Class XII 2024-25

Functions in Python:

- A function is a group of statements defined under a name and perform specific task.
- We may call function by its defined name and may be called/reused multiple times.
- Function is used to achieve modularity and reusability.

The advantages of functions:

Following are the advantages of using functions in a program:

- Increases readability, particularly for longer code as by using functions, the program is better organised and easy to understand.
- Reduces code length as same code is not required to be written at multiple places in a program. This also makes debugging easier.
- Increases reusability, as function can be called from another function or another program. Thus, we can reuse or build upon already defined functions and avoid repetitions of writing the same piece of code.
- Work can be easily divided among team members and completed in parallel.

Functions can be categorized in three types:

1. Built in functions
2. Functions defined in module
3. User defined functions

Built-in Functions:

1. Type Conversion Functions: These are the functions which converts the values from one type to another-

str( ) – To covert any value into string.

float( ) – To covert string into float.

int() To convert the string into integer.

2. input Functions: This function is used to take input from user in the form of string.

e.g. name=input("Enter your name : ")

3. eval function: This function is used to evaluate the value of a string.

e.g. x=eval("45+10")

print(x) # answer will be 55

4. min Function: This function returns the smallest value among given list of values.
5. max Function: This function returns the biggest value among given list of values.
6. abs Function: This function returns the absolute value of any integer which is always positive.
7. type Function: This function is used to identify the type of any value or variable.
8. len Function: This function returns the length of given string.
9. round Function: This function returns the rounded number of given number up to given position.
10. range Function: If you want the series between two numbers then you can use this function. This is good tool for FOR Loop. Its syntax is -

range(start, stop, step)

This gives the series from START to STOP-1 and the interval between two numbers of series will be STEP.

Python Modules

•Module is a .py file which contains the definitions of functions and variables.

•Module is a simple python file.

•When we divide a program into modules then each module contains functions and variables. And each functions is made for a special task.

•Once written code in modules can be used in other programs.

•When we make such functions which may be used in other programs also then we write them in module.

•We can import those module in any program and we can use the functions.

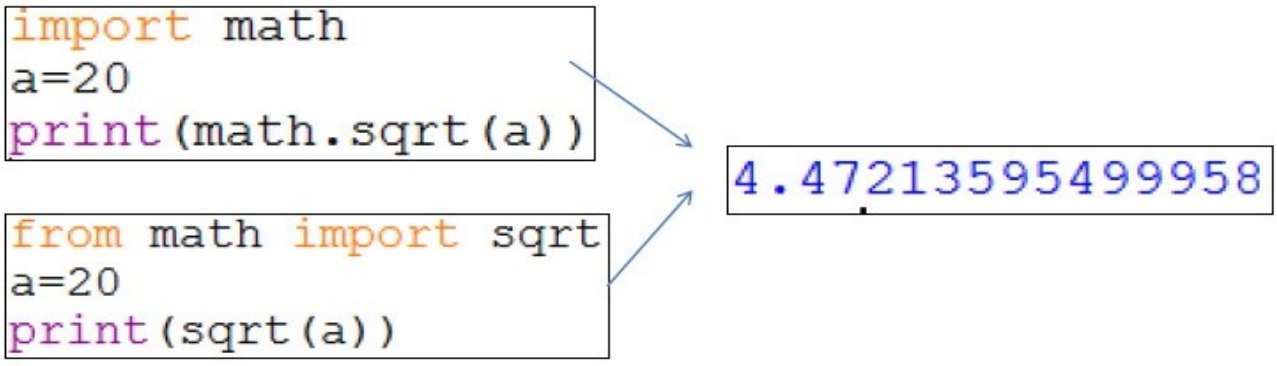
•Python provides two ways to import a module -

•import statement: to import full module.

•from: To import all or selected functions from the module.

Example:

import random
from random import*
import math
from math import*




```
Type "help", "copyright", "credits" or "licens
>>> import random
>>> random.random()
0.1324861821211809
>>> random()
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    random()
TypeError: 'module' object is not callable
>>> from random import*
>>> random()
0.8172754852204971
>>> randint(3,9)
4
>>> randrange(15)
11
>>> randrange(3,9)
4
>>>
```

math Module

•math module contains following functions–

–ceil(x) returns integer bigger than x or x integer.

–floor(x) returns integer smaller than x or x integer.

–pow(x, n) returns xn.

–sqrt(x) returns square root of x.

