FUNCTIONS

A **function** is a named sequence of statements that performs a specific task. Functions help programmers to break complex program into smaller manageable units. It avoids repetition and makes code reusable.

Types of Functions

Functions can be classified into

- BUILT-IN FUNCTIONS
- USER DEFINED FUNCTIONS

1. BUILT-IN FUNCTIONS

The Python interpreter has a number of functions that are always available for use. These functions are called built-in functions. The syntax is

```
function_name(parameter 1, parameter 2)
```

i) type()

```
>>>type(25)
<class 'int'>
```

The name of the function is **type().** The expression in parentheses is called the **argument** of the function. The result, for this function, is the type of the argument. Function takes an argument and returns a result. The result is also called the **return value**.

Python provides functions that convert values from one type to another. The **int()** function takes any value and converts it to an integer, if it can, or it shows error otherwise:

ii) Casting:

```
>>>int('25')
25
>>>int('Python')
valueError: invalid literal for int(): Python
```

int() can convert floating-point values to integers, but it doesn't round off; it chops off the fraction part:

```
>>>int(9.999999)
9
>>>int(-2.3)
-2
```

float() converts integers and strings to floating-point numbers:

```
>>>float(25)
25.0
>>>float('3.14159')
3.14159
```

Finally, *str()* converts its argument to a string:

```
>>>str(25)
'25'
```

iii) range()

The range() constructor returns an immutable sequence object of integers between the given start integer to the stop integer.

Python's range() Parameters

The range() function has two sets of parameters, as follows:

i) range(stop)

stop: Number of integers (whole numbers) to generate, starting from zero.

eg. range(3) == [0, 1, 2].

ii) range([start], stop[, step])

start: Starting number of the sequence.

stop: Generate numbers up to, but not including this number.

step: Difference between each number in the sequence.

Example:

```
>>>range(10)
[0,1,2,3,4,5,6,7,8,9]
>>>range(5,10)
[5,6,7,8,9]
>>>range[10,1,-2]
[10,8,6,4,2]
```

iv) Printing to the Screen

print() function will prints as strings ,everything in a comma separated sequence of expressions, and it will separate the results with single blanks by default.

Example:

```
>>> x=10
>>> y=7
```

>>>print('The sum of',x, 'plus', y, 'is', x+y)

Output:

The sum of 10 plus 7 is 17

print statement can pass zero or more expressions separated by commas.

v) Reading Keyboard Input:

Python provides a built-in function to read a line of text as a standard input, which by default comes from the keyboard. This function is:

- input()

The input() Function

The *input([prompt])* function print the string which is in the prompt and the cursor point will wait for an input.

```
>>>str = input("Enter your input: ");
Enter your input: 10
>>> print("The input is : ", str)
10
```

2. *USER-DEFINED FUNCTIONS*

If the user create own functions then these functions are called ***user-defined functions***.

Function Definition

Syntax:

A function definition is a heart of the function where we will write main operation of that function.

Syntax:

```
def function_name(parameter 1, parameter 2):
    #Function Definition statements
```

Components of function definition

1. Keyword **def** marks the start of function header.
2. A function name to uniquely identify it.
3. **Parameters** (arguments) through which pass values to a function. They are **optional.**
4. A colon (:) to mark the end of function header.
5. **Optional documentation string** (docstring) to describe what the function does.
6. One or more valid python statements that make up the function body. Statements must have same indentation level.

7. An **optional return** statement to return a value from the function.

Example:

```
>>>def welcome(person_name):
        """This function welcome the person passed in as parameter"""
        print(" Welcome " , person_name , " to learn Python")
```

Using Function or Function Call

Once we have defined a function, we can call it from another function, program or even the Python prompt. To call a function we simply type the function name with appropriate parameters.

Syntax:

```
function_name(parameter 1, parameter 2)
```

Example:

```
>>> welcome('Students')
```

Output:

```
Welcome Students to learn Python.
```

The return statement:

The return statement is used to exit a function and go back to the place from where it was called.

Syntax:

```
return variable_name
```

This statement can contain expression which gets evaluated and the value is returned. If there is no expression in the statement or the return statement itself is not present inside a function, then the function will return the None object.

Example:

```
>>>def absolute_value(num):
        """This function returns the absolute value of the entered number"""
        if num >= 0:
                return num
        else:
                return -num
```

```
>>>print(absolute_value(5))
>>>print(absolute_value(-7))
```

Output:

5

7

FLOW OF EXECUTION

The order in which statements run is **called the flow of execution**. Execution always begins at the first statement of the program. Statements are run one at a time, in order from top to bottom. Function definitions do not alter the flow of execution of the program, but when the function is called Instead of going to the next statement, the flow jumps to the body of the function, runs the statements there, and then comes back to pick up where it left off.