PYTHON PROGRAMMING

UNIT-II: Functions: Introduction-Built-in Functions-Composition of Functions-User Defined Functions-Parameters and Arguments-Function Calls- The Return Statement-Python Recursive Function-The Anonymous Functions-Writing Python Scripts.

TEXT BOOK

1. E.Balagurusamy, "Introduction to Computing and Problem Solving Using Python", McGraw Hill Education Private Limited, 1st Edition, New Delhi.

UNIT II -FUNCTIONS

- Functions are self-contained programs that perform some particular task.
- Once the function is created, this function can be called anytime to perform that task.
- Each function is given a name. A function may or may not return a value.
- Built in functions dir(), len(), abs() etc., provided in python.
- Users can build their own functions-called user-defined functions.

Advantages of using functions

- Reduce duplication of code in a program
- Break the large complex problems into small parts
- Help in improving the clarity of code
- Piece of code may be reused any number of times.

Built-in Functions

Functions already defined in the python programming

1.Type conversion:- Explicit Conversion

Convert one type of data into another type

Ex:

Int(5.5)

5

Int(‘python’) #string value cannot be int

Value error

Int(5)

5

Float(44)

44.0

Str(67)

‘67’

Print(‘python’+2.7) #cannot concatenate string and float

2.Type coercion-implicit conversion

It is automatically done by the interpreter.

Ex:using type conversion

Minute=59

Float(minute)/60

0.98333

Ex:using type coercion

Minute=59

Minute/60.0

0.98333 #operand is float, the other is automatically converted to float

3. Mathematical function

Python provides math module.

Module is a file that contains some predefined python codes

Module can define function classes and variables.

it is a collection of related functions grouped together.

```
>>>import math
```

To Access the function write the name of the module followed by dot(.) period

Ex:

```
Height=math.sin(90)
Degree=45
Angle=degree*2*math.pi/360.0
```

4. Date and time

- python have built in modules, time and calendar to work with date and time.
- Import time and calendar module.

Ex: getting current date and time

```
Import time;
lt=time.localtime(time.time())
Print "local current time",lt
```

Returns nine tuples, seconds, hour, minute, year, month, day….

Ex.getting formatted date and time

```
Import time;
lt=time.asctime(time.localtime(time.time()))
Print “local time is”,lt
```

Output:

Local time is wed nov 4 19:28:05 2020

Ex: getting calendar for a month

Python provides yearly or monthly calender

```
Import calendar
c=calendar.month(2020,11)
Print “calendar for november\n”,c)
```

Output:

Display the november month calendar.

4. dir() function

- dir() takes an object as an argument

- It returns a list of strings which are names of members of that object

Ex:

```
import math
list=dir(math)
Print list
```

Output

['cos', 'sin', 'tan', 'log', 'pi', 'pow'……]

6. help() function

- It is a built in function which is used to invoke the help system.
- It gives all the detailed information about the module.

Ex:

```
import math
help(math.sin)     #gives the detailed information about the sin function
help(math.cos)
```

Composition of functions

Syntax:

f(g(x))=f.g(x)

- f and g are functions
- Return value of function ‘g’ is passed into the function ‘f’ as parameters/arguments.

Ex:

x=math.sin(angle+math.pi/4)

x=math.exp(math.log(10.0))

User Defined Functions

- Python allows users to define their own functions
- Users have to define the function first known as function definition
- In function definition, users have to define a name for the new function and also the list of the statements that will execute when the function will be called.
- A function is a self contained block of one or more statements that performs a special task when called.

Syntax for function

```
def nameoffunction(parameters)    #function header
        statement1
        statement2
        .....
        statement n
```

Function body

- The function header may contain zero or more number of parameters. These parameters are called formal parameters.

Ex:

```
def display()
    print("welcome to python coding")
display()      #call function
```

Output

Welcome to python coding.

Ex:

```
def print_msg():
    str1=input("please enter your name")
    print("dear",str1,"welcome")
print_msg()          #call function
```

Output

```
dear sai welcome
```

Ex:

```
def sum(x,y):
s=0;
for i in range(x,y+1):
        s=s+i
print("sum of integers from ", x, "to"  y  "is", s)
sum(1,25)
sum(50,75)
sum(90,100)
```

PARAMETERS AND ARGUMENTS

Are the values passed to the functions between parenthesis

Ex:

```
>>>def print_line(line):
        print line
        print line
>>> print_line('Hello')
     Hello
     Hello
>>>print_line(17)
    17
    17
>>>print_line(math.pi)
    3.141
    3.141
```

There are 4 types of formal arguments using which a function can be called:

- Required arguments/positional arguments
- Keyword arguments
- Default arguments
- Variable length arguments

Required arguments

When we assign the parameters to a function at the time of function definition, at the time of calling, the arguments should be passed to a function in correct positional order.

The number of arguments should match the defined number of parameters.

