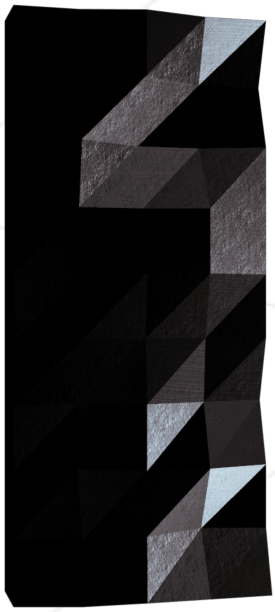

Discovery Piscine

Module8 - Python

Summary: In this Module, we see how to use methodes and scopes.

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 Discovering Methods!

	Exercise 00
Discovering Methods!	
Turn-in directory : *ex*00/	
Files to turn in : `hello_all.py`	
Allowed functions : `All`	

- Create a program called `hello_all.py`.
- Ensure this program is executable.
- This program should:
  - Define a method called `hello` that print "Hello, everyone!".
  - Call this method to display the message. Refer to the example below, except that the method definition has been hidden.

```
?> cat hello_all.py
# Your method definition

hello()
?> ./hello_all.py
Hello, everyone!
?>
```

`Search for "method definition in Python".`

Chapter V

Exercise 01: upcase_it

	Exercise 01
The Return of upcase!	
Turn-in directory : $ex01/$	
Files to turn in : `upcase_it.py`	
Allowed functions : `All`	

- Create a program called `upcase_it.py` (again!)
- Ensure this program is executable.
- Define a method in the program named `upcase_it`.
- The `upcase_it` method should take a string as an argument and return the string in uppercase.
- Test the method by calling it in your program. In the example below, we test it with "hello":

```
?> cat upcase_it.py
# Your method definition

print(upcase_it("hello"))
?> ./upcase_it.py
HELLO
?>
```

Chapter VI

Exercise 02: downcase_all

	Exercise 02
Let's Stroll in the Array	
Turn-in directory : *ex*02/	
Files to turn in : `downcase_all.py`	
Allowed functions : `All`	

- Create a program called `downcase_all.py`.
- Ensure this program is executable.
- Define a method in this program called `downcase_it`.
- The `downcase_it` method should take a string as an argument and return the string in lowercase.
- Apply this method to each program's parameters and display the return value for each.
- If there are no parameters, display "none" followed by a line break.

```
?> ./downcase_all.py
none
?> ./downcase_all.py "HELLO WORLD" "I understood Arrays well!"
hello world
i understood arrays well!
?>
```

Chapter VII

Exercise 03: greetings_for_all

	Exercise 03
Let's Say Hello to everyone!	
Turn-in directory : $ex03/$	
Files to turn in : `greetings_for_all.py`	
Allowed functions : `All`	

- Create a program called `greetings_for_all.py` that does not take any parameters.
- Ensure this program is executable.
- Create a method called `greetings` which takes a name as a parameter and displays a welcome message with that name.
- If the method is called without an argument, its default parameter should be "noble stranger".
- If the method is called with an argument that is not a string, an error message should be displayed instead of the welcome message.

```
?> cat greetings_for_all.py | cat -e
# your method definition here

greetings('Alexandra')
greetings('Wil')
greetings()
greetings(42)
```

will produce the following output:

```
?> ./greetings_for_all.py | cat -e
Hello, Alexandra.$
Hello, Wil.$
Hello, noble stranger.$
Error! It was not a name.$
?>
```

`Google default parameter, isinstance method.`

Chapter VIII

Exercise 04: methods_everywhere

	Exercise 04
Methods Everywhere!	
Turn-in directory : $ex04/$	
Files to turn in : `methods_everywhere.py`	
Allowed functions : `All`	

- Create a program called`methods_everywhere.py` that takes parameters.
- Ensure this program is executable.
- You need to create two different methods in this program:
  - The `shrink` method: Should take a string as a parameter and displays the first eight characters of that string.
  - The `enlarge` method: Should take a string as a parameter and appends 'Z' characters until the string has a total of eight characters. Then, it displays the resulting string.
- For each argument passed to the program:
  - If the argument has more than eight characters, call the `shrink` method on it.
  - If the argument has less than eight characters, call the `enlarge` method on it.
  - If the argument has exactly eight characters, display it directly followed by a new line.

```
?> ./methods_everywhere.py | cat -e
none$
?> ./methods_everywhere.py 'lol' 'physically' 'backpack' | cat -e
lolZZZZZ$
physical$
backpack$
?>
```

`shrink method: Use slices.`

`enlarge method: Similar to arrays, you can add characters to a string using the concatenation operator`

Chapter IX

Exercise 05: scope_that

	Exercise 05
Can't touch this!	
Turn-in directory : $ex05/$	
Files to turn in : `scope_that.py`	
Allowed functions : `All`	

- Create a program called `scope_that.py` that does not take any parameters.
- Ensure this program is executable.
- Inside the program, create a method called `add_one` that takes a parameter and adds 1 to the parameter.
- Initialize a variable in the body of the program, display it, and then call the method that adds 1.
- Display your variable again in the body of the program.
- What do you observe?

Chapter X

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 `module8`.
- Inside the `module8` 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.`