**Python Functions**

**Overview**

|  |
| --- |
| Functions in Python provide organized, reusable and modular code to perform a set of specific actions. Functions simplify the coding process, prevent redundant login, and make the code easier to follow.  Python has many built-in functions like **print**(), **input**(), **len**(). Besides built-ins you can also create your own functions to do more specific jobs - these are called *user-defined functions*. |

**Defining and calling simple functions**

Using the ***def*** statement is the most common way to define a function in python. This statement is a so called single compound statement with the following syntax:



* ***function\_name***is known as the identifier of the function.
* ***parameters*** is an optional list of identifiers that get bound to the values supplied as arguments when the function is called.

Here is an example of a simple function definition which purpose is to print **Hello** each time it’s called:

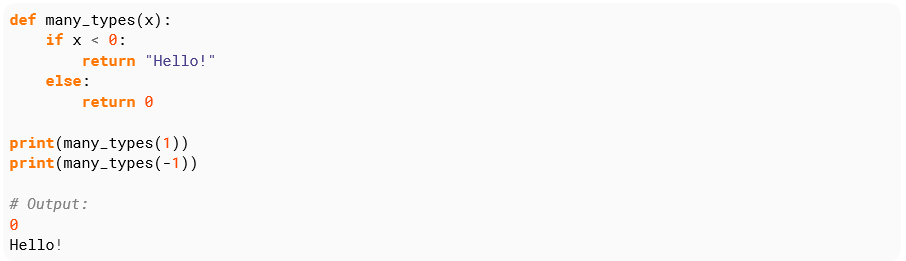


After declaring a function, it has to be called first in order to see the message or results. Not let’s call the defined ***greet()*** function*.*



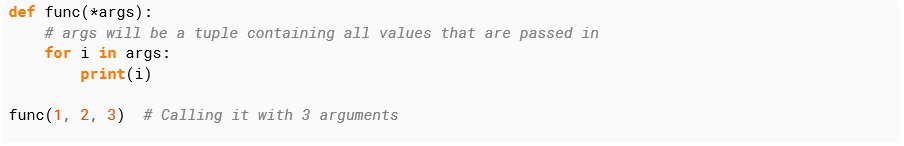
You’ll notice that unlike many other languages, you do not need to explicitly declare a return type of the function. Python functions can return values of any type via the **return** keyword.

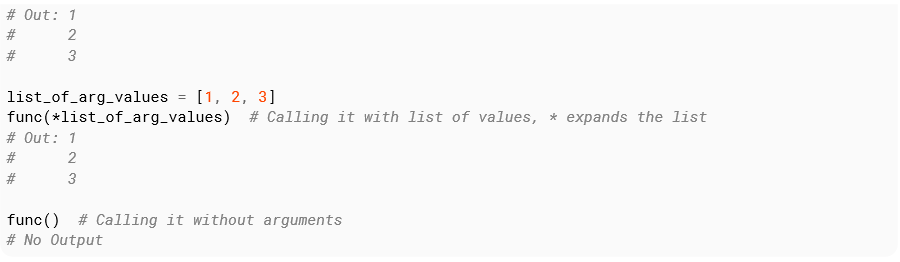
One function can return any number of different types!



**Function with an arbitrary number of arguments**

Defining a function capable of taking an arbitrary number of arguments can be done by prefixing one of the arguments with \*

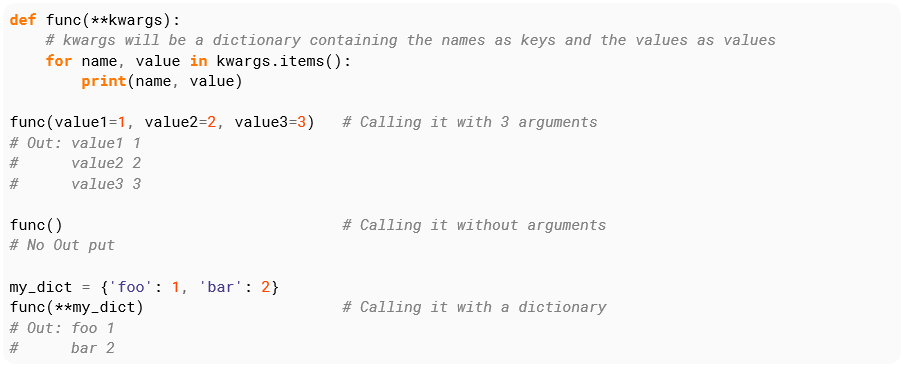




You **can’t** provide provide a default for args, for example ***func(\*args[1, 2, 3])*** will raise a syntax error. You can not provide args by name when calling the function, for example ***fun(\*args=[1,2,3])***. These arguments *(\*args)* can be accessed by index, for example *args[0]* will return the first argument.

**Arbitrary number of keyword arguments.**

You can take an arbitrary number of arguments with a name by defining an argument in the definition with **two \*\*** in front of it:



You can’t provide these without names, for example **fun(1 ,2, 3)** will raise a **TypeError.**

kwargs is a plain native python dictionary. For example, args[‘value1’] will give the value for argument value1. Be sure to check beforehand that there is such an argument or a KeyError will be raised.

**Task - Compulsory**

Let’s use functions to calculate your trip’s costs:

* Define a function called **hotel\_cost** with one argument **nights** as input. The hotel costs R140 per night. So, the function **hotel\_cost** should return 140 \* nights.
* Define a function called **plane\_ride\_cos**t that takes a string, **city**, as input. The function should return a different price depending on the location. Below are the valid destinations and their corresponding round-trip prices.

“Cape Town”: 2500

“Durban”: 2300

“JHB”: 2000

“BFN” 1800

* Below your existing code, define a function called **rental\_car\_cost** with an argument called days. Calculate the cost of renting the car: Every day you rent the car costs R40 pep day. If you rent the car for 7 or more days, you get R50 off your total(cost-=50). Alternatively (elif), if you rent the car for 3 or more days, you get R20 off your total. You cannot get both of the above discounts. Return that cost.
* Then, define a function called **trip\_cost** that takes two arguments, city and days. Like the example above, have your function return the sum of calling the **rental\_car\_cost**(days), **hotel\_cost**(days), and **plane\_ride\_cost**(city) functions.
* Modify your trip\_cost function definition. Add a third argument, spending\_money. Modify what the **trip\_cost** function does. Add the variable **‘spending\_money’** to the sum that it returns.