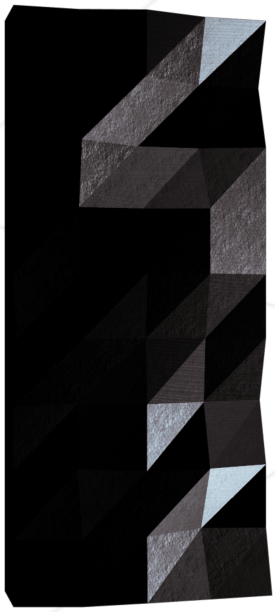

# Discovery Piscine

## Module7 - Python

*Summary: In this Module, we see how to use arrays and their associated functions.*

*Version: 1.00*

# Contents

# Chapter I

# A word about this Discovery Piscine

Welcome !
You will begin a Module of this Discovery Piscine of computer programming. Our goal is to introduce you to the code behind the software you use daily and immerse you in peer learning, the educational model of 42.

Programming is about logic, not mathematics. It gives you basic building blocks that you can assemble in countless ways. There is no single "correct" solution to a problem—your solution will be unique, just as each of your peers' solutions will be.

Fast or slow, elegant or messy, as long as it works, that's what matters! These building blocks will form a sequence of instructions (for calculations, displays, etc.) that the computer will execute in the order you design.

Instead of providing you with a course where each problem has only one solution, we place you in a peer-learning environment. You'll search for elements that could help you tackle your challenge, refine them through testing and experimentation, and ultimately create your own program. Discuss with others, share your perspectives, come up with new ideas together, and test everything yourself to ensure it works.

Peer evaluation is a key opportunity to discover alternative approaches and spot potential issues in your program that you may have missed (consider how frustrating a program crash can be). Each reviewer will approach your work differently—like clients with varying expectations—giving you fresh perspectives. You may even form connections for future collaborations.

By the end of this Piscine, your journey will be unique. You will have tackled different challenges, validated different projects, and chosen different paths than others—and that's perfectly fine! This is both a collective and individual experience, and everyone will gain something from it.

Good luck to all; we hope you enjoy this journey of discovery.

# Chapter II

# Introduction

What this Module will show you:

- You will learn how to handle arrays and their associated functions.

# Chapter III

# General instructions

Unless otherwise specified, the following rules apply every day of this Piscine.

- This document is the only trusted source. Do not rely on rumors.
- This document may be updated up to one hour before the submission deadline.
- Assignments must be completed in the specified order. Later assignments will not be evaluated unless all previous ones are completed correctly.
- Pay close attention to the access rights of your files and folders.
- Your assignments will be evaluated by your fellow Piscine peers.
- All shell assignments must run using `/bin/bash`.
- You must not leave any file in your submission workspace other than those explicitly requested by the assignments.
- Have a question? Ask your neighbor on your left. If not, try your neighbor on your right.
- Every technical answer you need can be found in the `man` pages or online.
- Remember to use the Piscine forum of your intranet and Slack!
- Read the examples thoroughly, as they may reveal requirements that aren't immediately obvious in the assignment description.
- By Thor, by Odin! Use your brain!!!

# Chapter IV

# Exercise 00: parameter_matching

|  | Exercise 00 |
|---|---|
| Identifying a Parameter | |
| Turn-in directory : *ex*00/ | |
| Files to turn in : `parameter_matching.py` | |
| Allowed functions : `All` | |

- Create a program called `parameter_matching.py`.
- Ensure this program is executable.
- The program should:
  - Take a string as a parameter.
  - Prompt the user to enter a word that matches the parameter passed as an argument, as shown in the example below.
  - Display "Good job!" followed by a newline if the word entered by the user matches the parameter passed.
  - Display "Nope, sorry..." followed by a newline if the word entered by the user does not match the parameter passed.
  - Display "none" followed by a newline if the number of parameters passed is different from 1.

```
?> ./parameter_matching.py
none
?> ./parameter_matching.py "Hello"
What was the parameter? Bonjour
Nope, sorry...
?> ./parameter_matching.py "Hello"
What was the parameter? Hello
Good job!
?>
```

Use one or more if-else statements.

