****

*** UNIT-IV Python Functions,Modules and Packages ***

****

- Functions, Module and packages are all used to develop modular program.

- Modular Programming : The process of breaking large program into small programs is known as modular programming.

- **Functions:**

- Function is a block of code.

- Function is a piece of code that performs particular task.

- Whenever we want, we can call that function.

- There are two different types of functions.

1) Built-in functions.

2) User defined functions.

- **Built-in Functions:**

- The function which is already exists and we can just re-use it.

- It is also known as predefined function.

- We will not write definition of these functions, we can simply call these functions.

- Examples:

Following are the built-in mathematical functions.

1) min() - Return smallest number among given two numbers.

```
m=min(20,35);
m=20;
```

2) max() - Return largest number among given to numbers.

```
m=max(20,35);
m=35;
```

3) pow() - it will calculate power of given numbers.

```
x=pow(2,3);
x=8;
```

4) round()-It will return the rounded version of specified number.

```
m=round(10.23)
m=10;
n=round(23.78);
n=24;
```

5) abs() - It will return the non-negative value of given number.

```
m=abs(-20);
m=20;
```

6) ceil() - It will give the next number if it contains dot.

```
m=math.ceil(8.1);
print("Ceil Number=",m);
Ceil Number=9
```

7) floor() - It will give the lowest number if it contains dot.

```
m=math.floor(8.1);
print("floor Number=",m);
```

```
floor Number=8
```

-Program:

```
import math;
m=max(20,35);
print("Largest Number=",m);
m=min(20,35);
print("Smallest Number=",m);
m=pow(2,3);
print("Power of 2^3 =",m);
m=round(20.55);
print("Rounded Number=",m);
m=abs(-23);
print("Abs Result=",m);
m=math.ceil(8.1);
print("Ceil Number=",m);
m=math.floor(8.1);
print("Floor Number=",m);
```

-OUTPUT:

```
Largest Number= 35
Smallest Number= 20
Power of 2^3 = 8
Rounded Number= 21
Abs Result= 23
Ceil Number= 9
```

Floor Number= 8

- **User defined Functions**:

- We can create user defined function for particular requirement.

- User defined function is a block of code which solve particular task.

- The main purpose of user defined function is to achieve modularity and enable resuablility of code.

- We can create user defined function using follwing two steps:

1) Function Definition:

- Function definition is a block of statements where we can write the code.

- We use def keyword for defining function.

- The arguments which we write with function definition is known as Formal Arguments.

- ParametersList is optional.

- Syntax:

```
def FunctionName(ParametersList):
        block of statements
```

- Exmaple:

```
def vjtech():
        print("This is user defined function");
```

2) Calling Function:

- The def statement only creates function but not call it.

- If we want to call that function then we can use below syntax:

- Syntax:

```
FunctionName(Arguments);
```

- **Example:**

```
vjtech();
```

- When calling function is executed then program controller goes to function definition.

- After successfully execution of function body, program controller come back to end of the calling function.

- The arguments which we passes through the calling function is known as Actual Arguments.

Program-1:

```
def vjtech():      #function definition
    print("This is user defined function");


vjtech();          #calling function
```

Program-2:

```
def EvenOdd():      #function definition
    no=int(input("Please enter any number:"));
    if(no%2==0):
        print("Number is EVEN!!!");
    else:
        print("Number is ODD!!!");

EvenOdd();          #calling function
```

=====================

Functions Arguments

======================

- Many built-in functions or user defined functions need arguments on which they will operate.

- The value of argument is assigned to variable is known parameter.

- There are four types of arguments:

1) Required Arguments

2) Keyword Arguments

3) Default Arguments

4) Variable length Arguments

- *****Required Arguments/Positional Arguments:**

- Required arguments are the arguments passed to a function in correct positional order.

- The no of arguments in the function call should match with function definition parameters.

- It is also known as positional arguments.

