeira_casa = obter_texto_cor(valor_string[0]);

  if (comprimento_valor > 1 || existia_ponto) then
    segunda_casa = obter_texto_cor(valor_string[1]);
  else
    segunda_casa = obter_texto_cor(0);
  end

  if (ohms_string.index("M")&nil? === false) then
    terceira_casa = obter_texto_cor(comprimento_valor + 4)
  elsif (ohms_string.index("k")&nil? === false) then
    terceira_casa = obter_texto_cor(comprimento_valor + 1)
  else
    terceira_casa = obter_texto_cor(comprimento_valor - 2)
  end

  "#{primeira_casa} #{segunda_casa} #{terceira_casa} gold"
end

def obter_texto_cor(numero)
  ['black', 'brown', 'red', 'orange', 'yellow', 'green', 'blue', 'violet', 'gray', 'white'][numero.to_i]
end
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Simple Fun #262: Case Unification](#)

JavaScript:

```
function caseUnification(s) {
  let indiceCaracterAtual = 0;
  let totalMaiusculas = 0;
  let totalMinusculas = 0;

  while (indiceCaracterAtual < s.length) {
    let codigoAsciiCaracterAtual = s.charCodeAt(indiceCaracterAtual);
    if (codigoAsciiCaracterAtual >= 65 && codigoAsciiCaracterAtual <= 90) {
      totalMaiusculas++;
    } else if (codigoAsciiCaracterAtual >= 97 && codigoAsciiCaracterAtual <= 122) {
      totalMinusculas++;
    }
    indiceCaracterAtual++;
  }

  if (totalMaiusculas > totalMinusculas) {
    return s.toUpperCase();
  }
  return s.toLowerCase();
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Return 1, 2, 3 randomly](#)

JavaScript:

```
function one_two_three() {
  while (true) {
    let rodada1 = one_two();
    let rodada2 = one_two();

    if (rodada1 == 1 && rodada2 == 1) return 1;
    if (rodada1 == 1 && rodada2 == 2) return 2;
    if (rodada1 == 2 && rodada2 == 1) return 3;
  }
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Simple Fun #182: Happy "g"](#)

JavaScript:

```
function gHappy(str) {
  return str.replace(/g{2,}/g, '').indexOf('g') == -1;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[altERnaTIng cAsE <=> ALTerNAtiNG CaSe](#)

JavaScript:

```
String.prototype.toAlternatingCase = function () {
  let retorno = '';
  for (let i = 0 ; i < this.length ; i++) {
    let ascii = this.charCodeAt(i);
    if (ascii >=65 && ascii <=90) {
      retorno += this[i].toLowerCase();
    } else {
      retorno += this[i].toUpperCase();
    }
  }
  return retorno;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Find the first non-consecutive number](#)

JavaScript:

```
function firstNonConsecutive (arr) {
  let anterior = null;
  for (let i of arr) {
    if (anterior !== null && i - 1 !== anterior) {
      return i;
    }
    anterior = i;
  }
  return null;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Simple Fun #221: Furthest Distance Of Same Letter](#)

JavaScript:

```
function distSameLetter(s) {
  let posicoesIniciais = {};
  let maiorDistancia = 0;
  let letraMaiorDistancia = '';
  for (let posicao in s) {
    let letra = s[posicao];
    if (posicoesIniciais[letra] == undefined) {
      posicoesIniciais[letra] = posicao;
    } else if (posicao - posicoesIniciais[letra] + 1 > maiorDistancia) {
      letraMaiorDistancia = letra;
      maiorDistancia = posicao - posicoesIniciais[letra] + 1;
    }
  }
  return letraMaiorDistancia + maiorDistancia;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Simple Fun #204: Smallest Integer](#)

Ruby:

```
def smallest_integer(matrix)
  matrix_flatten = matrix.flatten.sort!
  return 0 if matrix_flatten[-1] < 0
  numero_atual = nil
  (0 .. matrix_flatten[-1]).each do |n|
    numero_atual = n
    return n unless matrix_flatten.include? n
  end
  numero_atual + 1
end
```

- 7 years ago

- [Refactor](#)
- [Discuss](#)

7 kyu

[Flatten and sort an array](#)

