**Ruby Quest05**

Remember to git add && git commit && git push each exercise!

We will execute your function with our test(s), please DO NOT PROVIDE ANY TEST(S) in your file

For each exercise, you will have to create a folder and in this folder, you will have additional files that contain your work. Folder names are provided at the beginning of each exercise under submit directory and specific file names for each exercise are also provided at the beginning of each exercise under submit file(s).

**My Each**

* Submit directory: ex00
* Submit file: ["my\_each.rb"]

Time to print each elements of an array.

Create a function my\_each which receives an integer array as parameter and iterate over the array and use a function which print for each value. This function return nothing.

**Function prototype** (ruby)

##

##

## QWASAR.IO -- my\_each

##

##

## @param {String[]} param\_1

##

##

def my\_each(param\_1)

end

**Example 00**

Input: ["blah1", "blah2", "blah3"]

Output: blah1

blah2

blah3

Return Value: nil

**Example 01**

Input: ["blah1", "blah2"]

Output: blah1

blah2

Return Value: nil

**Example 02**

Input: ["1arg"]

Output: 1arg

Return Value: nil

*Tips* Google while YOURCODINGLANGUAGE Google for YOURCODINGLANGUAGE Google array.length YOURCODINGLANGUAGE

**My Map Mult Two**

* Submit directory: ex01
* Submit file: ["my\_map\_mult\_two.rb"]

Multiply by 2 each elements of an array.

Create a function my\_map\_mult\_two which receives an integer array as parameter and iterate over the array, perform a multiplication by 2 on each value and return the new array collected.

**Function prototype** (ruby)

##

##

## QWASAR.IO -- my\_map\_mult\_two

##

##

## @param {Integer[]} param\_1

##

## @return {integer[]}

##

def my\_map\_mult\_two(param\_1)

end

**Example 00**

Input: [1, 2, 3, 4, 5]

Output:

Return Value: [2, 4, 6, 8, 10]

**Example 01**

Input: []

Output:

Return Value: []

*Tips* Google while YOURCODINGLANGUAGE Google for YOURCODINGLANGUAGE Google array.length YOURCODINGLANGUAGE

**My Count On It**

* Submit directory: ex02
* Submit file: ["my\_count\_on\_it.rb"]

Count the size of each elements in an array.

Create a function my\_count\_on\_it, which receives a string array as parameter and returns an array with the length of each strings.

**Function prototype** (ruby)

##

##

## QWASAR.IO -- my\_count\_on\_it

##

##

## @param {String[]} param\_1

##

## @return {integer[]}

##

def my\_count\_on\_it(param\_1)

end

**Example 00**

Input: ["This", "is", "the", "way"]

Output:

Return Value: [4, 2, 3, 3]

**Example 01**

Input: ["aBc", "AbcE Fgef1"]

Output:

Return Value: [3, 10]

**Example 02**

Input: ["aBc"]

Output:

Return Value: [3]

*Tips* Google while YOURCODINGLANGUAGE Google for YOURCODINGLANGUAGE Google array.length YOURCODINGLANGUAGE

**My Array Uniq**

* Submit directory: ex03
* Submit file: ["my\_array\_uniq.rb"]

Create an array without any duplicates.

Create a function my\_array\_uniq, which receives an integer array as a parameter and returns an array with those integers but without any duplicates.

**Function prototype** (ruby)

##

##

## QWASAR.IO -- my\_array\_uniq

##

##

## @param {Integer[]} param\_1

##

## @return {integer[]}

##

def my\_array\_uniq(param\_1)

end

**Example 00**

Input: [1, 1, 2]

Output:

Return Value: [1, 2]

**Example 01**

Input: []

Output:

Return Value: []

**Example 02**

Input: [1, 1, 1, 2, 3, 4, 1]

Output:

Return Value: [1, 2, 3, 4]

*Tip* Google filter/uniq YOURCODINGLANGUAGE

**My Is Sort**

* Submit directory: ex04
* Submit file: ["my\_is\_sort.rb"]

Let's create a function which will tell us if an array is sorted or not. What is sorted? :-)

Write a function that takes an integer array as a parameter (input) and returns a boolean (true/false).

Your function should return true if the integer array is sorted in either ASC(ascending) or DESC(descending) order. Your function should return false if the integer array is not sorted.

Numbers will be from -2\_000\_000 to 2\_000\_000 Array might have duplicate(s).

**Function prototype** (ruby)

##

##

## QWASAR.IO -- my\_is\_sort

##

##

## @param {Integer[]} param\_1

##

## @return {boolean}

##

def my\_is\_sort(param\_1)

end

**Example 00**

Input: [1, 1, 2]

Output:

Return Value: true

**Example 01**

Input: [2, 1, -1]

Output:

Return Value: true

**Example 02**

Input: [4, 7, 0, 3]

Output:

Return Value: false

**Example 03**

Input: []

Output:

Return Value: true

*Tips* Google: what is a Boolean Google: sort ascending