Piscine Python

Day00 - Basic stuff - 11th Commandments

The goal of the day is to get started with the python language.

Notions of the day

Basic setup, variables, types, functions, …

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? It's the first exercise!

```
Not helper there
```

Exercise 00 - $PATH.

Exercise 01 - Rev Alpha.

Exercise 02 - The Odd, the Even and the Zero.

Exercise 03 - Functional file.

Exercise 04 - Return the functional part.

Exercise 05 - The right format.

Exercise 06 - A recipe.

Exercise 07 - Shorter, faster, pythonest.

Exercise 08 - S.O.S.

Exercise 09 - Secret number.

Exercise 10 - Loading bar !

Exercise 00 - $PATH

Turnin directory :	ex00
Files to turn in :	installer.sh
Forbidden function :	None
Remarks :	n/a

We noticed that the students had trouble with the setup of their python path. You are going to master python path !

In order to have a simple life lets create a script the check if the right python version is installed and install it if it is not, you are going to use miniconda installer : .
You will also be able to install packages via `pip` using this script.

```
$> ./installer.sh install-python
[v] python is installed.
$> ./installer.sh install-python
Python is already installed, do you want to reinstall it ?
[yes|no]> yes
[x] python has been removed.
[v] python is installed.
$> ./installer.sh install-python
Python is already installed, do you want to reinstall it ?
[yes|no]> no
exit.
```

The python folder will have to be installed in the `/goinfre` to save space on your session.

Exercise 01 - Rev Alpha

Turnin directory :	ex01
Files to turn in :	exec.py
Forbidden function :	None
Remarks :	n/a

You will have to make a program that reverses the order of a string and the case of its words.

```
$> python exec.py "Hello World!"
!DLROw OLLEw
$> python exec.py "Hello" "my Friend"
DNEIRf YM OLLEw
$> python exec.py
$>
```

Exercise 02 - The Odd, the Even and the Zero.

Turnin directory :	ex02
Files to turn in :	whois.py
Forbidden function :	None
Remarks :	n/a

You will have to make a program that checks if a number is odd, even or zero.
The program will accept only one parameter, an integer.

```
$> python whois.py 12
I'm Even.
$> python whois.py 3
I'm Odd.
$> python whois.py
$> python whois.py 0
I'm Zero.
$> python whois.py Hello
ERROR
$> python whois.py 12 3
ERROR
```

Exercise 03 - Functional file.

Turnin directory :	ex03
Files to turn in :	count.py
Forbidden function :	None
Remarks :	n/a

Create a function called `text_analyzer` that displays the sums of upper characters, lower characters, punctuation characters and spaces in a given text.

`text_analyzer` will take one parameter: the text to analyse. Handle the case where it is empty (maybe use a default value). If there is no text passed to the function, the user is prompted to give one.

Test it in the python console:

```
$> python
>>> from count import text_analyzer
>>> text_analyzer("Python 2.0, released 2000, introduced features like List compreh
The text contains 143 characters:
- 2 upper letters
- 113 lower letters
- 4 punctuation marks
- 18 spaces
>>> text_analyzer("Python is an interpreted, high-level, general-purpose programmin
The text contains 234 characters:
- 5 upper letters
- 187 lower letters
- 8 punctuation marks
- 30 spaces
>>> text_analyzer()
What is the text to analyse?
>> Python is an interpreted, high-level, general-purpose programming language. Crea
The text contains 234 characters:
- 5 upper letters
- 187 lower letters
- 8 punctuation marks
- 30 spaces
```

You're free to format your docstring and write it as you want to.

```
>>> print(text_analyzer.__doc__)

    This function counts the number of upper characters, lower characters,
    punctuation and spaces in a given text.
```

Exercise 04 - Return the functional part.

Turnin directory :	ex04
Files to turn in :	operations.py
Forbidden function :	None
Remarks :	n/a

You will have to make a program that prints the results of the four elementary mathematical operations of arithmetic (addition, subtraction, multiplication, division) and the modulo operation. First, start by writing a function that takes 2 numbers as parameters and returns 5 values.

```
$> python operations.py 10 3
Sum:         13
Difference:  7
Product:     30
Quotient:    3.3333333333333335
Remainder:   1
$> python operations.py 42 10
Sum:         52
Difference:  32
Product:     420
Quotient:    4.2
Remainder:   2
$> python operations.py
Usage: python operations.py <number1> <number2>
Example: python operations.py 10 3
```

Exercise 05 - The right format.

Turnin directory :	ex05
Files to turn in :	kata00.py kata01.py kata02.py kata03.py kata04.py
Forbidden function :	None
Remarks :	n/a

Let's get familiar with the useful concept of **string formatting** through a kata series.

kata00

```
t = (19,42,21)
```

Including the tuple above in your file, write a program that dynamically builds up a formatted string like the following:

```
$> python kata00.py
The 3 numbers are: 19, 42, 21
```

kata01

```
languages = {
    'Python': 'Guido van Rossum',
    'Ruby': 'Yukihiro Matsumoto',
    'PHP': 'Rasmus Lerdorf',
    }
```

Using the `languages` dictionary above, a similar exercise:

```
$> python kata01.py
Python was created by Guido van Rossum
Ruby was created by Yukihiro Matsumoto
PHP was created by Rasmus Lerdorf
```

kata02

```
(3,30,2019,9,25)
```

Given the tuple above, whose values stand for: `(hour, minutes, year, month, day)`, write a program that prints it as the following format:

```
$> python kata02.py
09/25/2019 03:30
```

kata03

```
phrase = "The right format"
```

Write a program to display the string above right-aligned with '-' padding and a total length of 42 characters:

```
$> python kata03.py
--------------------------The right format
```

kata04

```
( 0, 4, 132.42222, 10000, 12345.67)
```

Given the tuple above, get the following result:

```
$> python kata04.py
day_00, ex_04 :   132.42, 1.00e+04, 1.23e+04
```

Exercise 06 - A recipe.

