

Piscine Python

Day02 - Basics 3

Description of day --

Notions of the day

Decorators, multiprocessing, lambda, build package, ...

General rules

- The norm : during this pool you will follow the Pep8 standards
<https://www.python.org/dev/peps/pep-0008/>
- Forbidden functions : eval, ...

Helper

How to install and link python in the \$PATH

```
export ....  
install ....
```

Exercise 00 - Map, filter, reduce.

Exercise 01 - args and kwargs ?

Exercise 02 - The logger.

Exercise 03 - Json issues

Exercise 04 - Package ??

Exercise 00 - Map, filter, reduce.

Turnin directory :	ex00
Files to turn in :	ft_map.py ft_filter.py ft_reduce.py
Forbidden function :	map filter reduce
Remarks :	n/a

Implement the higher order functions `map()`, `filter()` and `reduce()`. Take the time to understand the use case of these three built-in functions.

How they should be prototyped:

```
ft_map()  
ft_filter()  
ft_reduce()
```

Exercise 01 - args and kwargs ?

Turnin directory :	ex00
Files to turn in :	main.py
Forbidden function :	
Remarks :	n/a

Implement the `what_are_the_vars` function that returns an object with the right attributes.
Have a look to `getattr`, `setattr`.

```
def what_are_the_vars(*args, **kwargs):
    """Your code"""
    pass

class ObjectC(object):
    def __init__(self):
        pass

def doom_printer(obj):
    for attr in dir(obj):
        if attr[0] != '_':
            value = getattr(obj, attr)
            print("{}: {}".format(attr, value))
    print("end")

if __name__ == "__main__":
    obj = what_are_the_vars(7)
    doom_printer(obj)
    obj = what_are_the_vars("ft_lol", "Hi")
    doom_printer(obj)
    obj = what_are_the_vars()
    doom_printer(obj)
    obj = what_are_the_vars(12, "Yes", [0, 0, 0], a=10, hello="world")
    doom_printer(obj)
    obj = what_are_the_vars(42, a=10, var_0="world")
    doom_printer(obj)
```

output

```
>> python main.py
var_0: 7
```

```
end
var_0: ft_lol
var_1: Hi
end
end
a: 10
hello: world
var_0: 12
var_1: Yes
var_2: [0, 0, 0]
end
ERROR
end
```

Exercise 02 - The logger.

Turnin directory :	ex02
Files to turn in :	logger.py
Forbidden function :	
Remarks :	n/a

You are going to learn more advanced features in python.

In this exercise, I want you to learn about decorators, and I am not talking about the decoration of your room.

```
import time
from random import randint

class CoffeeMachine():

    water_level = 100

    @log
    def start_machine():
        if self.water_level > 20:
            return True
        else:
            print("Please add water!")
            return False

    @log
    def boil_water():
        return "boiling..."

    @log
    def make_coffee(self):
        if self.start_machine():
            for _ in range(20):
                time.sleep(0.1)
                self.water_level -= 1
            print(self.boil_water())
            print("Coffee is ready!")

    @log
    def add_water(self, water_level):
        time.sleep(randint(1, 5))
        self.water_level += water_level
```

```
if __name__ == "__main__":
    machine = CoffeeMachine()
    for i in range(0, 5):
        machine.make_coffee()

    machine.make_coffee()
    machine.add_water(70)
```

Terminal

```
boiling...
Coffee is ready!
boiling...
Coffee is ready!
boiling...
Coffee is ready!
boiling...
Coffee is ready!
Please add water!
Please add water!
Glouglouglou...
```

machine.log

```
(cmaxime)Running: Start Machine
(cmaxime)Running: Boil Water
(cmaxime)Running: Make Coffee
(cmaxime)Running: Start Machine
(cmaxime)Running: Boil Water
(cmaxime)Running: Make Coffee
(cmaxime)Running: Start Machine
(cmaxime)Running: Boil Water
(cmaxime)Running: Make Coffee
(cmaxime)Running: Start Machine
(cmaxime)Running: Boil Water
(cmaxime)Running: Make Coffee
(cmaxime)Running: Start Machine
(cmaxime)Running: Make Coffee
(cmaxime)Running: Start Machine
(cmaxime)Running: Make Coffee
(cmaxime)Running: Add Water
[ exec-time = 0.001 ms ]
[ exec-time = 0.005 ms ]
[ exec-time = 2.499 s ]
[ exec-time = 0.002 ms ]
[ exec-time = 0.005 ms ]
[ exec-time = 2.618 s ]
[ exec-time = 0.003 ms ]
[ exec-time = 0.004 ms ]
[ exec-time = 2.676 s ]
[ exec-time = 0.003 ms ]
[ exec-time = 0.004 ms ]
[ exec-time = 2.648 s ]
[ exec-time = 0.011 ms ]
[ exec-time = 0.029 ms ]
[ exec-time = 0.009 ms ]
[ exec-time = 0.024 ms ]
[ exec-time = 5.026 s ]
```

Exercise 03 - Json issues

Turnin directory :	ex00
Files to turn in :	json_reader.py
Forbidden function :	json, eval
Remarks :	n/a

It's the context manager that will help you to handle this task.

Implement a `Loadjson` class that open, read, and parse a json file, store it in a data attribute as a nested dict, and close the file at the end of the usage.
you will also have to code the function `print_formated`.

```
>> cat list.json
{"quiz": {"sport": {"q1": {"question": "Which one is correct team name in NBA?", "option": ["New York Bulls", "Los Angeles Kings", "Golden State Warriros"]}}}
```

tester.py

```
from json_reader import loadjson, print_formated

if __name__ == "__main__":
    with Loadjson('list.json') as js:
        data = js.getdata()
        print_formated(data)
```

Output

```
>> python tester.py
"quiz": {
    "sport": {
        "q1": {
            "question": "Which one is correct team name in NBA?",
            "options": [
                "New York Bulls",
                "Los Angeles Kings",
                "Golden State Warriros",
```

```
        "Huston Rocket"
    ],
    "answer": "Huston Rocket"
}
},
"maths": {
    "q1": {
        "question": "5 + 7 = ?",
        "options": [
            "10",
            "11",
            "12",
            "13"
        ],
        "answer": "12"
    },
    "q2": {
        "question": "12 - 8 = ?",
        "options": [
            "1",
            "2",
            "3",
            "4"
        ],
        "answer": "4"
    }
}
```

Exercise 04 - MiniPack

Turnin directory :	ex00
Files to turn in :	build.sh, *.py
Forbidden function :	
Remarks :	n/a

You have to create a package called `42ai`.

I will have 2 functionalities:

- the progress bar (day00 ex09), that can be imported via `import 42ai.progressbar`,
- the logger (day02 ex02) `import 42ai.logging.log`,

The package will be installed via pip using the following command : `bash build.sh && pip install 42ai-1.0.0.tar.gz`

The build.sh script has to create the `42ai-1.0.0.tar.gz` file.