- Example:

✓ **#Test case-1:**

```
def Addition(a,b):
    c=a+b;
    print("Addition of two numbers=",c);
Addition(10,20)          #Required Arguments
```

OUTPUT:

```
Addition of two numbers=30
```

✓ **#Test Case-2:**

```
def Addition(a,b):
    c=a+b;
    print("Addition of two numbers=",c);
Addition()
```

OUTPUT:

```
Traceback (most recent call last):
  File "main.py", line 4, in <module>
    Addition()
TypeError: Addition() missing 2 required positional arguments: 'a' and 'b'
```

- ***Keyword Arguments:

- Keyword arguments are related to function call.

- When we use keyword arguments in the function call, the caller identifies the arguments by the parameter name.

- Example:

```
def display(name,age):
    print("Student Name:",name)
    print("Student Age :",age)

display(name="Mohan",age=35)
```

OUTPUT:

```
Student Name: Mohan
Student Age : 35
```

- *****Default Arguments:**

- A default argument is an argument that assumes a defualt value if a value is not provided in the function call.

- If we pass value to the default arguments then defualt value got override.

- Example:

```
def display(name,age=23):
    print("Student Name:",name)
    print("Student Age :",age)
```

```
display("James")
```

OUTPUT:

```
Student Name: James
Student Age : 23
```

- *****Variable Length Arguments:**

- In many cases where we are required to process function with more numbers of arguments than we specified in the function definition.

- These types of arguments are known as Variable length arguments.

- We can declare variable length arguments using * symbol.

- **Example:**

```
def display(*m):
    print("Value of m=",m);
    display(100,200,300,400,500,600,700,800,900)
```

OUTPUT:

```
Value of m= (100, 200, 300, 400, 500, 600, 700, 800, 900)
```

========================

return Statement:

========================

- The return statement is used to exit function.

- The return statement is used to return value from the function.

- Function may or may not be return value.

- If we write return statement inside the body of function then it means you are something return back to the calling function.

- Syntax:

```
          return(expression/value);
```

- **Example:**

```
def Addition():
   a=int(input("Enter First Number:"));
   b=int(input("Enter Second Number:"));
   c=a+b;
   return c;

m=Addition();
print("Addition of Two Number=",m);
```

OUTPUT:

```
Enter First Number:100
Enter Second Number:200
```

```
Addition of Two Number= 300
```

In python , we can return multiple values.

Example:

```
def Addition():
   a=int(input("Enter First Number:"));
   b=int(input("Enter Second Number:"));
   c=a+b;
   return a,b,c;

m,n,p=Addition();
print("Addition of ",m," and ",n," is ",p);
```

====================

Scope of Variable

====================

- Scope of variable means lifetime of variable.

- Scope of variable decide visibility of variable in program.

- According to variable visibility, we can access that variable in program.

- There are two basic scopes of variables in Python:

✓ **Local Variables:**

- Local variables can be accessed only inside the function in which they are declared.

- We can not access local variable outsie the function.

- Local variables are alive only for the function.

- Local variable is destroyed when the program controller exit out of the function.

✓ **Global Variables:**

- Global variables can be accessed throughout the program.

- We can access global variable everywhere in the program.

- Global variables are declared outside the all functions.

- Global variables are alive till the end of the program.

- Global variable is destroyed when the program controller exit out of the program.

- Example:

```
a=100;                          #Global  variable
def display():
    b=200;                      #local variable
    print("Local Variable b = ",b);
    print("Global Variable a= ",a);
display();
```

======================

Recursion Function:

=====================

- When function called itself is knowns as recursion function.

- A function is said to be recursive if it calls itself.

- Example:

```
def fact(n):
    if n==0:
        return 1;
    else:
        return n*fact(n-1);


result=fact(5);
print("Factorial of 5 number is ",result);
```

OUTPUT:

```
Factorial of 5 number is  120
```

✓ Advantages of Recursion:

1) Recursion functions make the code look clean.

2) Complex task we can manage easily using recursion.

3) Sequence generation is easier using recursion.

✓ DisAdvantages of Recursion:

1) Sometimes logic written using recursion is hard to understand.

2) Recursion function is expensive as they take lot of memory.

3) It consumes more storage space.

4) It is not more efficient in terms of speed and execution time.

