FUNCTIONS

If a group of statements is repeatedly required then it is not recommended to write these statements everytime seperately.We have to define these statements as a single unit and we can call that unit any number of times based on our requirement without rewriting. This unit is nothing but function.

The main advantage of functions is code Reusability.
Note: In other languages functions are known as methods,procedures,subroutines etc

Python supports 2 types of functions

1. Built in Functions
2. User Defined Functions

1. Built in Functions:

The functions which are coming along with Python software automatically,are called built in functions or pre defined functions

Eg:
id()
type()
input()
eval()
etc..

2. User Defined Functions:

The functions which are developed by programmer explicitly according to business requirements ,are called user defined functions.

Syntax to create user defined functions:

```
def   function_name(parameters) :
  """ doc string"""
  ----
  -----
  return value
```

Parameters

Parameters are inputs to the function. If a function contains parameters,then at the time of calling,compulsory we should provide values otherwise,otherwise we will get error.

Return Statement:

Function can take input values as parameters and executes business logic, and returns output to the caller with return statement.

Returning multiple values from a function:

In other languages like C,C++ and Java, function can return atmost one value. But in Python, a function can return any number of values.

Types of arguments

```
def  f1(a,b):
    ------
    ------
    ------
 f1(10,20)
```

a,b are formal arguments where as 10,20 are actual arguments

There are 4 types are actual arguments are allowed in Python.

1. positional arguments
2. keyword arguments
3. default arguments
4. Variable length arguments

1. positional arguments:

These are the arguments passed to function in correct positional order.

The number of arguments and position of arguments must be matched. If we change the order then result may be changed.

If we change the number of arguments then we will get error.

2. keyword arguments:

We can pass argument values by keyword i.e by parameter name.

Note:

We can use both positional and keyword arguments simultaneously. But first we have to take positional arguments and then keyword arguments,otherwise we will get syntaxerror.

3. Default Arguments:

Sometimes we can provide default values for our positional arguments.

If we are not passing any name then only default value will be considered.

***Note:**

After default arguments we should not take non default arguments

4. Variable length arguments:

Sometimes we can pass variable number of arguments to our function,such type of arguments are called variable length arguments.

We can declare a variable length argument with * symbol as follows

```
def f1(*n):
```

We can call this function by passing any number of arguments including zero number. Internally all these values represented in the form of tuple.

Note: After variable length argument,if we are taking any other arguments then we should provide values as keyword arguments.

Note: We can declare key word variable length arguments also.
For this we have to use **.

```
def  f1(**n):
```

Note: Function vs Module vs Library:

1. A group of lines with some name is called a function
2. A group of functions saved to a file , is called Module
3. A group of Modules is nothing but Library

Types of Variables

Python supports 2 types of variables.

1. Global Variables
2. Local Variables

1. Global Variables

The variables which are declared outside of function are called global variables.
These variables can be accessed in all functions of that module.

2. Local Variables:

The variables which are declared inside a function are called local variables.
Local variables are available only for the function in which we declared it.i.e from outside of function we cannot access.

global keyword:

We can use global keyword for the following 2 purposes:

1. To declare global variable inside function
2. To make global variable available to the function so that we can perform required modifications

Recursive Functions

A function that calls itself is known as Recursive Function.

Eg:

```
factorial(3)=3*factorial(2)
            =3*2*factorial(1)
            =3*2*1*factorial(0)
            =3*2*1*1
                =6
factorial(n)= n*factorial(n-1)
```

The main advantages of recursive functions are:

1. **We can reduce length of the code and improves readability**
2. **We can solve complex problems very easily.**

Anonymous Functions:

Sometimes we can declare a function without any name,such type of nameless functions are called anonymous functions or lambda functions.

The main purpose of anonymous function is just for instant use(i.e for one time usage)

Normal Function:

We can define by using def keyword.

```
def  squareIt(n):
    return n*n
```

lambda Function:

We can define by using lambda keyword

```
lambda n:n*n
```

Syntax of lambda Function:

```
lambda   argument_list : expression
```

Note:
Lambda Function internally returns expression value and we are not required to write return statement explicitly.

Note: Sometimes we can pass function as argument to another function. In such cases lambda functions are best choice.

We can use lambda functions very commonly with filter(),map() and reduce() functions,b'z these functions expect function as argument.

filter() function:

We can use filter() function to filter values from the given sequence based on some condition.

```
filter(function,sequence)
```

map() function:

For every element present in the given sequence,apply some functionality and generate new element with the required modification. For this requirement we should go for map() function.

reduce() function:

reduce() function reduces sequence of elements into a single element by applying the specified function.

reduce(function,sequence)

reduce() function present in functools module and hence we should write import statement.

Note:

- **In Python every thing is treated as object.**
- **Even functions also internally treated as objects only.**

Function Aliasing:

For the existing function we can give another name, which is nothing but function aliasing.

Nested Functions:

We can declare a function inside another function, such type of functions are called Nested functions.