# Chapter V

# Exercise 01: count_it

|  | Exercise 01 |
|---|---|
| Counting and Measuring Parameters | |
| Turn-in directory : $ex01/$ | |
| Files to turn in : `count_it.py` | |
| Allowed functions : `All` | |

- Create a program called `count_it.py`.
- Ensure this program is executable.
- The program should:
  - Display "parameters:" followed by the total number of parameters passed as arguments.
  - For each parameter, display the parameter itself and its length, ending with a newline.
  - If no parameters are provided, display "none" followed by a newline.

```
?> ./count_it.py | cat -e
none$
?> ./count_it.py "Game" "of" "Thrones" | cat -e
parameters: 3$
Game: 4$
of: 2$
Thrones: 7$
?>
```

This time, you must use a "for" loop instead of a "while" loop.

# Chapter VI

# Exercise 02: string_are_arrays

|  | Exercise 02 |
|---|---|
| Understanding Strings as Arrays | |
| Turn-in directory : *ex*02/ | |
| Files to turn in : `string_are_arrays.py` | |
| Allowed functions : `All` | |

- Create a program called `string_are_arrays.py` that takes a string as a parameter.
- This program should be executable.
- When executed, the program should display "z" for each occurrence of the character "z" in the string passed as a parameter, followed by a newline.
- If the number of parameters is different from 1, or if there are no "z" characters in the string, it should display "none" followed by a newline.

```
?> ./string_are_arrays.py | cat -e
none$
?> ./string_are_arrays.py "The character Z is not found in this string" | cat -e
none$
?> ./string_are_arrays.py "The character z is found in this string" | cat -e
z$
?> ./string_are_arrays.py "Zaz visits the zoo with Zazie" | cat -e
zzz$
?>
```

`Strings are also composed of individual characters, just like arrays. Give it a try!`

# Chapter VII

# Exercise 03: append_it

|  | Exercise 03 |
|---|---|
| Modifying Strings | |
| Turn-in directory : $ex03/$ | |
| Files to turn in : `append_it.py` | |
| Allowed functions : `All` | |

- Create a program called `append_it.py`.
- Ensure this program is executable.
- The program should display each parameter passed as an argument, one by one, by appending it with "ism".
- If a parameter already ends with "ism", it should be skipped and not displayed.
- If there are no parameters, the program should display "none" followed by a newline.

```
?> ./append_it.py | cat -e
none$
?> ./append_it.py "parallel" "egoism" "human" | cat -e
parallelism$
humanism$
?>
```

`Use matching.`

# Chapter VIII

# Exercise 04: free_range

|  | Exercise 04 |
|---|---|
| Just one step away! | |
| Turn-in directory : *ex*04/ | |
| Files to turn in : `free_range.py` | |
| Allowed functions : `All` | |

- Create a program called `free_range.py`.
- Ensure this program is executable.
- This program should take 2 numbers as parameters.
- You must construct an array containing all the values between these two numbers.
- You must display the array using the `print` function.
- If the number of parameters is different from 2, the program display `'none'` followed by a newline.

```
?> ./free_range.py | cat -e
none$
?> ./free_range.py 10 14 | cat -e
[10, 11, 12, 13, 14]$
?>
```

`Use the range function.`

# Chapter IX

# Submission and peer-evaluation

- You must have `discovery_piscine` folder at the root of your home directory.
- Inside the `discovery_piscine` folder, you must have a folder named `module7`.
- Inside the `module7` folder, you must have a folder for each exercise.
- Exercise 00 must be in the `ex00` folder, Exercise 01 in the `ex01` folder, etc.
- Each exercise folder must contain the files requested in the assignment.

`Please note, during your defense anything that is not present in the folder for the module will not be checked.`