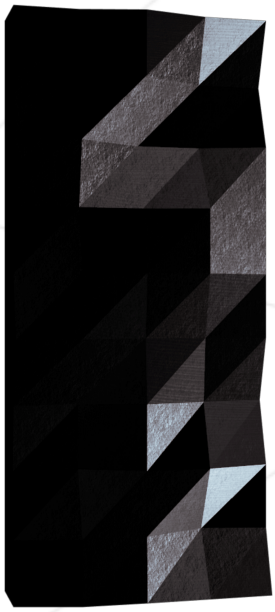

Discovery Piscine

Module9 - Python

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

Version: 1.00

Contents

I	A word about this Discovery Piscine	2
II	Introduction	3
III	General instructions	4
IV	Exercise 00: your_namebook	5
V	Exercise 01: family_affairs	7
VI	Exercise 02: help_your_professor	9
VII	Exercise 03: persons_of_interest	10
VIII	Submission and peer-evaluation	12

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: your_namebook

	Exercise 00
Organize this list of names for me!	
Turn-in directory : $ex00/$	
Files to turn in : `your_namebook.py`	
Allowed functions : `All`	

- Create a script called `your_namebook.py`.
- This script should contain a method called `array_of_names`.
- This method should take a dictionary that associates first names with last names as its parameter.
- It should build an array containing the full names of the people, with the first letter of each name capitalized. The methode should then return this array. Refer to the example.
- For example, the following script:

```
# your method definition here

persons = {
    "jean": "valjean",
    "grace": "hopper",
    "xavier": "niel",
    "fifi": "brindacier"
}

print(array_of_names(persons))
```

Will produce the following output:

```
?> ./your_namebook.py
['Jean Valjean', 'Grace Hopper', 'Xavier Niel', 'Fifi Brindacier']
?>
```

You can use the capitalize method in Python.

Chapter V

Exercise 01: family_affairs

	Exercise 01
Family Matters	
Turn-in directory : $ex01/$	
Files to turn in : `family_affairs.py`	
Allowed functions : `All`	

- Create a script called `family_affairs.py`.
- This script should contain a method called `find_the_redheads`.
- This method will take a dictionary as a parameter, where the kay are family members' first names and the values are their hair colors.
- The method should use the `filter` function to collect the first names of individuals with red hair into a new list, which it will return.
- For example, the following script:

```
# your method definition here
dupont_family = {
    "florian": "red",
    "marie": "blond",
    "virginie": "brunette",
    "david": "red",
    "franck": "red"
}

print(find_the_redheads(dupont_family))
```

Will produce the following output:

```
?> ./family_affairs.py
['florian', 'david', 'franck']
?>
```

`Google Python dictionary filter, keys(), tolist()`

Chapter VI

Exercise 02: help_your_professor

	Exercise 02
Give Your Professor a Helping Hand	
Turn-in directory : $ex02/$	
Files to turn in : `help_your_professor.py`	
Allowed functions : `All`	

- Create a script called `help_your_professor.py`.
- The script should contain a method called `average`.
- This method should take a dictionary as a parameter, where the keys are the students' first names and the values their scores on an assignment. The methode should then calculate the class average for that assignment.
- For example, the following script:

```
# your method definition here
class_3B = {
    "marine": 18,
    "jean": 15,
    "coline": 8,
    "luc": 9
}
class_3C = {
    "quentin": 17,
    "julie": 15,
    "marc": 8,
    "stephanie": 13
}

print(f"Average for class 3B: {average(class_3B)}.")
print(f"Average for class 3C: {average(class_3C)}.")
```

Will produce the following output:

```
?> ./help_your_professor.py
Average for class 3B: 12.5.
Average for class 3C: 13.25.
?>
```

Chapter VII

Exercise 03: persons_of_interest

	Exercise 03
People Worth Knowing	
Turn-in directory : $ex03/$	
Files to turn in : `persons_of_interest.py`	
Allowed functions : `All`	

- Create a script called `persons_of_interest.py`.
- The script should contain a method called `famous_births`.
- This method should take a dictionary as a parameter, representing historical figures. Each entry in the dictionary is itself a dictionary with the keys: `:name` and `:date_of_birth`.
- The method should sort the dictionary by birth dates and then display each entry (see the example below).
- For example, the following script:

```
# your method definition here

women_scientists = {
    "ada": { "name": "Ada Lovelace", "date_of_birth": "1815" },
    "cecilia": { "name": "Cecila Payne", "date_of_birth": "1900" },
    "lise": { "name": "Lise Meitner", "date_of_birth": "1878" },
    "grace": { "name": "Grace Hopper", "date_of_birth": "1906" }
}

famous_births(women_scientists)
```

will produce the following output:

```
?> ./persons_of_interest.py
Ada Lovelace is a great scientist born in 1815.
Lise Meitner is a great scientist born in 1878.
Cecila Payne is a great scientist born in 1900.
Grace Hopper is a great scientist born in 1906.
?>
```

Google python dictionary & sorted

You can also google the aforementioned names, they are worth learning about!

Chapter VIII

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 `module9`.
- Inside the `module9` 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.`