**Object Oriented Programming System:**

Oopl: smalltalk, simula-67, java, c++, python

1. Classes and objects:

* Object: It is anything that exist in real world. An object contains properties and does actions.
* Class: A group of object with same behavior. Class is a group name where every object has similar properties.
* Difference between class and object: An object exists physically but object does not.
* A class is a model or plan for creating objects.
* An object is instance (physical form) of class.
* Objects are created from class.
* An object does not exist without class.
* But a class can exist without any object.
* An objects behavior is represented by its properties and actions.
* Properties are represented by variables and actions represented by methods.
* A class represents the common behavior (properties and actions) of group of objects.
* So class contains variable methods.

Syntax:

Class Person:

#properties- variables

#actions-methods

#create the object

P1= Person() #PVM allocates memory block.

P1.methodname()

1. What happens when an object is created?

Ans:

* Pvm allocates memory block.
* Pvm stores objects memory address into ”self”.
* Pvm executes \_\_init\_\_() method.
* Pvm returns the memory address of the object.

Parametrized constructor is used to initialize every object with different data.

1. Instance variable:

Whose separate copy is available to every object.

Example:

def \_\_init\_\_(self):  
 self.id=1001  
 self.name="vaish"

Instance method:

are the method that acts on instance variable.

Obj.method is used to call it.

It has self, has no decorator.

Types of instance method:

* + - * 1. Accessor method: they access or read data,

Getxxx()=getter method=getId(),getname(),getsalary()

* + - * 1. Mutator method: they not only read the data but also modify the data.

Setxxx()=setter method=setId(), setname(), setsalary()

1. Class variable:

is a variable whose single copy is shared by all object.

Constructor is not needed for class variable

Example:

class Myclass:  
 x=10 #this is class variable.

We cannot apply the instance method on the class variable that’s why we have to modify it that’s why we use decorator

@classmethod #decorator  
# cls stores the memory of class namespace.  
# self stores the memory of objects in that class.  
def modify(cls):  
 cls.x+=1 # increment class variable by 1.

Class method:

A method that acts on class variable is called class method

Obj.method or Classname.method()

Cls variable is used

@classmethod is used

1. Static variable:

Same as class variable

Static method

Acts on static variable

Obj.method() or Classname.method() #generally use classname.method to optimize the code for speed and reduce memory.

No default parameter

@staticmethod is used

Class method affects all objects.

Static method works at class level but will not affect any object.

1. Inner class:

A class return inside another class is called inner class. It is also called as nested class.

B is inner class of A.

Inner class is written when data can be sub grouped and processed separately.

1. Encapsulation

Bundling up of data and methods as a single unit.

A class is example for encapsulation.

Student data will not affect emp data they will not interfere with each other. We can use same variable in both the class. Encapsulation secure the data and don’t share the data with other class. The data will not be rewritten.

A class protects its members (vars+method)

We can use same names for the members of two class.

1. Abstraction

Hiding unnecessary data from the user.

Here in bank the normal clerk can access the balance amount but not the loan similarly accountant can access my loan amount but not balance amount but manager can access both.

This method of hiding data is called as encapsulation.

\_\_ before any variable make it private and this variable will not be available outside the class.

Ex:

self.\_\_loan=87579

Name mangling:

Modifying a name in order to break abstraction is called name mangling.

Ex:

print(d.\_Bank\_\_loan)

print(d.\_\_loan)#will not work.

1. Inheritance:

Creating new classes from existing classes. Such that new classes will inherit the properties of existing classes.

New class is sub class or derived class.

Existing class is base class or super class.

Main advantage is reusability and it will increase the productivity of programmer.

Syntax:

Class Subclass(superclass):

Super() is a function that refers to super class from a subclass.

Super().\_\_init\_\_(args) refers to super class constructor

Super().method(args) calls super class method.

Types of inheritance

1. Single inheritance:

Deriving sub classes from a single super class.

1. Multiple inheritance:

Deriving sub classes from multiple super classes.

Imp: A class cannot be empty.

Syntax of multiple inheritance:

Class Subclass(Super1, Super2, Super3)

Imp: Java does not support multiple inheritance.

1. Polymorphism:

Poly means many, Morphos means forms

If something exists in many forms it is called as polymorphism.

1. Operator overloading:

Same operators perform many actions/operations.

1. Method overloading:

Same method performing more than one operations.

1. Method overriding:

Writing same method in subclass as that of super class.

1. Duck Typing:

Executing method without knowing its type.