**Ruby Quest03**

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 First Script With Args**

* Submit directory: ex00
* Submit file: ["my\_first\_script\_with\_args.rb"]

Let's do our first loop statement!

Create a file my\_first\_script\_with\_args.rb.

It will print any argument received to the script

**Example 00 (In Javascript)**

$>node my\_first\_script\_with\_args.js blah1 blah2 blah3

blah1

blah2

blah3

$>

**Example 01 (In Python)**

$>python my\_first\_script\_with\_args.py blah1

blah1

$>

**Example 02 (In Ruby)**

$>ruby my\_first\_script\_with\_args.rb "blah1 blah2 blah3"

blah1 blah2 blah3

$>

*Tip* Google the following: script in YOURCODINGLANGUAGE receiving arguments (argv)

**My Is Negative**

* Submit directory: ex01
* Submit file: ["my\_is\_negative.rb"]

Let's get starting with some if-else statement!

Create a my\_is\_negative function. This function my\_is\_negative returns 1 or 0 depending on the integer's sign entered as a parameter. If n is negative, return 0. If n is positive or 0, return 1.

**Function prototype** (ruby)

##

##

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

##

##

## @param {Integer} param\_1

##

## @return {integer}

##

def my\_is\_negative(param\_1)

end

*Tip* (In Ruby) Your script will look like something close to this:

def my\_is\_negative(n) {

if (XXXXX)

return XXX;

else

return XXX;

end

end

puts my\_is\_negative(-1);

puts my\_is\_negative(1);

puts my\_is\_negative(0);

# puts my\_is\_negative(1337);

# REMEMBER WHEN YOU ARE FINISHED TO COMMENT ALL CALL TO YOUR

# FUNCTION my\_is\_negative function

# OTHERWISE IT WILL FAIL THE AUTOMATIC TEST SYSTEM

#

# <- yes this a way to comment your code

**My Abs**

* Submit directory: ex02
* Submit file: ["my\_abs.rb"]

Create a my\_abs function.

Reproduce behavior of an abs() function. It returns always the positive value of a number.

**Function prototype** (ruby)

##

##

## QWASAR.IO -- my\_abs

##

##

## @param {Integer} param\_1

##

## @return {integer}

##

def my\_abs(param\_1)

end

**Example 00**

Input: -30

Output:

Return Value: 30

**Example 01**

Input: 30

Output:

Return Value: 30

**Example 02**

Input: 0

Output:

Return Value: 0

**My Add**

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

Create a my\_add function which takes 2 parameters (nbr1 and nbr2) and returns a value. This value is the result of the addition of nbr1 and nbr2 parameters.

**Function prototype** (ruby)

##

##

## QWASAR.IO -- my\_add

##

##

## @param {Integer} param\_1

## @param {Integer} param\_2

##

## @return {integer}

##

def my\_add(param\_1, param\_2)

end

**Example 00**

Input: 0 && 1

Output:

Return Value: 1

**Example 01**

Input: 10 && 10

Output:

Return Value: 20

**Example 02**

Input: -10 && 10

Output:

Return Value: 0

**My Sub**

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

Create a my\_sub function which takes 2 parameters (nbr1 and nbr2) and returns a value. This value is the result of the subtraction of nbr1 and nbr2 parameters.

**Function prototype** (ruby)

##

##

## QWASAR.IO -- my\_sub

##

##

## @param {Integer} param\_1

## @param {Integer} param\_2

##

## @return {integer}

##

def my\_sub(param\_1, param\_2)

end

**Example 00**

Input: 0 && 1

Output:

Return Value: -1

**Example 01**

Input: 10 && 10

Output:

Return Value: 0

**Example 02**

Input: -10 && 10

Output:

Return Value: -20

**My Mult**

* Submit directory: ex05
* Submit file: ["my\_mult.rb"]

Create a my\_mult function which takes 2 parameters (nbr1 and nbr2) and returns a value. This value is the result of the multiplication of nbr1 and nbr2 parameters.

**Function prototype** (ruby)

##

##

## QWASAR.IO -- my\_mult

##

##

## @param {Integer} param\_1

## @param {Integer} param\_2

##

## @return {integer}

##

def my\_mult(param\_1, param\_2)

end

**Example 00**

Input: 0 && 1

Output:

Return Value: 0

**Example 01**

Input: 10 && 10

Output:

Return Value: 100

**Example 02**

Input: -10 && 10

Output:

Return Value: -100

**My String Formatting**

* Submit directory: ex06
* Submit file: ["my\_string\_formatting.rb"]

Create a my\_string\_formatting function which takes 3 parameters (firstname, lastname and age) and prints a string composed value.

Formatting should be: "Hello, my name is FIRSTNAME LASTNAME, I'm AGE."

There is a newline added.

**Function prototype** (ruby)

##

##

## QWASAR.IO -- my\_string\_formatting

##

##

## @param {String} param\_1

## @param {String} param\_2

## @param {Integer} param\_3

##

##

def my\_string\_formatting(param\_1, param\_2, param\_3)

end

**Example 00**

Input: "john" && "doe" && 37

Output: Hello, my name is john doe, I'm 37.

Return Value: nil

**Example 01**

Input: "Baby" && "Yoda" && 50

Output: Hello, my name is Baby Yoda, I'm 50.

Return Value: nil

**Example 02**

Input: "Marie" && "Curie" && 26

Output: Hello, my name is Marie Curie, I'm 26.

Return Value: nil

*Tip* You should use Google to learn about String interpolation :-)