Ex:

```
def display(name,age)
    print(“name=“,name, “age”,=age)
display(“radha”)  #print error message, the argument for age is missing
```

Ex:

```
def display(name,age)
Print(“name=“,name,”age=“,age)
display(“radha”,21)
Display(21,”ramya”)          #it passes 21 to name and ramya to age
```

Keyword arguments

- In keyword arguments the calling recognises the argument by the parameters names
- The programmer can pass a keyword argument to a function by using its corresponding parameter name rather than its position.

Ex:

```
>>> def print_info(name,age):
        print “name:”,name
        print “age:”, age
        return
>>>print_info(age=15, name=‘radha’);
```

Output:

```
name:radha
Age:21
```

Keyword argument example

Precautions for using keyword arguments

1. A positional argument cannot follow keyword arguments

Ex:

```
def display(num1,num2):
    display(40,num2=10)
    display(num2=10,40) #wrong invoking
```

Because the positional arguments 40 appears after the keyword argument num2=10.

2. The programmer cannot duplicate an argument by specifying its as both,a positional argument and a keyword argument.

Ex: consider a functional definition

```
def display(num1,num2)
```

The programmer cannot invoke the display() function as

```
display(40,num1=40)#error
```

i.e multiple values for the parameter num1

3. Default arguments

We can assign a value to a parameter at the time of function definition. This value is considered as default value to that parameter.

Ex:

```
>>>def info(name,age=35):
            print “name:”,name
             print “age:”,age
             return
>>>info(age=20,name=‘radha’);
      output
      name: radha
       age:20
>>>info(name=‘radha’);
     output
     name:radha
      age:35
```

4. variable-length arguments

In many cases where we are required to process a function with more number of arguments than we specified in the function definition. These types of arguments are known as variable length arguments.

For these arguments we use (*) before the name of the variable, which holds the value of all non-keyword variable arguments.

```
>>>def  info(arg1,*vartuple):
            print "result is",arg1
      for var in vartuple:
            print var
       return
```

```
>>>info(10);
```

Output

Result is 10

```
>>>info(90,60,40);
```

output

result is 90

60

40

Function calls

```
>>> def m(a,b):
      mul=a*b
      return mul
>>>a=4
>>>b=3
>>>m1=m(a,b)
>>>print(m1)
```

Output: 12

Ex:

```
def fact(n1):
Fact1=1
Print("entered number is ",n1)
For i in range(1,n1+1)
     fact1=fact1*i
 print("factorial of number",n1 "is",fact1)
Num=int(input("enter the number"))
Fact(num)
```

Output

Enter the number 5

Entered number is 5

Factorial of number 5 is 120

- The function is called using the name with which it was defined earlier, followed by a pair of parenthesis(()). Any input parameters or arguments are to be placed within these calling parenthesis.
- All parameters/arguments which are passed in functions are always passed by reference in python.

The return statement

- The return statement is used to exit a function
- A function may or may not return a value
- If a function returns a value it is passed back by the return statement as argument to the caller

If it does not return a value, we simply write return with no arguments.

Syntax

```
return(expression)
```

Ex:

```
>>>def div(arg1,arg2):
        division=arg1/arg2
        return division
>>>arg3=div(20,10)
>>>print arg3
```

Output

2

Python Recursive Function

- Recursion is generally understood to be the process of repeating something in a self similar way.
- Function can call another function, it is also possible that a function call itself.

Ex:Factorial of a number

4!=4*3*2*1=24

```
>>>def fact(x):
    if x==1:
        return 1
    else:
        return(x*fact(x-1))
>>>fact(4)
```

Output 24

Fibonacci numbers 1,1,2,3,5,8…….

```
Def fib(n):
   if n==0:
        return 1
   if n==1:
       return 1
   return fib(n-1)+fib(n-1)
Print("the value of fibonacci numbers", fib(8))
```

The anonymous functions

- Functions created by lambda keyword.
- They are not defined by using def keyword. For this reason they are called anonymous functions
- We can pass any number of arguments to a lamda form functions, but still they return only one value in the form of expression.
- It is a single line statement function

Syntax

lamda [arg1,arg2….argn]:expression

Ex:

```
>>>mult=lamda val1,val2:val1*val2;
>>>print “value”, mult(20,40)
```

Output

Value 800

The lamda function is defined with 2 arguments val1 and val2. The val1*val2 does the multiplication of 2 values. We can call the mult function with two valid values as arguments.

Writing python scripts

A scripting language is a programming language that uses an interpreter to translate its source code. A python script will be full of functions that can be imported as a library of functions in other scripts, or a python script can be a command.

Two ways of executing python program:

1. Through the python terminal called interactive mode
2. Through scripting

Method 2-called scripting

- We write a python program in notepad and then save the program with .py extension.
- When we have to run the program we type the name of the program in command prompt as:

  python filename.py

  before executing the script the path variable must be correctly set.

To execute the file

1. Open cmd
2. Change directory to python folder
   a. c:\>cd
   b. C:\python2.7\
3. Run python script first.py

   c:\python2.7\>pythonpath>python first.py