Python Course

Class 2 : Data Structures and functions

Presented by : Sofiane MEDJKOUNE
Email : sofiane.medjkoune@protonmail.com

Functions

- Functions group a certain set of instructions so that we can use it anywhere in our code without having to write that set of instructions again. For example if you wanna get the results of a second degree equation you want need to write to calculate the delta and the 2 solutions each time, we'll be writing a function that returns the solutions and we'll use it whenever needed.
- We have seen a certain number of functions already : print(), input(), range().
- A function is made of :
  - Identifier (follows the same rules as variables for example : my_function_id())
  - Parameters : are variables that we give to the function, for example to calculate a second degree equation $ax^2 + bx + c = 0$ we'll be giving the parameters (a, b, c), to call the function we'll use : second_degree_equation(a, b, c).
  - a set of instructions
  - a return statement, as we said, a function can return a value (this is optional)

How to create a function ?

- The basic syntax for the functions is :

```
def function_identifier (parameter1, parameter2,...):
        Set of instructions
        return something (if you want to)
```

- You are free to give one, two, three, n parameters (or none) depending on your needs ! the parameters must be separated by a “,”. This parameter can be any type, int, float, Boolean, string, list...etc
- *DON’T FORGET THE TWO POINTS “:” AND THE INDENTATION*, the set of instructions is contained in your function so you need to respect the indentation as we’ve seen in conditions and loops.
- The return instruction is optional, for example if your function prints a person’s informations, you don’t return anything so no need for the return (not recommended though).

Function parameters

- As we said previously a function can have a certain number of parameters let’s have an example :

```
def body_mass_index(weight, height):
        BMI = weight / (height * height)
        return BMI
```

- We can also define a default value for a parameter, for example i wanna print the multiplication table for 6, generally we print from 1x6 to 10x6, it will suffice to some users but other users will want to print all the way to 20x6, what to do to satisfy both of them ? USE A DEFAULT PARAMETER

```
def mult_table(value,max=10):
        for i in range(1,max+1):
                print(“ix”,value,”=”,(i*value))
```

Lists

- In python a ***list*** is an ***object*** that can contain other variables of any type, this is how it goes :
  - declaration : list_identifier = [] #you can also write = list()
  - you can add elements to your list at the creation : my_first_list = [12, -6, True, “yes even a string”, [“another list baby !”]]
- To ***add*** elements to your list **after** you declare it, this is what to do :

  ```
  my_second_list = []
  my_second_list.append(your element)
  ```
- To access an element of your ***list*** you can use something you’ve seen before in other languages : my_first_list[2], and this will print True .
- You can modify an element of the list this way : my_first_list[2] = False
- There are a huge number of functions (methods) that are used to : add elements, modify, delete, sort...etc and can all be found in the official documentation (it’s really important to learn to access and read documentations). https://docs.python.org/3/tutorial/datastructures.html

Tuples

- A ***tuple*** is an element that resembles a list, only difference, a set is immutable (you can’t change it’s values once created)
- Creating a ***tuple*** is as easy as creating a list : my_set_identifier = ()
- You can add values to the set this way :
  - my_first_set = (1.83, 79)
  - any_other_set = (1, 4, 7, 8)
  - You can create a **tuple** as result of an ***affectation*** after a ***function return***

Practice !

In this practice, we’ll be taking the BMI example. Get the height and weight of five people, send them to a function the will return a list of each person’s BMI and send that list to another function that will do a quick classification (skinny, well shaped, fat...etc)