FUNCTION

FUNCTION- A function is a programming block of codes which is used to perform a single, related task. It only runs when it is called. We can pass data, known as **parameters**, into a function.

A function can return data as a result. We have already used some python **built in functions** like print(),etc. But we can also create our own functions. These functions are called **user-defined functions**.

Advantages of Using functions:

1.**Program development made easy and fast** : Work can be divided among project members thus implementation can be completed fast.

2.**Program testing becomes easy** : Easy to locate and isolate a faulty function for further investigation

3.**Code sharing becomes possible** : A function may be used later by many other programs this means that a python programmer can use function written by others, instead of starting over from scratch.

4.**Code re-usability increases** : A function can be used to keep away from rewriting the same block of codes which we are going use two or more locations in a program. This is especially useful if the code involved is long or complicated.

5.**Increases program readability** : It makes possible top down modular programming. In this style of programming, the high level logic of the overall problem is solved first while the details of each lower level functions is addressed later. The length of the source program can be reduced by using functions at appropriate places.

6.**Function facilitates procedural abstraction** : Once a function is written, it serves as a black box. All that a programmer would have to know to invoke a function would be to know its name, and the parameters that it expects

7.**Functions facilitate the factoring of code** : A function can be called in other function and so on…

Creating & calling a Function(user defined)

A function is defined using the def keyword in python. E.g. program is given below.

```
def my_own_function():
   print("Hello from a function")

#program start here. program code
print("hello before calling a function")
my_own_function() #function calling.now function codes will be executed
print("hello after calling a function")   #Save the this source code in python file and execute it
```

Variable's Scope in function

There are three types of variables with the view of scope.
Local variable – accessible only inside the functional block where it is declared.
Global variable – variable which is accessible among whole program using global keyword.
Non local variable – accessible in nesting of functions, using nonlocal keyword.

Local variable program:	**Global variable program:**

def fun():
  s = "I love India!" #local variable
  print(s)
s = "I love World!"
fun()
	def fun():
	  global s #accessing/making global variable for fun()
	  print(s)
	  s = "I love India!“ #changing global variable's value
	  print(s)

```
print(s)

Output:
I love India!
I love World!
```

```
s = "I love world!"
fun()
print(s)

Output:
I love world!
I love India!
I love India!
```

Global variables in nested function

```
def fun1():
    x = 100
   def fun2():
       global x
       x = 200
   print("Before calling fun2: " + str(x))
   print("Calling fun2 now:")
   fun2()
   print("After calling fun2: " + str(x))

fun1()
print("x in main: " + str(x))
```

OUTPUT:

```
Before calling fun2: 100
Calling fun2 now:
After calling fun2: 100
x in main: 200
```

Non local variable

```
def fun1():
    x = 100
   def fun2():
      nonlocal x #change it to global or remove this declaration
      x = 200
   print("Before calling fun2: " + str(x))
   print("Calling fun2 now:")
   fun2()
   print("After calling fun2: " + str(x))

x=50
fun1()
print("x in main: " + str(x))
```

OUTPUT:

```
Before calling fun2: 100
Calling fun2 now:
After calling fun2: 200
x in main: 50
```

Parameters / Arguments

These are specified after the function name, inside the parentheses. Multiple parameters are separated by comma. The following example has a function with two parameters x and y. When the function is called, we pass two values, which is used inside the function to sum up the values and store in z and then return the result(z):

```
def sum(x,y):    #x, y are formal
    arguments z=x+y
   return z #return the result

x,y=4,5
r=sum(x,y) #x, y are actual arguments
print(r)
```

Note:-

1.Function Prototype is declaration of function with name,argument and return type.

2. A formal parameter, i.e. a parameter, is in the function definition. An actual parameter, i.e. an argument, is in a function call.