–log10(x) returns logarithm of x with base-10

–cos(x) returns cosine of x in radians.

–sin(x) returns sine of x in radians.

–tan(x) returns tangent of x in radians.

help Function

•If you forgot that how a library function works then help( ) is very useful for this situation.

•If this function is used with any module then it provides the information and details of the given module and its all the contents.

```
>>> import math
>>> print(help(math.cos))
Help on built-in function cos in module math:

cos(...)
    cos(x)

    Return the cosine of x (measured in radians).

None
```

string Module

•We have already studied about string module in class XI. Here are some other functions of string module.

– **String.capitalize()** Converts first character to Capital Letter

– **String.find()** Returns the Lowest Index of Substring

– **String.index()** Returns Index of Substring

– **String.isalnum()** Checks Alphanumeric Character

– **String.isalpha()** Checks if All Characters are Alphabets

– **String.isdigit()** Checks Digit Characters

– **String.islower()** Checks if all Alphabets in a String, are Lowercase

– **String.isupper()** returns if all characters are uppercase characters

– **String.join()** Returns a Concatenated String

– **String.lower()** returns lowercased string

– **String.upper()**returns uppercased string

– **len()**Returns Length of an Object

– **ord()**returns Unicode code point for Unicode character

– **reversed()**returns reversed iterator of a sequence

– **slice()**creates a slice object specified by range()

random Module

•When we require such numbers which are not known earlier e.g. captcha or any type of serial numbers then in this situation we need random numbers. And here random module helps us. This contains following functions-

•randrange (): This method always returns any integer between given lower and upper limit to this method. default lower value is zero (0) and upper value is one(1).

```
>>> import random
>>> random.randrange(15)
0
>>> random.randrange(15)
10
```

```
import random
sub=["CS","IP","Phy","Chem","Maths"]
ran_index=random.randrange(3)
print(sub[ran_index])
```

random (): This generates floating value between 0 and 1. it does not require any argument.

```
>>> import random
>>> random.random()
0.9576497696403067
>>> random.random()
0.05005384317166639
```

randint (): This method takes 2 parameters a,b in which first one is lower and second is upper limit. This may return any number between these two numbers including both limits. This method is very useful for guessing applications.

```
#Program to generate any two numbers between 0 to 10

import random
print(random.randint(0,10))
print(random.randint(0,10))
```

uniform (): This method return any floating-point number between two given numbers.

```
>>> import random
>>> print("Uniform Lottery Number (1,100): ",random.uniform(1,100))
Uniform Lottery Number (1,100):  85.66278453560943
```

choice (): this method is used for random selection from list, tuple or string.

```
#Program to select My food from a list

import random
lst=["Daal","Chaaval","Roti","MixVeg","Achar","PaneerMasala","malaiKofta"]
print("My First item is : ",random.choice(lst))
print("My Second item is : ",random.choice(lst))
```

shuffle (): this method can shuffle or swap the items of a given list.

```
#Program to shuffle My food list

import random
lst=["Daal","Chaaval","Roti","MixVeg","Achar","PaneerMasala","malaiKofta"]
random.shuffle(lst)
print("My shuffled list is  : ",lst)
random.shuffle(lst)
print("My Second time shuffled list is : ",lst)
= RESTART: C:/Users/KVBBKServer/AppData/Local/Programs/Python/Python36/a.py =
My shuffled list is  :  ['Achar', 'Daal', 'Chaaval', 'malaiKofta', 'PaneerMasala', 'Roti', 'MixVeg']
My Second time shuffled list is :  ['Chaaval', 'PaneerMasala', 'Daal', 'MixVeg', 'Roti', 'Achar', 'malaiKofta']
```

User-defined Functions

- These are the functions which are made by user as per the requirement of the user.
- Function is a kind of collection of statements which are written for a specific task.
- We can use them in any part of our program by calling them.
- def keyword is used to make user defined functions.

A function definition begins with def (short for define).
The syntax for creating a user defined function is as follows:

- The items defined in "( )" are called parameters and they are optional. Hence, a function may or may not have parameters. Also, a function may or may not return a value.
- Function header always ends with a colon (:).
- Function name should be unique. Rules for naming identifiers also applies for function naming.
- The statements outside the function indentation are not considered as part of the function.