====================

Modules

====================

- Modules are primarily the .py file which contains python programming code defining functions,clas,variables,etc.

- File containing .py python code is known as module.

- Most of time, we need to use existing python code while developing projects.

- We can do this using module feature of python.

- Writing module means simply creating .py file which can contain python code.

- To include module in another program, we use import statement.

- Module helps us to achive resuablility features in python.

- Follow below steps while creating module:

1) Create first file as python program with extension .py.This is your module file where we can write functions, classes and variables.

2) Create second file in the same directory which access module using import statement. Import statement should be present at top of the file.

- Example:

✓ **Step-1: #creating Arithmetic.py module**

```
def Add(a,b):
    c=a+b;
    print("Addition of two numbers=",c);
def Sub(a,b):
    c=a-b;
    print("Subtraction of two numbers=",c);
def Div(a,b):
    c=a/b;
    print("Division of two numbers=",c);
def Mul(a,b):
    c=a*b;
    print("Multiplication of two numbers=",c);
```

✓ **Step-2: Accessing Arithmetic module in second file**

```
import Arithmetic
Arithmetic.Add(100,200);
Arithmetic.Sub(100,50);
Arithmetic.Div(500,100);
Arithmetic.Mul(2,4);
```

==

Different Ways of importing modules in Python

===

- While accessing modules, import statement should be written at top of the file.

- import statement is used to import specific module using its name.

- There are different ways of importing modules.

1) Use "import ModuleName":

- In this approach while accessing functions, we have to use module name again and again.

- It means we use sytanx like ModuleName.FunctionName();

- Example:

✓ **Step-1: #creating CheckEvenOdd.py module**

```
def EvenOdd():
    print("Enter Any Integer Number:");
    no=int(input());
    if(no%2==0):
        print("Number is EVEN!!!");
    else:
        print("Number is ODD!!!");
```

✓ **Step-2: Accessing CheckEvenOdd module in second file**

```
import CheckEvenOdd
CheckEvenOdd.EvenOdd();
```

2) Use "from ModuleName import FunctionName"

- In this approach , we can import any particular functions of module.

- We can import multiple functions from the given module but you have to use comma separated by module name((Eg. from ModuleName import Function1,Function2,Function3)

- Here, we can use * symbol for accessing all functions of the module(Eg. from ModuleName import *)

<u>- Example:</u>

✓ **Step-1: #creating Arithmetic.py module**

```
def Add(a,b):
    c=a+b;
    print("Addition of two numbers=",c);
def Sub(a,b):
    c=a-b;
    print("Subtraction of two numbers=",c);
def Div(a,b):
    c=a/b;
    print("Division of two numbers=",c);
def Mul(a,b):
    c=a*b;
    print("Multiplication of two numbers=",c);
```

✓ **Step-2: Accessing Arithmetic module in second file**

```
from Arithmetic import Add,Sub,Div,Mul
Add(10,5);
Sub(10,5);
Div(10,5);
Mul(10,5);
```

3) Rename module name:

- In this approach, we can give another name to existing module.

- But this new name is only applicable for this program only.

- Syntax: import ModuleName as NewName

- Example:

✓ **Step-1: #creating CheckEvenOdd.py module**

```
def EvenOdd():
    print("Enter Any Integer Number:");
    no=int(input());
    if(no%2==0):
        print("Number is EVEN!!!");
    else:
        print("Number is ODD!!!");
```

✓ **Step-2: Accessing CheckEvenOdd module in second file**

```
import CheckEvenOdd as VJ
```

```
VJ.EvenOdd();
```

- **##Practice Program-1:**

✓ **Step-1: #creating FindSquareCube.py module**

```
   def Square(no):
   result=no*no;
   return(result);

def Cube(no):
   result=no*no*no;
   return(result);
```

✓ **Step-2: Accessing FindSquareCube module in second file**

```
from FindSquareCube import *
x=int(input("Please Enter Any Number:"));
m=Square(x);
print("Square of given number=",m);

m=Cube(x);
print("Cube of given number=",m);
```

=========================

Python Built-in Modules:

=========================

****Math & cmath Modules:**

- Python provided two important modules named as math & cmath using this we can perform certain operations. We can access functions related to hyperbolic, trigonometric and logarithmic topics.

