

# **Python - Functions and Modules**

## **Notebook**

## What is a function in Python?

A function is a group of related statements that performs a specific task. Functions help break our program into smaller and modular chunks. As our program grows larger and larger, functions make it more organized and manageable. Furthermore, it avoids repetition and makes the code reusable.

Once a function is created, the details of how it works can almost be forgotten about. In this way the detail is abstracted, allowing the programmer to focus on the bigger picture. Once a programmer defines a function, they can then call it whenever they need it, simply using its name. Additionally, to work, the function will probably require some inputs or parameters, which are given to the function each time it is called.

Some functions, including `print`, can be called without any parameters. However, every function call must include the parentheses used to pass parameters, even if they are empty - For example built-in functions like `print()`.

### Defining new functions

Creating a new function is straightforward and is something you should consider doing for any task you're likely to want your program to do regularly.

For example, imagine you have a program in which you will often need to print out the contents of a list line by line. The code to do this might look like this:

```
# created list
target_list = ["193.3.19.156", "118.64.255.34", "112.85.42.128"]

# running this code more than once
for target in target_list:
    print(target)
```

This is only three lines of code, but if you will need to print lists regularly, you should create a function. To create a new function, you need to define it before it is used for the first time, usually at the start of your program (it makes them load first, python interpret files from the first line to the last and the load process works in the same order):

```
# Created function
def display_targets():
    for target in target_list:
        print(target)

# Created list
target_list = ["193.3.19.156", "118.64.255.34", "112.85.42.128"]

# Call the function to print the list contents
display_targets()
```

The function above is not include parameters - mostly when the purpose of the function is to print something this is the case.

When the function purpose is wider and we should use it for different lists we could append a parameter inside the function braces - parameter name could be anything, this name will behave like a variable for further data you specify.

```
# Created function with "mylist" parameter
# mylist will be a placeholder and a template variable
def display_list(mylist):
    for item in mylist:
        print(item)

target_list = ["193.3.19.156", "118.64.255.34", "112.85.42.128"]
number_list = [1, 2, 3, 4, 5]
wordlist = ["password", "123456", "qwertyty"]

# Call the function with different assignments
display_list(target_list)
display_list(number_list)
display_list(wordlist)
```

In the above example we could see that **mylist** parameter used as a placeholder for the further assignments - that's mean that we specify the parameter it will act in the function like **mylist** parameter.

### The return Statement

Note that as you're printing something in your function, you don't really need to return it. However, if you want to continue to work with the result of your function and try out some operations on it, you will need to use the return statement to return a value, such as a string, an integer, .... consider the following scenario where your function returns a string of a result.

For example, a function that return the first item in a list for further processing:

```
def firstin(mylist):
    return mylist[0]

target_list = ["193.3.19.156", "118.64.255.34", "112.85.42.128"]
number_list = [1, 2, 3, 4, 5]
wordlist = ["password", "123456", "qwertyty"]

print(f"The first target is {firstin(target_list)}")
```

Or more complicated scenario when your function needs to find the longest item in a list:

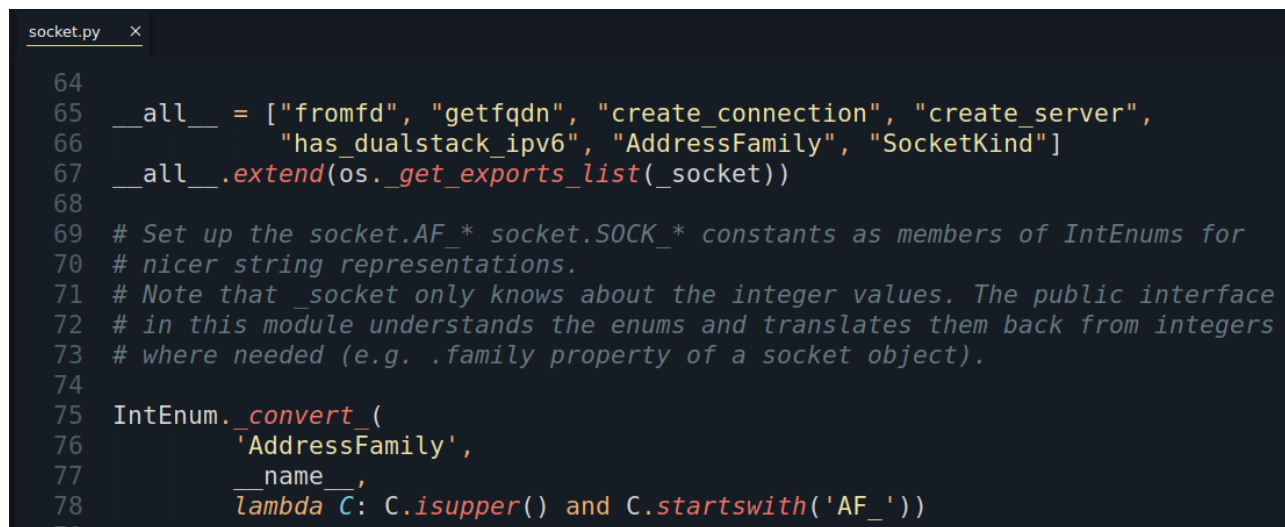
```
def longest(mylist):
    max_item = ""
    for i in mylist:
        if len(i) > len(max_item):
            max_item = i
    return max_item
```

```
wordlist = ["password", "123456", "qwertyty"]
print(f"The longest {longest(wordlist)}")
```