Syntax:

```
def function_name(parameter1, parameter2,……):
    set of instructions to be executed
    return <value>
```

User-defined Functions with argument and without return

File Edit Format Run Options Window Help

```
#Program to add two numbers using function whith argument
def add(p,q):  #Function definition
    r=p+q
    print("The sum is:",r)


if(__name__=='__main__'):    #Top level segment
    a=int(input("Enter value of a:"))
    b=int(input("Enter value of b:"))
    add(a,b)      #Function call
```

```
may5func.py
Enter value of a:14
Enter value of b:16
The sum is: 30
```

User-defined Functions with argument and with return value

File Edit Format Run Options Window Help

```
'''Program to add two numbers using function whith argument
and return statement'''
def add(p,q):  #Function definition
    r=p+q
    return r


if(__name__=='__main__'):    #Top level segment
    a=int(input("Enter value of a:"))
    b=int(input("Enter value of b:"))
    c=add(a,b)      #Function call
    print("The result is",c)
```

```
may5BBS_funct2.py
Enter value of a:12
Enter value of b:17
The result is 29
```

User-defined Functions with multiple return values

File Edit Format Run Options Window Help

```
''' Program which return multiple values:'''
def compute(a,b):
    add=a+b
    sub=a-b
    mul=a*b
    div=a/b
    return add,sub,mul,div


#__main__
x=int(input("Input first number"))
y=int(input("Input second number"))
p,q,r,s=compute(x,y)
print("The sum is",p)
print("The subtraction is",q)
print("The multiplication is",r)
print("The division is",s)
```

```
5May_func3.py
Input first number21
Input second number4
The sum is 25
The subtraction is 17
The multiplication is 84
The division is 5.25
```

```
''' Program which return multiple values:'''
def compute(a,b):
    add=a+b
    sub=a-b
    mul=a*b
    div=a/b
    return add,sub,mul,div


#__main__
x=int(input("Input first number"))
y=int(input("Input second number"))
p,q,r,s=compute(x,y)
z=compute(x,y) # Will receive a tuple
print(z)
print("The sum is",p)
print("The subtraction is",q)
print("The multiplication is",r)
print("The division is",s)
```

```
5May_func3.py
Input first number12
Input second number4
(16, 8, 48, 3.0)
The sum is 16
The subtraction is 8
The multiplication is 48
The division is 3.0
```

Parameters and Arguments in Functions :

•When we write header of any function then the one or more values given to its parenthesis ( ) are known as parameter.

•These are the values which are used by the function for any specific task.

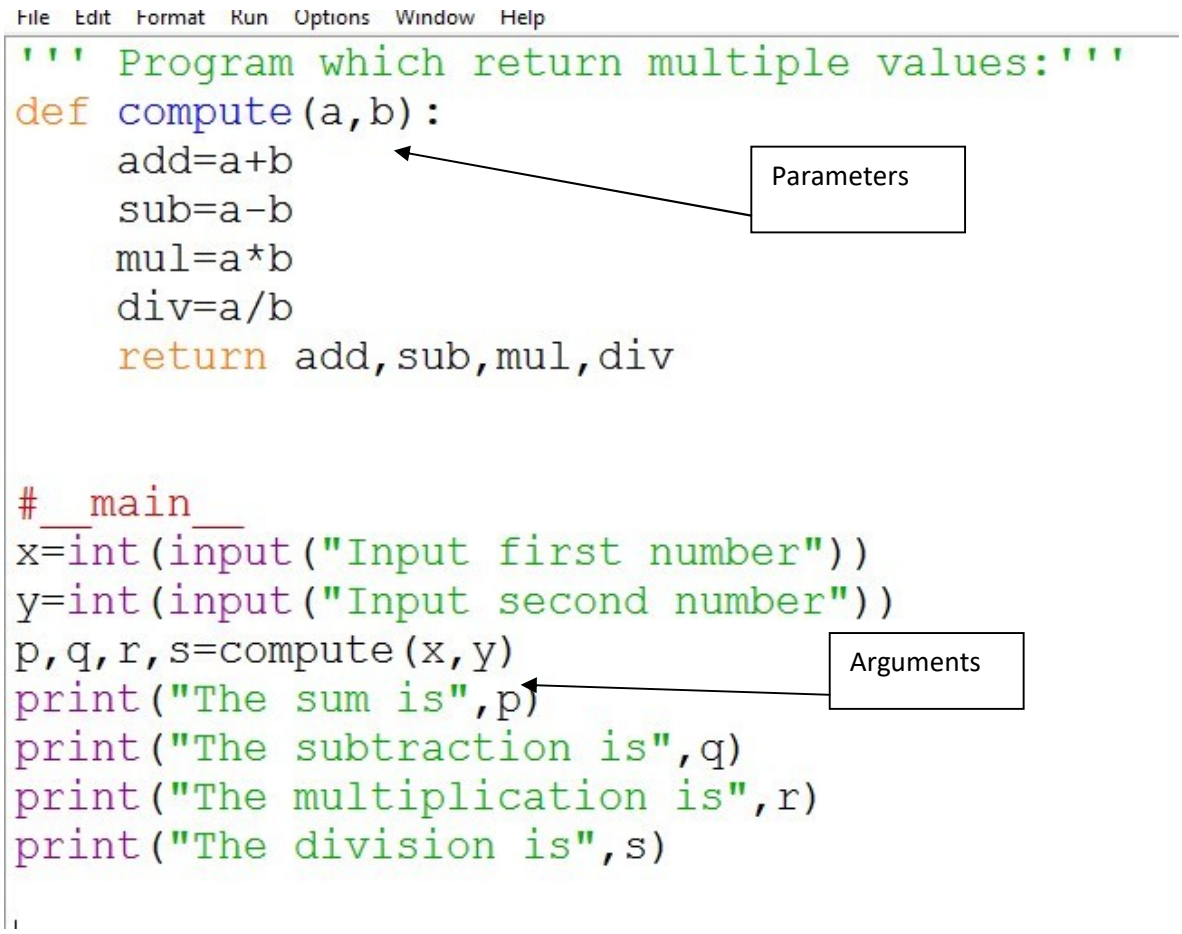
•While argument is the value passed at the time of calling a function.