Turnin directory :	ex06
Files to turn in :	recipe.py
Forbidden function :	None
Remarks :	n/a

This is time to discover Python dictionaries. Dictionaries are collection that contain mappings of unique keys to values.
Hint: check what is a nested dictionary in Python.

First, you will have to create a cookbook dictionary called `cookbook`.

`cookbook` will store 3 recipes:

- sandwich
- cake
- salad

Each recipe will store 3 values:

- ingredients: a **list** of ingredients
- meal: type of meal
- prep_time: preparation time in seconds

Sandwich's ingredients are *ham*, *bread*, *cheese* and *tomatoes*. This is a *lunch* and it takes *10* minutes of preparation.
Cake's ingredients are *flour*, *sugar* and *eggs*. This is a *dessert* and it takes *60* minutes of preparation.
Salad's ingredients are *avocado*, *arugula*, *tomatoes* and *spinach*. This is a *lunch* and it takes *15* minutes of preparation.

1. Get to know dictionaries. In the first place, try to print only the `keys` of the dictionary. Then only the `values`. And to conclude, all the `items`.

2. Write a function to print a recipe from `cookbook`. The function parameter will be: name of the recipe.

3. Write a function to delete a recipe from the dictionary. The function parameter will be: name of the recipe.

4. Write a function to add a new recipe to `cookbook` with its ingredients, its meal type and its preparation time. The function parameters will be: name of recipe, ingredients, meal and prep_time.

5. Write a function to print all recipe names from `cookbook`. Think about formatting the output.

6. Last but not least, write a program using the four functions you just wrote.
   The program will prompt the user to make a choice between printing the cookbook, printing only one recipe, adding a recipe, deleting a recipe or quitting the cookbook.

It could look like the example below but feel free to organize it as you wish:

```
$> python recipe.py
Please select an option by typing the corresponding number:
1: Add a recipe
2: Delete a recipe
3: Print a recipe
4: Print the cookbook
5: Quit
>> 3

Please enter the recipe's name to get details:
>> cake

Recipe for cake:
Ingredients list: ['flour', 'sugar', 'eggs']
To be eaten for dessert.
Takes 60 minutes of cooking.
```

Exercise 07 - Shorter, faster, pythonest.

Turnin directory :	ex07
Files to turn in :	filterwords.py
Forbidden function :	filter, len
Remarks :	n/a

Using list comprehension, you will have to make a program that removes all the words in a string that are shorter than or equal to n letters and returns the filtered list with no punctuation.
For loops are forbidden.
The program will accept only two parameters: a string and n, an integer.

```
$> python filterwords.py "Hello, my friend" 3
['Hello', 'friend']
$>  python filterwords.py "A robot must protect its own existence as long as such p
['protect', 'existence', 'protection', 'conflict']
$> python filterwords.py Hello World
ERROR
$> python filterwords.py 300 3
ERROR
```

Exercise 08 - S.O.S.

Turnin directory :	ex08
Files to turn in :	sos.py
Forbidden function :	None
Remarks :	n/a

You will have to make a morse encryption function.
The input will accept all alpha numeric caracteres.

```
$> python sos.py "SOS"
... ––– ...
$> python sos.py
$> python sos.py "HELLO / WORLD"
ERROR
$> python sos.py "96 BOULEVARD" "Bessiere"
––––. –.... / –... ––– ..– .–.. . ...– .– .–. –.. / –... . ... ... .. . .–. .
```

@ref: https://morsecode.scphillips.com/morse.html

Exercise 09 - Secret number.

Turnin directory :	ex09
Files to turn in :	guess.py
Forbidden function :	None
Remarks :	n/a

You will have to make a program that will be an interactive guessing game. It will ask the user to guess a number between 1 and 99. The program will tell the user if his/her input is too high or too low. The game ends when the user finds out the secret number or types `exit`.

You will have to import the `random` module with the `randint` function to get a random number.
You have to count of the number of trials and print them when the user wins.

```
$> python guess.py
This is an interactive guessing game!
You have to enter a number between 1 and 99 to find out the secret number.
Type 'exit' to end the game.
Good luck!

What's your guess between 1 and 99?
>> 54
Too high!
What's your guess between 1 and 99?
>> 34
Too low!
What's your guess between 1 and 99?
>> 45
Too high!
What's your guess between 1 and 99?
>> A
That's not a number.
What's your guess between 1 and 99?
>> 43
Congratulations, you've got it!
You won in 5 attempts!
```

If the user discovers the secret number on the first try, tell him/her.
Bonus: if the secret number is 42, make a reference to Douglas Adams.

```
$> python guess.py
What's your guess between 1 and 99?
>> 42
The answer to the ultimate question of life, the universe and everything is 42.
Congratulations! You got it at the first try!
```

Exercise 10 - Loading bar !

Turnin directory :	ex10
Files to turn in :	loading.py
Forbidden function :	None
Remarks :	n/a

This is a bonus exercice ! You are about to discover the `yield` operator !
So let's create a function called `ft_progress(list)`.

The function will display the progress of a `for` loop.

```
list = range(1000)
for elem in ft_progress(list):
    sleep(0.01)
```

```
$> python script.py
ETA: 8.67s [ 23%][=====>                ] 233/1000 | elapsed time 2.33s
```