# Functions in Python :

Function is a group of statements performing a specific task.
When a program gets bigger in size and its complexity grows, it gets difficult for a programmer to keep track of which piece of code does what, in order to reduce the space for repeated codes, functions come into picture.
A function can be reused by the programmer in a given program for any number of times.

Syntax:

```
def function_name(parameters):
   function_body
   return output
```

Ex_1.1:

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

## Function call:

Whenever we want to call a function, we put the name of the function followed by parentheses

Syntax:

```
function_name().
```

You can also include arguments if function contains parameter such as addition(2, 3) in case of Ex_1.1.

## Types of functions in python:

There are two types of functions:

1. Built-in function
2. User defined function

## Built-in function:

These are standard functions that are present in python are readily available for use.
e.g., print(), len(), range() etc.,

## User defined function:

The function that we create based on the requirements of our own.

## Types of python Function Arguments:

Python supports various types of arguments that can be passed at the time of the function call.

1. Default arguments
2. Keyword arguments
3. Positional arguments
4. Arbitrary arguments

## Default arguments:

A default arguments is a parameter that assumes a default value if a value is not provided in the function call for that argument. The following example illustrates Default arguments.

Ex_1.2:

```
def addition(a, b=10):
   c = a+b
   return c

addition(2)
```

In this function, b is provided by default argument as 10.So, the function call takes 2 for a and since b is not provided it will take the default value 10.

## Keyword arguments:

The idea is to allow the caller to specify the argument name with values so that the caller does not need to remember the order of parameters.

Ex_1.3:

```
def name(first_name, last_name):
   return first_name+“ ”+last_name

name(first_name = “Vijaiey”, last_name = “Anand”)
name(last_name = “Anand”, first_name = “Vijaiey”)
```

Since the parameters are explicitly mentioned in the function call, its order does not matter. Both of them give the same output.

## Positional arguments:

We used the Positional arguments during the function call so that the first argument (or value) is assigned to the first parameter and the second argument (or value) is assigned to the second parameter. By changing the position, or if you forget the order of the positions, the values can be used in the wrong places.

Ex_1.4:

```
def marks_scored(subject, marks):
   return f‘Mark scored in {subject} is {marks}’

marks_scored(‘Maths’, 98)
marks_scored(98, ‘Maths’)
```

In the first function call the arguments are in correct order with respect to the function parameters but in the second the order is changed, so the output will be wrong for the second.

## Arbitrary arguments:

In Python Arbitrary Keyword Arguments, *args and **kwargs can pass a variable number of arguments to a function using special symbols. There are two special symbols:

- *args in Python (Non-Keyword Arguments)
- **kwargs in Python (Keyword Arguments)

Ex_1.5:

```
def function_check(*args, **kwargs):
   if args:
      print(args)
   if kwargs:
      print(kwargs)

function_check('Maths', 'Science', 60, 75, frst_name='Vijaiey', last_name='Anand')
```

In this arguments without parameter/keyword is assigned with *args and arguments with parameter/keyword is assigned with **kwargs.

## Recursion:

A function that calls itself is said to be recursive, and the technique of employing a recursive function is called recursion.
Recursion is a common mathematical and programming concept.
This has the benefit of meaning that you can loop through data to reach a result.

Ex_1.6:
```
def factorial(n):
    if n==0 or n==1:
        return 1
    else:
        return n*factorial(n-1)

factorial(5)
```