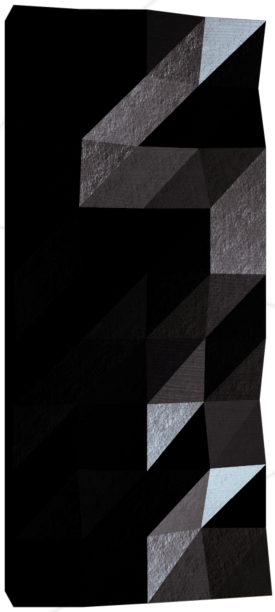

Discovery Piscine

Module6 - 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: Display First Parameter

	Exercise 00
Display a parameter	
Turn-in directory : *ex*00/	
Files to turn in : `aff_first_param.py`	
Allowed functions : `All`	

- Create a program called `aff_first_param.py`.
- Ensure this program is executable.
- The program should display the first string parameter passed, followed by a newline.
- If there are no parameters, display "none" followed by a newline.

```
?> ./aff_first_param.py | cat -e
none$
?> ./aff_first_param.py "Code Ninja" "Numeric" "42" | cat -e
Code Ninja$
?>
```

`Look up how to use "if" conditions in Python.`

Chapter V

Exercise 01: UPCASE_IT

	Exercise 01
Go Big	
Turn-in directory : $ex01/$	
Files to turn in : `upcase_it.py`	
Allowed functions : `All`	

- Create a program called `upcase_it.py` that takes a string as a parameter.
- Ensure this program is executable.
- The program should display the string in uppercase, followed by a newline.
- If the number of parameters is different from 1, display "none" followed by a newline.

```
?> ./upcase_it.py | cat -e
none$
?> ./upcase_it.py "initiation" | cat -e
INITIATION$
?> ./upcase_it.py 'This exercise is quite easy!' | cat -e
THIS EXERCISE IS QUITE EASY!$
?>
```

Chapter VI

Exercise 02: downcase_it

	Exercise 02
	Go Small
Turn-in directory : *ex*02/	
Files to turn in : `downcase_it.py`	
Allowed functions : `All`	

- Create a program called `downcase_it.py`.
- Ensure this program is executable.
- The program takes a string as a parameter.
- It should display the string in lowercase, followed by a newline.
- If the number of parameters is different from 1, it should display "none" followed by a newline.

```
?> ./downcase_it.py | cat -e
none$
?> ./downcase_it.py "FIREFLY" | cat -e
firefly$
?> ./downcase_it.py 'This exercise is quite easy!' | cat -e
this exercise is quite easy!$
?>
```

`This exercise should not take you more than 10 seconds.`

Chapter VII

Exercise 03: aff_rev_params

	Exercise 03
Display an Array in Reverse Order	
Turn-in directory : $ex03/$	
Files to turn in : `aff_rev_params.py`	
Allowed functions : `All`	

- Create a program called `aff_rev_params.py`.
- Ensure this program is executable.
- When executed, the program should display all the strings passed as parameters, followed by a newline, in reverse order.
- If there are fewer than two parameters, it should display `"none"` followed by a newline.

```
?> ./aff_rev_params.py | cat -e
none$
?> ./aff_rev_params.py "coucou" | cat -e
none$
?> ./aff_rev_params.py "Python" "piscine" "hello" | cat -e
hello$
piscine$
Python$
?>
```

Chapter VIII

Exercise 04: scan_it

	Exercise 04
Scanning a Text	
Turn-in directory : *ex*04/	
Files to turn in : `scan_it.py`	
Allowed functions : `All`	

- Create a program called `scan_it.py`.
- Ensure this program is executable.
- This program should take two parameters.
- The first parameter is a keyword to search for in a string.
- The second parameter is the string to be searched.
- When executed, the program should display the number of times the keyword appears in the string.
- If the number of parameters is different from 2 or if the first string does not appear in the second, it should display `none` followed by a newline.

```
?> ./scan_it.py | cat -e
none$
?> ./scan_it.py "the" | cat -e
none$
?> ./scan_it.py "the" "hello world" | cat -e
none$
?> ./scan_it.py "the" "the quick brown fox jumps over the lazy dog" | cat -e
2$
?>
```

You can use the ‘re.findall()‘ method in Python’s ‘re‘ module.

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 `module6`.
- Inside the `module6` 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.`