•In other words the arguments are used to invoke a function.

•Formal parameters are said to be parameters and actual parameters are said to be arguments.

•Both argument and parameter refers to the same value.

•



Types of Arguments

•Python supports 4 types of arguments-

1.Positional Arguments

2.Default Arguments

3.Keyword Arguments

4.Variable Length Arguments

1. Positional Arguments

•These are the arguments which are passed in correct positional order in function.

•If we change the position of the arguments then the answer will be changed.

2. Default Arguments

•These are the arguments through which we can provide default values to the function.

•If we don’t pass any value to the function then it will take a pre defined value.

```
> def greet(message="Good morning",name="India"):
.     print(message,name)
.
> greet()
Good morning India
> greet("Good evening","Pakistan")
Good evening Pakistan
> greet("Good evening")
Good evening India
>
```

3. Keyword Arguments

•If a function have many arguments and we want to change the sequence of them then we have to use keyword arguments.

•Biggest benefit of keyword argument is that we need not to remember the position of the argument.

•For this whenever we pass the values to the function then we pass the values with the argument name. e.g.

```
>>> def compute(a,b,c):
...     print("a=",a)
...     print("b=",b)
...     print("c=",c)
...
>>> compute(a=5,b=7,c=9)
a= 5
b= 7
c= 9
>>> compute(c=10,a=2,b=17)
a= 2
b= 17
c= 10
>>> compute(3,8,10)
a= 3
b= 8
c= 10
>>> compute(b=5,7,9)
SyntaxError: positional argument follows keyword argument
>>>
```

4. Variable Length Arguments

•As we can assume by the name that we can pass any number of arguments according to the requirement. Such arguments are known as variable length arguments.

•We use (*) asterik to give Variable length argument

```
>>> def sum(*n):
        s=0
        for i in n:
                s=s+i
        print("The sum = ",s)


>>> sum()
The sum =  0
>>> sum(10)
The sum =  10
>>> sum(10,20)
The sum =  30
>>> sum(10,20,30)
The sum =  60
```

Passing ARRAYS /LISTS to Function

•In python we use list as array. Well we have to import numpy module to use array in python.

•We will pass here list to the function. As we know that list is better than array.

```
#Program to print the average of integers using list

def list_avg(lst):
    l=len(lst)
    sum=0
    for i in lst:
        sum+=i
    return sum/l

a=input("Input Integers")
a=a.split()
for i in range(len(a)):
    a[i]=int(a[i])

avg=list_avg(a)

print("Average is : ")
print(round(avg,2))
```

Scope of Variable

•Scope of variable means the part of program where the variable will be visible. It means where we can use this variable.

•We can say that scope is the collection of variables and their values.

•Scope can of two types -

•Global (module)

–All those names which are assigned at top level in module or directly assigned in interpreter.

•Local (function)

–Those variables which are assigned within a loop of function.