- Examples.

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

```
#For Math & cmath Modules
import math
import cmath

#math module functions
x=math.ceil(1.1);
print("Result of ceil(1.1) =",x);
x=math.floor(1.1);
print("Result of floor(1.1) =",x);
x=math.trunc(1.1);
print("Result of trunc(1.1) =",x);
x=math.factorial(5);
print("Result of factorial(5) =",x);
x=math.sin(90);
print("Result of sin(90) =",x);
```

```
x=math.cos(60);
print("Result of cos(60) =",x);
x=math.pow(2,3);
print("Result of pow(2,3) =",x);
x=math.sqrt(9);
print("Result of sqrt(9) =",x);

#cmath module functions
m=2+2j;
print("Result of exp(2+2j) =",cmath.exp(m));
x=cmath.log(m,2);
print("Result of log(2+2j,2) =",x);
x=cmath.sqrt(m);
print("Result of sqrt(2+2j) =",x);
```

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

OUTPUT:

```
Result of ceil(1.1) = 2
Result of floor(1.1) = 1
Result of trunc(1.1) = 1
Result of factorial(5) = 120
Result of sin(90) = 0.8939966636005579
Result of cos(60) = -0.9524129804151563
Result of pow(2,3) = 8.0
Result of sqrt(9) = 3.0
Result of exp(2+2j) = (-3.074932320639359+6.71884969742825j)

Result of log(2+2j,2) = (1.5000000000000002+1.1330900354567985j)

Result of sqrt(2+2j) = (1.5537739740300374+0.6435942529055826j)
```

====================

***Statistics Module:

====================

- This module provides functions which calculate mean,mode,median,etc

<u>- Example:</u>

```
import statistics


x=statistics.mean([2,5,6,9]);

print("Result of mean([2,5,6,9])=",x);


x=statistics.median([1,2,3,8,9]);

print("Result of median([1,2,3,8,9]=",x);


x=statistics.mode([2,5,3,2,8,3,9,4,2,5,6]);

print("Result of mode([2,5,3,2,8,3,9,4,2,5,6])=",x);
```

OUTPUT:

```
Result of mean([2,5,6,9])= 5.5

Result of median([1,2,3,8,9]= 3
```

Result of mode([2,5,3,2,8,3,9,4,2,5,6])= 2

===========================

Python Packages

===========================

- Package is a collection of modules and sub-packages.

- This helps you to achieve reusablility in python.

- It will create hierarchical structure of the modules and that we can access it using dot notation.

- Package is collection of python modules.

- While creating package in python, we have to create empty file named as __init__.py and that file should be present under the package folder.

- Follow below steps while creating packages:

1) First, we have to create directory and give package name.

2) Second, need to create modules and put it inside the package directory.

3) Finally, we have to create empty python file named as __init__.py file. This file will be placed inside the package directory. This will let python know that the directory is a package.

4) Access package in another file using import statement.

- Example:

✓ **STEP-1: Create Module1.py file**

```
def display():
    print("This is display method of module-1");
```

- ✓ **STEP-2: Create Module2.py file**

```
def show():
    print("This is show method of module-2");
```

STEP-3: Create directory named as MyPKG and stored Module1.py and Module2.py file inside it.

STEP-4: Finally, create empty file named as __init__.py file and stored it inside the MyPKG directory.

STEP-5: We can access package named as MyPKG using below

syntax:

```
-> from PackageName import ModuleName;
-> from PackageName.ModuleName import MethodName;
-> import PackageName.ModuleName;
```

======================

Predefined Packages:

======================

- Predefined packages are numpy,scipy,matplotlib,pandas,

- ✓ **numpy:**

- Numpy is the fundamental package for scientific computing with python.

- Numpy stands for numerical python.

- It provided high performance multi-dimensional array object and tools for working with these objects.

- Numpy array is a collection of similiar types of data.

- Numpy array size is fixed. Once it is created we can not change it later.