Ruby:

```
def flatten_and_sort(array)
  array.flatten.sort
end
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

5 kyu

[A Chain adding function](#)

JavaScript:

```
function add (valor) {
  var funcaoAuxiliar = function(v) {
    return add(valor + v);
  }
  funcaoAuxiliar.valueOf = function() {
    return valor;
  }
  return funcaoAuxiliar;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Simple Fun #6: Is Infinite Process?](#)

JavaScript:

```
function isInfiniteProcess(a, b) {
  if ((a > b) || (a + b) % 2 == 1) return true;

  return false;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[All unique](#)

JavaScript:

```
function hasUniqueChars(str){
  let caracteresAnteriores = []
  for (character of str.split('')) {
    if (caracteresAnteriores.indexOf(character) > -1) return false;
    caracteresAnteriores.push(character);
  }

  return true;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Simple Fun #17: Rounders](#)

JavaScript:

```
function rounders(value) {
  let retorno = '';
  let valorAIterar = String(value).split('').reverse();
  let acrescetarAoProximo = 0;
  for (let i in valorAIterar) {
    let atual = parseInt(valorAIterar[i]) + acrescetarAoProximo;
    if (atual >=5) {
      acrescetarAoProximo = 1;
    } else {
      acrescetarAoProximo = 0;
    }

    if (i < String(value).length - 1) atual = '0';

    retorno = String(atual) + retorno;
  }

  return parseInt(retorno);
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

5 kyu

[Directions Reduction](#)

Ruby:

```
def dirReduc(arr)
  opostos = {'NORTH' => 'SOUTH', 'SOUTH' => 'NORTH', 'EAST' => 'WEST', 'WEST' => 'EAST'}
  reducao = []
  arr.each_with_index do |elemento, indice|
    if opostos[elemento] == reducao.last
      reducao.pop
    else
      reducao.push elemento
    end
  end
  reducao
end
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Valid Parentheses](#)

Ruby:

```
def valid_parentheses(string)
  total_aberturas = 0
  string.split('').each do |caracter|
    if caracter == ")"
      return false if total_aberturas == 0
      total_aberturas = total_aberturas - 1
    elsif caracter == "("
      total_aberturas = total_aberturas + 1
    end
  end
  total_aberturas == 0
end
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

```
def valid_parentheses(string)
  retorno = true
  total_parenteses_abertura = 0
  total_parenteses_fechamento = 0
  string.each_char do |char|
    if char == "("
      total_parenteses_abertura = total_parenteses_abertura + 1
    elsif char == ")"
      return false if total_parenteses_abertura == 0
      total_parenteses_abertura = total_parenteses_abertura - 1
    end
  end
  total_parenteses_abertura == 0
end
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Two Oldest Ages](#)

Ruby:

```
# return the two oldest/oldest ages within the array of ages passed in.
def two_oldest_ages(ages)
  maior = 0
  segundo_maior = 0
  ages.each do |age|
    if age > maior
      segundo_maior = maior
      maior = age
    elsif age > segundo_maior
      segundo_maior = age
    end
  end
  [segundo_maior, maior]
end
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

5 kyu

[The Hashtag Generator](#)

JavaScript:

```
function generateHashtag (str) {
  let retorno = '';
  for (let palavra of str.split(" ")) {
    retorno = retorno + palavra.charAt(0).toUpperCase() + palavra.slice(1);
  }

  if (retorno == '') return false;

  retorno = "#" + retorno;

  if (retorno.length > 140) return false;

  return retorno;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Ruby:

```
def generateHashtag str
  retorno = (str.gsub /\s+/, ' ').split(" ").map {|palavra| palavra.capitalize}.join("")

  return false if retorno.length >=139 or retorno == "";
  "#" + retorno;
end
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function generateHashtag($str) {
  $stringReturn = '';

  // Changing 2+ spaces by a unique space
  $str = preg_replace("/\s+/", " ", $str);

  foreach (explode(' ', $str) as $word) {
    $stringReturn = $stringReturn . ucfirst($word);
  }

  if (strlen($stringReturn) >=139 || empty($stringReturn)) return false;

  return "#" . $stringReturn;
}
```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

```
function generateHashtag($str) {
  $stringReturn = '';

  // Changing 2+ spaces by a unique space
  $str = preg_replace("/\s+/", " ", $str);

  foreach (explode(' ', $str) as $word) {
    $stringReturn = $stringReturn . ucfirst($word);
  }

  if (strlen($stringReturn) >139 || empty($stringReturn)) return false;

  return "#" . $stringReturn;
}
```