Function Arguments

Functions can be called using following types of formal arguments −

Required arguments - arguments passed to a function in correct positional order
Keyword arguments - the caller identifies the arguments by the parameter name
Default arguments - that assumes a default value if a value is not provided to argu.
Variable-length arguments – pass multiple values with single argument name.

```
#Required arguments
```

```
#Keyword arguments
```

def square(x):
  z=x*x
  return z
 	
r=square()
print(r)
#In above function square() we have to definitely need to pass some value to argument x.
	def fun(name, age):
	  print ("Name: ", name)
	  print ("Age ", age)
	  return;
	
	# call function
	fun(age=15, name="mohak")
	# value 15 and mohak is being passed to relevant argument based on keyword used for them.

#Default arguments
def sum(x=3,y=4):
  z=x+y
  return z
 	
r=sum()
print(r)
r=sum(x=4)
print(r)
r=sum(y=45)
print(r)
#default value of x and y is being used when it is not passed
	#Variable length arguments
	def sum(*vartuple):
	  s=0
	  for var in vartuple:
	  s=s+int(var)
	  return s;
	
	r=sum(70, 60, 50)
	print(r)
	r=sum(4,5)
	print(r)
	#now the above function sum() can sum n number of values

Lambda--A lambda function is a small anonymous function which can take any number of arguments, but can only have one expression.

E.g.

```
x = lambda a, b : a * b
print(x(5, 6))
```

OUTPUT:

30

Mutable/immutable properties of data objects with respect to function

Everything in Python is an object, and every objects in Python can be either mutable or immutable.

Since everything in Python is an Object, every variable holds an object instance. When an object is initiated, it is assigned a unique object id. Its type is defined at runtime and once set can never change, however its state can be changed if it is mutable.

Means a mutable object can be changed after it is created, and an immutable object can't.

Mutable objects: list, dict, set, byte array

Immutable objects: int, float, complex, string, tuple, frozen set ,bytes

---

How objects are passed to Functions

Pass arrays to functions

Arrays are popular in most programming languages like: Java, C/C++, JavaScript and so on. However, in Python, they are not that common. When people talk about Python arrays, more often than not, they are talking about Python lists. Array of numeric values are supported in Python by the array module.

e.g.

```
def dosomething( thelist ):
    for element in thelist:
    print (element)
```

dosomething(['1','2','3'])
alist = ['red','green','blue']

#Pass by reference def updateList(list1): print(id(list1)) list1 += [10] print(id(list1)) n = [50, 60] print(id(n)) updateList(n) print(n) print(id(n)) **OUTPUT** 34122928 34122928 34122928 [50, 60, 10] 34122928 #In above function list1 an object is being passed and its contents are changing because it is mutable that's why it is behaving like pass by reference	#Pass by value def updateNumber(n): print(id(n)) n += 10 print(id(n)) b = 5 print(id(b)) updateNumber(b) print(b) print(id(b)) **OUTPUT** 1691040064 1691040064 1691040224 5 1691040064 #In above function value of variable b is not being changed because it is immutable that's why it is behaving like pass by value

dosomething(alist)
OUTPUT:
1
2
3
red
green
Blue
Note:- List is mutable datatype that's why it treat as pass by reference. It is already explained in topic Mutable/immutable properties of data objects with respect to function

---

Functions using libraries

Mathematical functions: Mathematical functions are available under math module. To use mathematical functions under this module, we have to import the module using import math.
For e.g.To use sqrt() function we have to write statements like given below.

```
import math
r=math.sqrt(4)
print(r)
```

OUTPUT :
2.0

Functions available in Python Math Module

ceil(n), floor(n), log2(n),sin(n),sqrt(n),factorial(n),pow(n,y),tan(n),sin(n),fmod(x,y),exp(n)

Functions using libraries(System defined function)

String functions: String functions are available in python standard module.
capitalize(),count(),find(),index(),isalnum(),isupper(),islower(),isspace(),istitle(),isdigit(),isalpha(),replace(),title(),upper(),split(),swapcase(),splitlines()
e.g.
s="i love programming"

```
r=s.capitalize()
print(r)
```
OUTPUT:
```
I love programming
```