As Per CBSE Syllabus 2022-23

Computer Science

Class XII

Chapter-1

Functions In Python

What is a Function?

A Function is a set of codes that are design to perform a single or related task, it provide modularity and increase code reusability. A function executes only when t is being called. Python provides many build in function like print(), len(), type()..etc, whereas the functions created by us, is called User Defined Functions.

When a function is called for execution, Data can be passes to the function called as function parameters. After execution the function may return data from the function.

Benefits of using Function

- Code Reusability: A function once created can be called countless number of times.
- Modularity: Functions helps to divide the entire code of the program into separate blocks, where each block is assigned with a specific task.
- Understandability: use of Functions makes the program more structured and understandable by dividing the large set of codes into functions
- Procedural Abstraction: once a function is created the programmer doesn’t have to know the coding part inside the functions, Only by knowing how to invoke the function, the programmer can use the function.

Types of Functions in Python

1. **Built-in Function:** Ready to use functions which are already defined in python and the programmer can use them whenever required are called as Built-in functions. Ex: len(), print(), type()..etc

2. **Functions defined in Modules:** The functions which are defined inside python modules are Functions defined in modules. In order to use these functions the programmer need to import the module into the program then functions defined in the module can be used.

```
import math
math.pow(5,2)
```

To use the function pow(), first we need to import the module math, because pow() is #defined inside the math module.

3.User Defined Function

Functions that are defined by the programmer to perform a specific task is called as User Defined Functions. The keyword def is used to define/create a function in python.

Defining a Function Syntax:	**Lets define our first user defined function:**

def <Function Name>([parameters]):
        body of the function
        [statements]
	def myFirstFunction (a):
	    print("inside my first function")
	    print(a)

	myFirstFunction("bye")
	Output: Inside my first function bye

Function Name

Function Argument

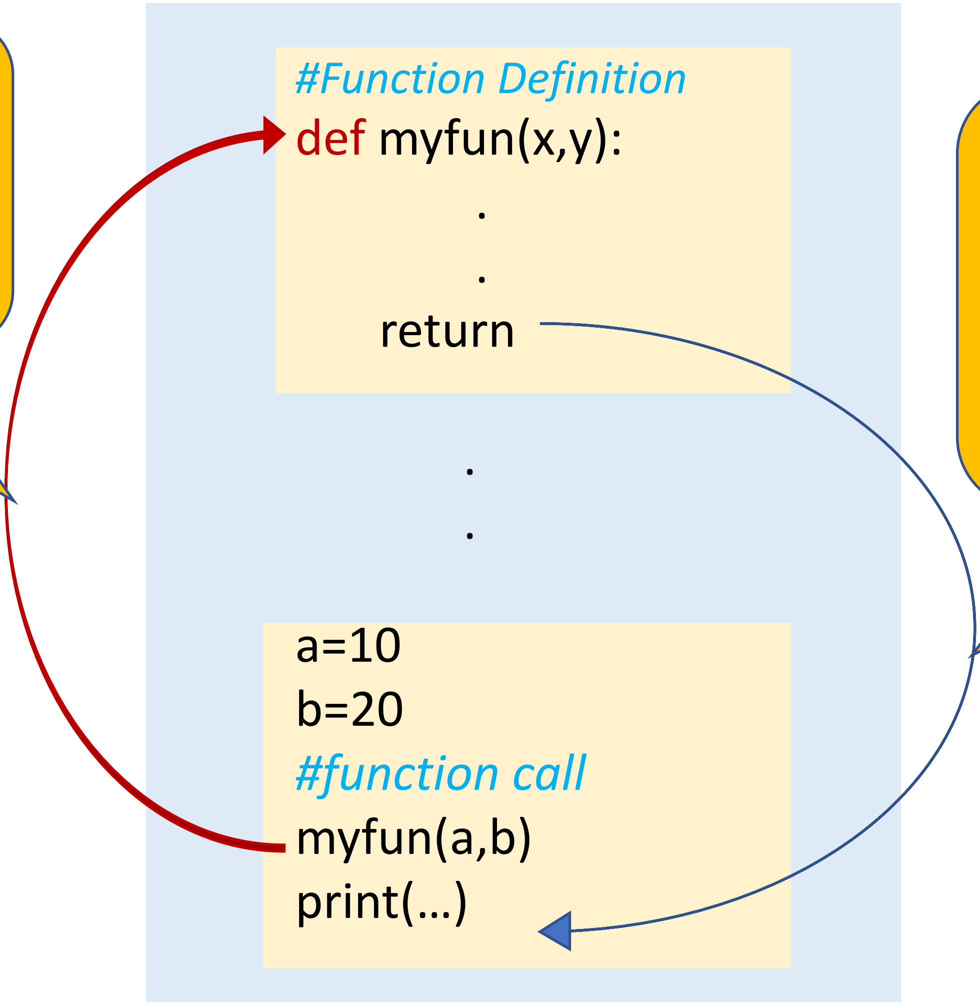
Function call

Elements in Function Definition

- Function Header: The first line of the function definition is called as function header. The header start with the keyword def, and it also specifies function name and function parameters. The header ends with colon (:). def sum(a,b): *#function Header*
- Parameter: Variables that are mentioned inside parenthesis ( ) of the function Header.
- Function Body: The set of all statements/indented-statements beneath the function header that perform the task for which the function is designed for.
- Indentation: All the statements inside the function body starts with blank space (convention is four statements).
- Return statement: A function may have a return statement that is used to returning values from functions. A return statement can also be used without any value or expression.