## What is a module in Python?

A module is simply a Python file with a **.py** extension that can be imported inside another Python program. The name of the Python file becomes the module name.

You could find the **module** files locally on the system and sometimes read the comments could clear things better than google:



```
socket.py x
64
65 __all__ = ["fromfd", "getfqdn", "create_connection", "create_server",
66           "has_dualstack_ipv6", "AddressFamily", "SocketKind"]
67 __all__.extend(os._get_exports_list(_socket))
68
69 # Set up the socket.AF_* socket.SOCK_* constants as members of IntEnums for
70 # nicer string representations.
71 # Note that _socket only knows about the integer values. The public interface
72 # in this module understands the enums and translates them back from integers
73 # where needed (e.g. .family property of a socket object).
74
75 IntEnum._convert_(
76     'AddressFamily',
77     __name__,
78     lambda C: C.isupper() and C.startswith('AF_'))
```

The module contains mostly:

- definitions and implementation of classes
- variables
- functions that can be used inside another program

Just think about the idea that every time that you would like to calculate some mathematical function with a **PI** you had to define its value, as you know the **PI** certainly exact value are very long and you could specify it in a better way using the **Math** library.

Other good example could come with the **random** library usage - when we would like to add some random functions that will randomize my list values or throw some random value we would have to work and prepare a special function with this desire, but instead, we just use a function within the **random** library.

## Importing Modules

Importing a module could happen only if the module is installed and exists in the default search path configured in the core of python (don't be afraid of that statement, it's just a wider explanation that you probably never modify).

We will talk later about how to install some modules, there are a lot that comes built in with python installation.

Importing modules works with **import** statement in the code followed by the name of the library.

```
import random
```

The function of the module is another learning process, there is a built-in function in python that could list you all the available functions and methods as well as short research for the certain function we search could be much better.

```
print(dir(random))

'''
['BPF', 'LOG4', 'NV_MAGICCONST', 'RECIP_BPF', 'Random', 'SG_MAGICCONST', 'SystemRandom', 'TWOPI', '_Sequence', '_Set', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', '_accumulate', '_acos', '_bisect', '_ceil', '_cos', '_e', '_exp', '_floor', '_inst', '_log', '_os', '_pi', '_random', '_repeat', '_sha512', '_sin', '_sqrt', '_test', '_test_generator', '_urandom', '_warn', 'betavariate', 'choice', 'choices', 'expovariate', 'gammavariate', 'gauss', 'getrandbits', 'getstate', 'lognormvariate', 'normalvariate', 'paretovariate', 'randbytes', 'randint', 'random', 'randrange', 'sample', 'seed', 'setstate', 'shuffle', 'triangular', 'uniform', 'vonmisesvariate', 'weibullvariate']
'''
```

Above is a full list of available functions you could use in your code when **random** imported. What you could see above divided into some parts that consists in every library (module):

- BPG,LOG4,etc - it's a special variable
- Everything started with underscores its objects and classes
- The others are the functions

When we call a function form an imported module, we should specify the name of **module** with dot and then the function name.

```
import random
print(random.randint(1,100))

# Output: 81
```

If you want to get help about the function you search, you could use the **help()** function

```
import random
print(help(random.randint))

'''
Help on method randint in module random:

randint(a, b) method of random.Random instance
    Return random integer in range [a, b], including both end points.
'''
```