Functions in Python

Functions are a fundamental part of Python (and most other programming languages). Python includes many built-in functions, but you can also create your own functions.

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function typically performs a specific task that you can run in your program. For example, print() is a built-in Python function that allows you to display text to the screen. Calling *print("Hello")* results in *Hello* being displayed.

One of the great things about functions is that they can be used like a "black box". You do not need to know how the function is coded in order to use it. In fact, you do not even need to see the function code before using it. All you need to know is how to call the function.

You call a function by referring to its name, and providing any arguments that it might require. So calling *print("Hello")* calls the *print()* function and passes an argument of *"Hello"*.

Creating a function

Here is an example of a basic function.

```
def greeting():
    print("This is a greeting function")
    print("I am still in the greet function")


greeting()
```

Output

```
This is a greeting function
I am still in the greet function
```

Calling *greeting()* function will call the function and execute the code inside that function.

Example

Write a function that takes input from user and prints the length of a string.

```
def calculateLength():
    a = input("Enter any sentence = ")
    print(f"Length of sentence is {len(a)}")


calculateLength()
calculateLength()
```

Output

```
Enter any sentence = hello world
Length of sentence is 11
Enter any sentence = python is best language
Length of sentence is 23
```

Passing arguments to a function

Information can be passed into functions as arguments.

Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.

Below you can see there is a function with one argument (name). When the function is called, we pass along a name, and it will greet us.

```
def greeting(name):
    print(f"Hello {name}. Welcome to learn python.")


greeting("James")
greeting("Vandana")
```

Output

```
Hello James. Welcome to learn python.
Hello Vandana. Welcome to learn python.
```

Example

Passing 3 numbers and calculating sum of 3 numbers.

```
def total(a, b, c):
    t = a + b + c
    print(f"Total = {t}")


total(10, 20, 30)
total(-10, -10, 100)
total(50, 50, 200)
```

Output

```
Total = 60
Total = 80
Total = 300
```

Note: If there are 3 parameters required, and if we send 2 only, we will get an error.

Function with return value

The return statement is used to return a value from a function. In the case below, we will try to return the total of 3 numbers and printing it outside the function.

```
def total(a, b, c):
    t = a + b + c
    return t


a = total(10, 20, 30)
print(a)

b = total(-10, -10, 100)
print(b)

print(total(50, 50, 200))
```

```
60
80
300
```

Here, we can see the function ***total*** calculates the total and returns it.

After returning the value from the function, we need to store it in a variable so we can use it print.

Example

Check if the number passed to a function is odd or even. If even return TRUE, else return FALSE.

```
def check(n):
    if n % 2 == 0:
        return True
    else:
        return False


print(check(45))
print(check(100))
print(check(10))
```

Output

```
False
True
True
```

Named Parameters in a function

We can also send the data along with their name so it is easy for other coders to understand it. Let us see an example, to calculate total of 3 subjects.

Without Named Parameter

```
def total(maths, science, computer):
    t = maths + science + computer
    print(f"Marks in Maths are {maths}")
    print(f"Marks in Science are {science}")
    print(f"Marks in Computer are {computer}")
    print(f"Total Marks are {t}")


total(60, 50, 80)
```

Output

```
Marks in Maths are 60
Marks in Science are 50
Marks in Computer are 80
Total Marks are 190
```

With Named Parameter

```
def total(maths, science, computer):
    t = maths + science + computer
    print(f"Marks in Maths, Science and Computer are {maths},
{science} and {computer}")
    print(f"Total Marks are {t}")


total(maths=80, science=99, computer=78)
total(computer=95, science=44, maths=10)
```

Output

```
Marks in Maths, Science and Computer are 80, 99 and 78
Total Marks are 257
Marks in Maths, Science and Computer are 10, 44 and 95
Total Marks are 149
```

With named parameter, it is easy to send parameter and easy of other developers to understand your code.

Passing list as an argument in function

Let us pass a list to a function, and iterate it in the function.

```
def myfunction(mylist):
    for i in mylist:
        print(i, end=" ")
    print()


myfunction([55, 33, 22, 11])
myfunction([67, 89, 87, 47, 88, 11, 23])
```

Output

```
55 33 22 11
67 89 87 47 88 11 23
```

Example

Pass 2 list in a function, combine them, and print them.

```
def myfunction(mylist1, mylist2):
    print(mylist1 + mylist2)


myfunction([55, 33, 22, 11], [100, 200, 300])
myfunction([67, 89, 87, 47, 88, 11, 23], [80, 90, 100])
```

Output

```
[55, 33, 22, 11, 100, 200, 300]
[67, 89, 87, 47, 88, 11, 23, 80, 90, 100]
```

PASS Statement

function definitions cannot be empty, but if you for some reason have a ***function*** definition with no content, put in the pass statement to avoid getting an error.

```
def myfunction():
    pass
```

Built-In Functions

Python comes with a lot of built-in functions. They are available anywhere, and you can get a list of them by inspecting the ***builtins*** module with ***dir(__builtins__)***, or by going to the official Python documentation. Unfortunately, we don't have the room to go through all of them here. We've already seen some of them, such as ***any , bin , bool , divmod , filter , float , getattr , id , int , len , list , min , print , set , tuple , type ,*** and ***zip*** , but there are many more, which you should read about at least once. Get familiar with them, experiment, write a small piece of code for each of them, and make sure you have them at your fingertips so that you can use them when needed.

You can find a list of built-in functions in the official documentation, here:

https://docs.python.org/3/library/functions.html

Practice Examples (Functions)

1. Write a Python function to find the Max of three numbers.
2. Write a Python function to sum all the numbers in a list.
3. Write a Python function to reverse a string.
4. Write a Python function that takes a number as a parameter and check the number is prime or not.
5. Write a Python program to print the even numbers from a given list.
   Sample List: [1, 2, 3, 4, 5, 6, 7, 8, 9]

   Expected Result: [2, 4, 6, 8]
6. Write a Python function that checks whether a passed string is palindrome or not.

For solutions, check the end of the book.