Session 5

Pepe García

September 11, 2020

In this async session we will learn about functions and then we will have some exercises to do in order to put in practice all stuff we have learnt so far.

There is a forum open in Campus, if you feel stuck with anything and would like to get some help, please post there. I will check in several times during the day to answer questions that may pop. Please, also consider that I will prioritize questions in the forum before questions you send directly to my email.

If, after reading this handout you feel you want to deepen your knowledge of Python in general, or functions in particular, you can try Python for Everybody [1].

[1] Charles Russell Severance. Python for everybody. `https://py4e.com`

Finally, please send over to my email your solutions to the exercises, they will be considered as homework. Use a subject line like follows:

```
Solutions exercise 5 - <name_and_last_name>
```

If I had to send it i'd do it with a subject line like this:

```
Solutions exercise 5 - Pepe García
```

Do not upload these solutions to Campus as assignments, just email them to me.

Functions

Functions are groups of Python statements that have a name and can be executed on demand. They will behave as black boxes to which we will be able to pass data in the form of **parameters** and that will give us output in the form of **return values**.

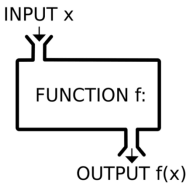

Figure 1: Programming functions are similar to mathematical functions, they have a domain (`x` in our example), and a codomain (`f(x)`), and they map values from the domain to values in the codomain.

Calling functions

The syntax for calling functions is the following:

```
function_name(parameter1, parameter2, parameterN)
```

When naming functions we will need to apply the same naming rules as for variables.

We have already seen some functions, such as **print()**, **type()**, **str()**, etc. We used them as follows:

```
type('hello')
str(3)
int(True)
```

Notice that so far, we have used all these functions by passing only one argument, but there are others that can receive more than one argument.

Declaring functions

We can declare our own functions using the **def** keyword with the following syntax:

```
def function_name(parameter1, parameter2):
    #<body>
```

When creating a function we need to indent the body to tell Python what piece of code we want to include inside the function, as we did with **if** statements.

Returning values from functions

Functions in Python can return values after doing all the operations they perform. In order to return something from a function, we need to use the **return** keyword and then pass the value we want to return. Here is a simple example of a function that always returns my name:

```
def return_my_name():
    return 'Pepe'
```

Function Parameters

Parameters are values that are injected to the function body when we call it. We declare the parameters between the parentheses in the function definition.

Here's an example of a function used to calculate the area of a square, that receives a parameter called side, and returns the area.

```
def area_square(side):
    return side * side
```

Exercises

exercise 1

Create a function `weekly_commute_time` that asks the user their daily commute time and returns their weekly time spent commuting.

exercise 2

What do the following expressions return?

- `True or 11 > 34`
- `False and (1 == 1)`
- `(77 // 11) > 6 and False`

exercise 3

Create a function `area_triangle` that takes the base and height of a triangle and returns its area

exercise 4

Create function `area_triangle_rectangle` that takes the base, height, and the kind of shape and calculates its area. It should work for both triangles and rectangles.

exercise 5

Create a function `im_in_love` that takes a weekday number (from monday to friday), and returns how that weekday is (according to The Cure!)[2]:

```
I don't care if Monday's blue
Tuesday's grey and Wednesday too
Thursday I don't care about you
It's Friday, I'm in love
```

[2] Find some inspiration for this exercise listening to them `https://www.youtube.com/watch?v=mGgMZpGYiy8`

References

Charles Russell Severance. Python for everybody. `https://py4e.com`.