Flow of Execution in a Function Call

When a function is called the programs control flow will shift to the function definition



```
#Function Definition
def myfun(x,y):
        .
        .
    return

        .
        .

a=10
b=20
#function call
myfun(a,b)
print(…)
```

After the last statement of the function is executed or when a return statement is encountered the control flow will shift back to the place from where the function is called

Arguments and parameters in Function

In Python, the variables mentioned withing parathesis () of a function definition are referred as parameters(also as formal parameter/formal argument) and the variables present inside parathesis of a function call is referred as arguments (also as actual parameter/actual argument).

Python supports 4 types of parameters/formal arguments

1. Positional Argument
2. Default Argument
3. Keyword Argument
4. Variable length argument

1.Positional Arguments:

When function call statement must contain the same number and order of arguments as defined in the function definition, then the arguments are referred as positional argument.

```
def myfun(x,y,z):
          [statements]
```

```
        ....
myfun(a,b,c)     # 3 values(all variables) are passed to the function as arguments
myfun(a,8,9)     # 3 values(1 variables and 2 literals) are passed to the function as arguments
myfun(a,b)       # Error: missing 1 required positional argument: `z’
```

Positional arguments are required arguments or mandatory arguments as no value can be skipped from function call or you cant change the order of the argument passing.

2.Default Argument

Default values can be assigned to parameter in function header while defining a function, these default values will be used if the function call doesn't have matching arguments as function definition.

```
def findresult (mark_secured, passing_mark=30):
        if mark_secured>= passing mark:
                print("pass")
        else:
                print("fail")

findresult(46,40)
findresult(52)
```

Here the argument passing_mark has assigned with a default value, the default value can only be used if the function call doesn't have matching arguments

for this function call default value will not be used as call have matching arguments, hence the value 46 is assigned to mark_secured and 40 assigned to passing_mark.

for this function call default value will be used as function call doesn't have matching arguments as function definition, hence the value 52 is assigned to mark_secured and the default value 30 is assigned to passing_mark.

Output:

```
pass
pass
```

**Note: non-default arguments cannot follow default arguments*

Keyword Argument

Python allows to call a function by passing the in any order, in this case the programmer have to spcify the names of the arguments in function call.

```
def findpow(base, exponent):
        print(base**exponent)

findpow(5,2)
findpow(exponent=5,base=2)
```

In the 1st function call base gets the value 5 and exponent is assigned with the value 2

Keyword Arguments
In the 2nd function call base gets the value 2 and exponent is assigned with the value 5

Output:

```
25
32
```

Variable Length Argument

Variable Length Argument in python permits to pass multiple values to a single parameter. The variable length argument precedes the symbol ‘*’ in the function header of the function call.

```
def findsum(*x):
        sum=0
        for i in x:
              sum=sum+i
        print(“sum of numbers is: ”, sum)

findsum(6,2,8,5,2)
findpow(34,2,4,10,22,14,6)
```

Variable Length Arguments
The function parameter x can hold any number of vaues passed in function call to this function, here the parameter x behaves like a tuple.

Output:

```
sum of numbers is: 23
sum of numbers is: 92
```

Returning Values from Function

Functions in python can return back values/variables from the function to the place from where the function is called using return statement. Function may or may not return values, python function can also return multiple values.

return statement terminates the function execution if it is encountered.

```
def sayhello(name):
        message=”hello “+ name
        return message

m=sayhello(“amit”)
print(m)
```

Function returns a single value back to the place from where function was called.

Output:

```
hello amit
```

Scope of Variables

Part(s) of the program where the variable is legal and accessible is referred as its scope. Based on scope, variable are categorized into 2 categories

Local Variable-: Variable defined inside a function is called as Local Variable. Its scope is limited only within the function in which it is defined.

Global Variable: Variable defined outside all function is called as Global variable.

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

x,y=10,20
m=findsum(x,y)
print(m)
```

Output:

```
30
```

Here the variable a, b and c is defined inside the function, so a, b and c is Local variables.

The variable x, y and m is are not defined inside any function, so x, y and m is Global variables.

Using a Global variable in local scope

To access a variable declared outside all functions(Global Variable) in a local scope then global statement is used.

When global statement is used for a name, it restrict the function to create a local variable of that name instead the function uses the global variable.

```
a=10
def add(x):
        a=x+5
        print(x)

add(20)
print(a)
```

Use of the global statement restrict the function to create a local variable with name a, instead the function is accessing the global variable a. So any changes made to a inside the function will affect the value of global variable a.

```
a=10
def add(x):
        global a
        a=x+5
        print(x)
add(20)
print(a)
```

Output:
```
25
10
```

Inside the function a local variable with name a is created, so changes to the value of local variable a, doesn't affect the value of the global variable a.

Output:
```
25
25
```