- 6 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Number of People in the Bus](#)

JavaScript:

```
var number = function(busStops){
  retorno = 0;
  for (let movimentacoes of busStops) {
    retorno = retorno + movimentacoes[0] - movimentacoes[1];
  }

  if (retorno < 0) retorno = 0;
  return retorno;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Remove exclamation marks](#)

JavaScript:

```
function removeExclamationMarks(s) {
  return s.split('!').join('');
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function remove_exclamation_marks($string) {
  return str_replace('!', '', $string);
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Simple Fun #137: S2N](#)

JavaScript:

```
function S2N(m, n) {
  let soma = 0;
  let baseAtual = 0;
  let expoenteAtual = 0;

  while (baseAtual <= m) {
    expoenteAtual = 0;
    while (expoenteAtual <= n) {
      soma += Math.pow(baseAtual, expoenteAtual);
      expoenteAtual++;
    }
    baseAtual++;
  }

  return soma;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Volume of a Cuboid](#)

JavaScript:

```
var Kata;

Kata = (function() {
  function Kata() {}

  Kata.getVolumeOfCuboid = function(length, width, height) {
    return length * width * height;
  };

  return Kata;
})();
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
$kata = new class {
  public function get_volume_of_cuboid($length, $width, $height) {
    return $length * $width * $height;
  }
};
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Java:

```
public class Kata {

  public static double getVolumeOfCuboid(final double length, final double width, final double height) {
    // Your code here
    return length * width * height;
  }

}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Python:



```
    return [maiorDiferenca, menorDiferenca];
  }

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Simple Fun #13: Magical Well](#)

JavaScript:

```
function magicalWell(a, b, n) {
  let retorno = 0;
  while (n > 0) {
    retorno = retorno + a * b;
    a++;
    b++;
    n--;
  }
  return retorno;
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

PHP:

```
function magical_well($a, $b, $n) {
  $retorno = 0;
  while ($n > 0) {
    $retorno = $retorno + $a * $b;
    $n--;
    $a++;
    $b++;
  }

  return $retorno;
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Keep Hydrated!](#)

JavaScript:

```
function litres(time) {
  return Math.floor(0.5*time);
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Character limits: How long is your piece of string?](#)

JavaScript:

```
function charCheck(text, max, spaces){
  if (! spaces) {
    text = text.replace(/\s/g, '');
  }

  let estouroulimite = false;
  if (text.length > max) {
    estouroulimite = true;
  }

  return [!estouroulimite, text.substr(0, max)];
};

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

```
function charCheck(text, max, spaces){
  //Do your magic here!
  if (! spaces) {
    text = text.split(' ').join('');
  }

  let booleanoTamanho = text.length <= max;

  return [booleanoTamanho, text.substr(0, max)];
};

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)



7 kyu

[Replace all items](#)

JavaScript:

```
function replaceAll(seq, find, replace) {
  console.log(typeof seq);
  console.log(seq);
  console.log(find);
  console.log(replace);

  if (Array.isArray(seq)) {
    seq.forEach(function(item, i) {
      if (item == find) {
        seq[i] = replace;
      }
    });
  } else if (typeof seq == "string") {
    return seq.split(find).join(replace);
  }
  return seq;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Are they square?](#)

JavaScript:

```
var isSquare = function(arr){
  console.log(arr);
  if (!(Array.isArray(arr)) || arr.length == 0) {
    return undefined;
  }
  return arr.every(item => Math.sqrt(item) == Math.floor(Math.sqrt(item)))
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Are they the "same"?](#)

JavaScript:

```
function comp(array1, array2){
  if(array1 == null || array2 == null) {
    return false;
  }

  for (let item of array2) {
    let posicaoNoArray1 = array1.indexOf(Math.sqrt(item));
    if (posicaoNoArray1 == -1) {
      return false;
    }

    array1.splice(posicaoNoArray1, 1);
  }
  return true;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Find The Parity Outlier](#)