- Use following command to install predefined package Numpy:

```
python -m pop install numpy;      #windows OS
```

- In Numpy dimensions are called as axes. The number of axes is rank. Numpy array class is called as ndarry. It is also known by the alias array.

- Basic attributes of ndarry class as follow:

1) shape - Specifies the no of elements for each dimension of the array.

2) size - total no of elements in the array.

3) ndim - Deternimes the dimension an array

4) nbytes - number of bytes used to store the data.

5) dtype - determines the datatype of elements stored in array.

- Example:

```
import numpy

a=numpy.array([[10,20,30],[40,50,60]]);
```

```
print("Array Elements:",a);
print("No of dimension:",a.ndim);
print("Shape of array:",a.shape);
print("Size of array:",a.size);
print("Data Type of array elements:",a.dtype);
print("No of bytes:",a.nbytes);
```

OUTPUT:

```
Array Elements: [[10 20 30]
 [40 50 60]]
No of dimension: 2
Shape of array: (2, 3)
Size of array: 6
Data Type of array elements: int32
No of bytes: 24
```

====================

scipy package

====================

- scipy is a library that uses numpy for more mathematical functions.

- Scipy uses numpy arrays as the basic data structre and comes with modules for various commonly used task in scientific programming, including algebra,integration,differential equation and signal processing.

- We use below statement for installation of scipy package.

```
python -m pip install scipy
```

- Scipy package organized into subpackages.

-> cluster - clustering algorithms

-> constants - physical and mathematical constants

-> fftpack - fast fourier tranform routines.

-> linalg - linear algebra

-> odr - orthogonal distance regression.

-> signal - signal processing

-> optimize - optimazation and root finding routines.

-> sparse - sparse matrix and associated routines.

-> special - special functions

-> stats - statistical distributions and functions.

-> ndimage - N-dimensional image processing.

-> spatial - spatial data structre and algorithms

-> io - read data from and write data to file.

- **Example1:**

```
import numpy as np
from scipy import linalg
a=np.array([[1.,2.],[3.,4.]]);
print(linalg.inv(a)) #find inverse of array
```

OUTPUT:

```
[[-2.   1. ]
 [ 1.5 -0.5]]
```

- Example2:

```
import numpy as np

from scipy import linalg

a=np.array([[1,2,3],[4,5,6],[7,8,9]]);

print(linalg.det(a)) #find determinant of array
```

OUTPUT:

```
0.0
```

====================

Matplotlib package

====================

- this package is used for 2D graphics in python programming language.

- It can be used in python script,shell,web application servers and other graphical user interface toolkits.

- There are various plots which can be created usinh python matplotlib like bar graph,histogram,scatter plot,area plot,pie plot.

- Following statement used to install this package:

```
        python -m pip install matplotlib
```

- Example1:

```
#line plot
from matplotlib import pyplot;
x=[2,6,10,2];
y=[2,8,2,2];

pyplot.plot(x,y);
pyplot.show();
```

- Example2:

```
#for bar graph
from matplotlib import pyplot

x=[2,4,8,10];
y=[2,8,8,2];

pyplot.xlabel('X-Axis');
pyplot.ylabel('Y-Axis');
pyplot.bar(x,y,label="Graph",color='r',width=0.5)
pyplot.show();
```

===================

Pandas package

===================

- Pandas is an open source python library providing high performance data manipulation and analysis tool using its powerful data structure.

- It is built on the numpy package and its key data structure is called the DataFrame.

- DataFrame allow you to store and manipulate data in tabular format.

- Following statement we use for installation of Pandas

```
        python -m pip install pandas
```

- Example1:

```
#using dataframe data structure of panda.

import pandas as pd;

dict={"Name":["Vishal","Mohan","Soham","Nilam"],"Salary":[12000,
13000,67000,11000]};

df=pd.DataFrame(dict);

print(df);
```

OUTPUT:

```
    Name  Salary
0  Vishal   12000
1   Mohan   13000
2   Soham   67000
3   Nilam   11000
```

Inspiring Your Success

VJTech Academy…