WORKING WITH PYTHON OOPs

PYTHON Object Oriented Programming System:(OOPs)

Python is truly object oriented programming language. In Python everything is an Object like:classes, objects, functions, methods, modules etc..These can passed as arguments to functions or assign to variables.

OOPS Principles

Oops are the rules (or) guidelines which are supposed to be satisfied by any programming language in order to call that programming language as oopl. Different oops principles are:

1. Class 2. Object 3.Inheritance
4.Polymorphism 5.Encapsulation 6.Abstraction

Define Class?

Class can be described as :

1 Pre-defined keyword.
2 Class is Template or Blue print for Objects
3 Collections of variables and methods
4 Collection of objects that has common properties

Example:

Class Name ==> Phone
Rotary Phone Cellular Phone
Mobile Phone Touch Phone

Example:

```
print(type('Hello'))
```

```
def Hello():
    print("Wish Programmers")
print(type(Hello))
```

Example:

Human ==> Class
Midhun, Daniel, Scott are the Objects
Two hands, Height, weight, color, eyes ==> Attributes
eating(), walking(), sleeping(), writing() ==> Methods

Class in Python

We can define a class by using 'class' keyword.

Syntax1:

```
class Account:
"""This is docstring, explains brief about the class"""
        pass
sb_account=Account()
current_account=Account()
```

Syntax2:

```
class Account(object):
"""This is docstring, explains brief about the class"""
        pass
sb_account=Account()
current_account=Account()
```

Syntax3:

```
class Account():
"""This is docstring, explains brief about the class"""
        pass
sb_account=Account()
current_account=Account()
```

NOTE: Don't miss the colon after the class name

Example:

```
class Car():
    pass
print(Car)
print(type(Car))
print(Car.__name__)
print(Car.__doc__)
```

Example:

```
class Car():
    """This is docstring, explains brief about the class"""
    pass
CC=Car()
print(type(CC))
print(CC)
print(isinstance(CC,Car))
```

NOTE:
Class Name should starts with Capital letter, It is convention in PYTHON

Define Object?
Collection of data and its functionality.
OR
It is an entity that has states & behaviors is known as Object.
OR
An Object is an instance of class, which contains attributes & methods

Define Reference Variable?
The variable which can be used to refer object is called reference variable. By using reference variable, we can access properties and methods of object.

NOTE:
State means Properties/Attributes, Behaviour means methods/functionalities.

Example:
Object1: Human
Identity: Midhun
Properties: Color, Height, Weight
Functionalities: walk () , see() , run() ....

Example:

```
class MyBank_SBI():
          pass
class MyBank_AB:
         pass
```

```
class MyBank_HDFC(object):
          pass
#SS is an Object or Instance
SS=MyBank_SBI()
print(SS)
#AA is an Object or Instance
AA=MyBank_AB()
print(AA)
#HH is an Object or Instance
HH=MyBank_HDFC()
print(HH)
```

NOTE:
Instance indicates relationship between an Object to its Class
(Instanace means reference)

Example:
```
class Banking():
    #Class Body or Logic
    print("Welcome to Banking")
#Creating Instance or Object
#Driver Code
BB=Banking()
print(BB)#display the address of Object or Instance
```

isinstance() method
It checks if the object (first argument) is an instance or subclass of classinfo class (second argument)

Syntax:
```
isinstance(object, classinfo)
```

Example:
```
class Example:
    name = 'NareshIT'
Ex=Example()
print(isinstance(Ex,Example))
```

Comparison between type() and isinstance()
type()
It returns the type of the object

isinstance()
it checks to see if the object passed in the first argument is of the type of any of the type objects passed in the second argument.

Define Method?
A method is an operation on object , with in the class

Example:
```
class Banking():
#It is a Method of Member
#Class inside all are members
    def MyCustomer():
        print("Welcome to Banking")
        return()
#Driver Code
```

```
BB=Banking()
BB.MyCustomer()
#TypeError: MyCustomer() takes 0 positional arguments but 1 was given
```

Self variable:
self is the default variable which is always pointing to current object (like this keyword in JAVA) By using self we can access instance variables and instance methods of object.

Example:
```
class Banking():
    def MyCustomer(self):
        print("Welcome to Banking")
        return()
#Driver Code
BB=Banking()
BB.MyCustomer()
```

Example:
```
class Human():
    "This class demonstrates the creation of Objects"
    #Instance Attributes
    num=100
    #Instance Method
    def EampleMethod(self):
        print("I am a method inside the class")

#Creating Object or Instance of Human Class
HH=Human()
print(HH.num)
#Calling Method
HH.EampleMethod()
print(Human.__doc__)
print(help(Human))
```

Example:
```
class Banking():
    def MyCustomer(x):
        print("Welcome to Banking")
        return()
#Driver COde
BB=Banking()
BB.MyCustomer()
```

Example:
```
class Banking():
    def MyCustomer(self):
        print("Welcome to Banking")
        print(id(self))
#Driver Code
BB=Banking()
BB.MyCustomer()
print(id(BB))
```

NOTE:
As per PYTHON Documentation, self is not mandatory, It is not keyword, self replace by any argument, but first argument is default & acting

as self. self value provided by PVM.

Example:

```
class MyTry_Finally():
    def Exceptions(self):
        try:
            x=int(input("Enter Any Number: "))
            y=int(input("Enter Any Number: "))
            z=x/y
            print("The Result is: ",z)
        except ZeroDivisionError:
            print("Y must not be ZERO")
        except ValueError:
            print("Invalid Input")
        finally:
            print("FinallyBlockSuccess")

#Creating Instance or Object
MyObj=MyTry_Finally()
MyObj.Exceptions()
MyObj.Exceptions()
```

Difference between Function and Method in PYTHON:

Function:

1 It is always independent, not related to any Class
2 Outside the class a piece of code
3 No Self Parameter required
4 No Instance required to call

Method:

1 It is always related to a class
2 Inside the class a piece of code
2 Self Parameter required
4 Always instance required to call

Constructor in PYTHON:

1 It is a special method or magical method in python.
2 Name of the constructor should be __init__
3 It will be executed automatically at the time of object creation.
4 The main advantage of constructor is to declare and initialize instance variables.
5 For every instance constructor will be exeucted one-time
6 It can take atleast one argument(self)
7 It is an optional, if we are not providing any constructor then Python will provide default constructor.

Constructors can be of two types.

1 Parameterized Constructor
2 Non-parameterized Constructor

Example:

```
class MyBanking():
    def __init__(self):
        print("Hello I am Always First Get Executed")
        print("Thank U")
#Creating an Object
MM=MyBanking()
```

Example:

```
class MyBanking():
    def __init__(self,name):
        print("Hello I am Always First Get Executed")
        print("Thank U")
      #Instance Variable is name
        self.name=name
        print("My Name is: ",self.name)
#Creating Object or Instance/Driver Code
MM=MyBanking("PYTHON")
```

Example:

```
class MyBanking():
    def __init__(self,a):
        print("Hello I am Always First Get Executed")
        print("Thank U")
      #Instance Variable is name
        self.name=a
        print("My Name is: ",self.name)
#Creating Object or Instance/Driver Code
MM=MyBanking("PYTHON")
```