PHP:

```
function find($integers) {
  $pares = [];
  $impares = [];
  foreach($integers as $numero) {
    if ($numero % 2 == 0) {
      $pares[] = $numero;
    } else {
      $impares[] = $numero;
    }
  }

  if (count($pares) == 1) {
    return $pares[0];
  } elseif (count($impares) == 1) {
    return $impares[0];
  }

  throw new \InvalidArgumentException('Existe mais de 1 par e mais de 1 impar');
}
```

- 7 years ago

- [Refactor](#)
- [Discuss](#)

7 kyu

[Remove the minimum](#)

JavaScript:

```
function removeSmallest(numbers) {
  var retorno = numbers;
  retorno.splice(retorno.indexOf(Math.min(...numbers)),1);
  return retorno;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

8 kyu

[Remove String Spaces](#)

Ruby:

```
def no_space(x)
  x.gsub(/\s/, "")
end
```

- 7 years ago
- [Refactor](#)

Python:

```
def no_space(x):
  return x.replace(" ", "")
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Count the Characters](#)

PHP:

```
function count_char(string $s, string $c): int {
  // Your mission, should you choose to accept it.
  $c = strtolower($c);
  $s = strtolower($s);
  $total = 0;
  for ($i = 0; $i < strlen($s); $i++) {
    if ($s[$i] == $c) {
      $total++;
    }
  }
  return $total;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Jaden Casing Strings](#)

PHP:

```
function toJadenCase($string)
{
  $partes = explode(' ', $string);

  foreach($partes as $indice => $parte) {
    $partes[$indice] = ucfirst($parte);
  }

  return implode($partes, ' ');
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Two to One](#)

PHP:

```
function longest($a, $b) {
  $string = $a . $b;
  return extrairCaracteresUnicos($string);
}

function extrairCaracteresUnicos($string) {
```

```

$caracteresUnicos = array();
for ($i = 0 ; $i < strlen($string); $i++) {
    if (false === array_search($string[$i], $caracteresUnicos)) {
        $caracteresUnicos[] = $string[$i];
    }
}
sort($caracteresUnicos);

return implode($caracteresUnicos, '');
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Get the Middle Character](#)

JavaScript:

```

function getMiddle(s)
{
    var posicaoCaracteresMeio = null;
    posicaoMeio = Math.floor(s.length / 2) - 1;

    if (s.length & 1 == 1) {
        return s.substring(posicaoMeio + 1, posicaoMeio + 2);
    }
    return s.substring(posicaoMeio, posicaoMeio + 2);
}

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Is a number prime?](#)

Ruby:

```

# Test if number is prime
def isPrime(num)
    num = num.to_i
    return false unless num > 1
    divisor = num / 2
    while divisor >= 2
        return false if num / divisor == num.to_f / divisor
        divisor = divisor - 1
    end
    true
end

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[#~For Kids~# d/m/Y -> Day of the week.](#)

Ruby:

```

require 'date'

def dayOfTheWeek(date)
    DateTime.parse(date).strftime('%A')
end

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Sum of two lowest positive integers](#)

Ruby:

```

def sum_two_smallest_numbers(numbers)
    numbers.sort!
    numbers[0] + numbers[1]
end

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

5 kyu

[What's a Perfect Power anyway?](#)

Ruby:

```

def isPP(numero)
    base = 2
    expoente = 2
    pares = []

```

```

while base ** expoente <= numero do
  while base ** expoente <= numero do
    resultado = base ** expoente
    pares.push(base, expoente) if resultado == numero
    base = base + 1
  end

  base = 2
  expoente = expoente + 1
end

pares = nil if pares.empty?
return pares
end

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

6 kyu

[Split Strings](#)

Ruby:

```

def solution(str)
  i = 0
  array_final = []
  while i < (str.length.to_i + 1)/2 do
    resultado = str.slice(i*2, 2)
    resultado = resultado + "_" if resultado.length.to_i < 2
    array_final.push(resultado)
    i = i + 1
  end
  array_final
end

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

```

def solution(str)
  i = 0
  array_final = []
  puts str
  while i < (str.length.to_i + 1)/2 do
    resultado = str.slice(i*2, 2)
    resultado = resultado + "_" if resultado.length.to_i < 2
    array_final.push(resultado)
    i = i + 1
  end
  puts array_final.inspect
  array_final
end

```

- 7 years ago
- [Refactor](#)

7 kyu

[Friend or Foe?](#)

Ruby:

```

def friend(friends)
  friends.map{|nome|nome if nome.length==4}.compact
end

```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

7 kyu

[Mumbling](#)

Ruby:

```

def accum(s)
  i=-1
  texto = s.chars.map do |item|
    i = i+1
    item.upcase + item.downcase * i + "-"
  end.join
  texto[0, texto.length - 1]
end

```

- 7 years ago
- [Refactor](#)

Retired

[Circles intersection](#)

JavaScript:

```

function circles_intersects(circle1, circle2) {
  let distance = Math.sqrt(Math.abs(circle1.center.x - circle2.center.x) + Math.abs(circle1.center.y - circle2.center.y)) ;

```

```
return (circle1.radius + circle2.radius) > distance;
}
```

- 7 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Number of diagonals](#)

PHP:

```
function diagonals($sides) {
    return $sides * ($sides - 3) / 2;
}
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Sum of itens major than 3](#)

Ruby:

```
def sum_items
  t = 0
  items.each do |i|
    t += i if i > 3
  end
  t
end
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Max number](#)

Ruby:

```
def max(items)
  r = 0

  for i in items do
    if i > r
      r = i
    end
  end

  r
end
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Number of vowels](#)

Ruby:

```
def vowels arg
  total = 0
  arg.downcase!

  arg.each_char do |c|
    if c == "a" or c == 'e' or c == "i" or c == "o" or c == "u"
      total = total + 1
    end
  end

  total
end
```

- 5 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Alphabet order](#)

Ruby:

```
def order s1, s2
  return s1.downcase() < s2.downcase() ? 1 : 2
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Sum of items major than 3](#)

Ruby:

```
def sum_3 arr
  sum = 0
  arr.each do | item |
    sum = sum + item if item > 3
  end
  sum
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Draft

[andreapt82's Kumite #67](#)

JavaScript:

```
function sum(items) {
  let sum = 0;

  for (const item of items) {
    sum = sum + (item.unitary_price * item.quantity);
  }

  return sum;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Sum of all items is 10](#)

Ruby:

```
def sum arr
  arr.each {|i|
    return false if i.reduce(:+) != 10
  }

  true
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Pie in the face](#)

Ruby:

```
def game data
  if data[0] < data[1]
    return data[2] ? 1 : 2
  else
    return data[2] ? 2 : 1
  end
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Minor items](#)

Ruby:

```
def minor arr, limit
  ret = []
  arr.each{|i|
    ret.push(i) if i < limit
  }
  ret
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Expression with square brackets](#)

Ruby:

```
def solve expression
  return eval(expression.gsub("[", "(").gsub("]", ")"))
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Sum of faces of dice you can see](#)

Ruby:

```
def sum face
  21 - face
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[A great number in a list](#)

PHP:

```
function hasBigNumber($numbers) {
    $sum = array_sum($numbers);

    foreach ($numbers as $number) {
        if ($sum - $number < $number) {
            return true;
        }
    }
    return false;
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Is the calculation true?](#)

Ruby:

```
def calculate a, operation, b, result
  return ((eval "#{a} #{operation} #{b}").to_i == result)
end
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Second degree](#)

JavaScript:

```
function segundoGrau(a, b, c) {
  const delta = b*b - 4*a*c;
  if (delta < 0) {
    return null;
  }

  const root1 = (-b + Math.sqrt(delta)) / (4 * a);

  if (delta == 0) {
    return [root1];
  }

  const root2 = (-b - Math.sqrt(delta)) / (4 * a);

  return [root1, root2];
}
```

- 3 years ago
- [Refactor](#)
- [Discuss](#)

Retired

[Sum of two items is other item](#)

Ruby:

```
def sum items
  items.each_with_index { |item1, index1|
    items.each_with_index { |item2, index2|
      if index1 == index2
        next
      end
      return true if items.include? item1 + item2
    }
  }
}
```

false  
end

- 3 years ago
- [Refactor](#)
- [Discuss](#)
  
- © 2024 Codewars
- [About](#)
- [API](#)
- [Blog](#)
- [Privacy](#)
- [Terms](#)
- [Code of Conduct](#)
- [Contact](#)

built on **Qualified**   
by andela

built on 