Introduction

In all programming and scripting language, a function is a block of program statements which can be used repetitively in a program. It saves the time of a developer. In Python concept of function is same as in other languages. There are some built-in functions which are part of Python. Besides that, we can defines functions according to our need.

User defined function

- In Python, a user-defined function's declaration begins with the keyword def and followed by the function name.
- The function may take arguments(s) as input within the opening and closing parentheses, just after the function name followed by a colon.
- After defining the function name and arguments(s) a block of program statement(s) start at the next line and these statement(s) must be indented.

Here is the syntax of a user defined function.

Syntax:

```
def function_name(argument1, argument2, ...) :
     statement_1
     statement_2
     ....
```

Call a function

Calling a function in Python is similar to other programming languages, using the function name, parenthesis (opening and closing) and parameter(s). See the syntax, followed by an example.

Syntax:

```

function_name(arg1, arg2)
```

Example:

```
def avg_number(x, y):

    print("Average of ",x," and ",y, " is ",(x+y)/2)

avg_number(3, 4)
```

Copy
Output:

```
Average of  3  and  4  is  3.5
```

Explanation:

1. Lines 1-2 : Details (definition) of the function.
2. Line 3 : Call the function.
3. Line 1 : Pass parameters : x = 3, y = 4
4. Line 2 : Print the value of two parameters as well as their average value.

Function without arguments:

The following function has no arguments.

```
def function_name() :
   statement_1
   statement_2
   ....
```

Example:

```
def printt():

    print("This is Python 3.2 Tutorial")

    print("This is Python 3.2 Tutorial")

    print("This is Python 3.2 Tutorial")

printt()
```

Copy
Output:

```
This is Python 3.2 Tutorial
               This is Python 3.2 Tutorial
               This is Python 3.2 Tutorial
```

Explanation:

1. Lines 1-4 : Details (definition) of the function.
2. Line 5 : Call the function.

3. Line 1 : No parameter passes.
4. Line 2-4 : Execute three print statements.

The Return statement in function

In Python the return statement (the word return followed by an expression.) is used to return a value from a function, return statement without an expression argument returns none. See the syntax.

```
def function_name(argument1, argument2, ...) :
    statement_1
    statement_2
    ....
    return expression

function_name(arg1, arg2)
```

Example:

The following function returns the square of the sum of two numbers.

```
def nsquare(x, y):

    return (x*x + 2*x*y + y*y)

print("The square of the sum of 2 and 3 is : ", nsquare(2, 3))
```

Copy
Output:

```
The square of the sum of 2 and 3 is :  25
```

Explanation:

1. Lines 1-2 : Details (definition) of the function.
2. Line 3 : Call the function within a print statement.
3. Line 1 : Pass the parameters x = 2, y = 3
4. Line 2 : Calculate and return the value of $(x + y)^2$

Default Argument Values

In function's parameters list we can specify a default value(s) for one or more arguments. A default value can be written in the format "argument1 = value",

therefore we will have the option to declare or not declare a value for those arguments. See the following example.

Example:

The following function returns the square of the sum of two numbers, where default value of the second argument is 2.

```
def nsquare(x, y = 2):

    return (x*x + 2*x*y + y*y)

print("The square of the sum of 2 and 2 is : ", nsquare(2))

print("The square of the sum of 2 and 3 is : ", nsquare(2,4))
```

Copy
Output:

```
The square of the sum of 2 and 2 is:  16
                The square of the sum of 2 and 4 is :  36
```

Explanation:

Lines 1-2 : Details (definition) of the function.

For first print statement [Line no 3]
Line 3 : Call the function without a second argument, within a print statement.
Line 1 : Pass the parameters x = 2, default value.
Line 2 : Calculate and return the value of $(x + y)^2$

For second print statement [Line no 4]
Line 3 : Call the function with all arguments, within a print statement.
Line 1 : Pass the parameters x = 2, y = 4.
Line 2 : Calculate and return the value of $(x + y)^2$

Keyword Arguments:

We have already learned how to use default arguments values, functions can also be called using keyword arguments. Arguments which are preceded with a variable name followed by a '=' sign (e.g. var_name='') are called keyword arguments.

All the keyword arguments passed must match one of the arguments accepted by the function. You may change the order of appearance of the keyword. See the following example.

Example:

```
def marks(english, math = 85, science = 80):

print('Marks in : English is - ', english,', Math - ',math, ', Science - ',science)

marks(71, 77)

marks(65, science = 74)

marks(science = 70, math = 90, english = 75)
```

Copy
Output:

```
Marks in : English is -  71 , Math -  77 , Science -  80
Marks in : English is -  65 , Math -  85 , Science -  74
Marks in : English is -  75 , Math -  90 , Science -  70
```

Explanation:

Line 1: The function named marks has three parameters, there is no default value in the first parameter (english) but remaining parameters have default values (math = 85, science = 80).
Line 3: The parameter english gets the value of 71, math gets the value 77 and science gets the default value of 80.
Line 4: The parameter english gets the value of 65, math gets the default value of 85 and science gets the value of 74 due to keyword arguments.
Line 5: Here we use three keyword arguments and the parameter english gets the value 75, math gets the value 90 and science gets the value 70.

Arbitrary Argument Lists:

The arbitrary argument list is an another way to pass arguments to a function. In the function body, these arguments will be wrapped in a tuple and it can be defined with *args construct. Before this variable, you can define a number of arguments or no argument.

Example:

```
def sum(*numbers):

    s = 0

    for n in numbers:

          s += n

    return s
```

```
print(sum(1,2,3,4))
```

Copy

Output:

```
10
```