OOP'S

==========

Object Oriented Principles or Features or Concepts

==========

=>In real time, to develop any project or application, we must choose a language and it can satisfy two types of principles. They are

1. Functional(Procedure) Oriented Principles-----C,Pascal, cobol,8086,oracle7.3,PYTHON
2. Object Oriented Principles.--------PYTHON C++ JAVA, .NET.....

=>Even though, PYTHON programming Belongs both Functional and Object Oriented Programming language and internally everything is treated as object.

Advantages of Object Oriented Principles

---

1. Objects allows us to store Large Volume of Data (Platform Independent)
2. The Data is visiting between two machines in the form of Cipher Text (encrypted Format). So that we can achieve the Security
3. The Large Volume of Data can be transferred between multiple machines all at once in the form of objects and obtains effective communication.
4. With Objects we can build high effective Re-Usable Applications.
5. The Data is always available around Objects (Effective Memory Usage) and functions can operate on the objects.

==========

Classes

==========

=>The Purpose of Classes Concept is that "To Develop Programmer-Defined Data Type+ To develop any real Time Application in OOPs"

=>The Purpose of Developing Programmer-Defined Data Type is that "To Store Customized Data and To Perform Customized Operations".

=>To Develop Programmer-Defined Data Type by using Classes concept, we use "class" keyword

=>In Python Programming, All Class Names are treated as Programmer-Defined Data Type.

=>Every Program in OOPs, Must starts with Classes concept.

Definition:

=>A Class is a collection Variables (Data Members) and Method .

=>When we define a class, Memory space is not created for Data Members and Methods But Whose Memory Space is created when we create an object w.r.t Class Name.

OOP'S

```
                ============================================
                     Syntax for Defining a class in Python
                ============================================
        class  <clsname>:
               class level data member
               def  instancemethodname(self, list of formal params if any):
                     ------------------------------------
                     ------------------------------------
                     Specify Instance Data Members and Operations
                     ------------------------------------
                     ------------------------------------

               @classmethod
               def  classlevelmethodname(cls,list of formal params if any):
                     ------------------------------------
                     Specify Class Level data members and Operations
                     ------------------------------------
               @staticmethod
               def  staticmethodname(list of formal params if any):
                      ----------------------------------------
                      Utility or Universal Operations
                      ----------------------------------------
```

=>In a class of Python, we define Two Types of Data Members . They are
1. Instance Data Members
2. Class Level Data Members

=>In a class of Python, we define Three Types of Methods . They are
1. Instance Method
2. Class Level Method
3. Static Method

```
                ================================================
                  Types of Data Members in a class of Python
                ================================================
```

=>In a class of Python, we define Two Types of Data Members . They are
1. Instance Data Members
2. Class Level Data Members

Instance Data Members

=>Instance Data Members are used for Storing Specific or Perticular Values of an object and hence Instance Data Members are called Object Level Data Members

=>Instance Data Members Memory space taking Every Time when an object is created.

=>We can Specify the Instance Data Members to the object in three ways. They are

a) Through an Object
b) Through Instance Methods
c) Through Constructors.

=>We can access the Instance Data Members by using Object Name

ObjName.Instance Data Member Name

OOP'S

Class Level Data Members

=>Class Level Data Members are used for Common Values for all Objects.
=>Class Level Data Members Memory space taking One Time Irrespective Number of objects are created.
=>We can Specify the class Level Data Members to the object in two ways. They are

a) Inside Class Name
b) Inside Class Level Method

=>We can access the ClassLevel Data Members in Four Ways.

1) By using Class Name
   ClassName.Class Level Data Member Name
2) By using Object Name
   ObjectName.Class Level Data Member Name
3) By using cls
   cls.Class Level Data Member Name
4) By using self
   self.Class Level Data Member Name

==========

Types of Methods in a Class

==========

=>In a class of Python, we can define Three Types of Methods. They are

1. Instance Method
2. Class Level Method
3. Static Method

Instance Method

=>Instance Methods are used for Performing Specific Operations on the data of object and Hence Instance Methods are called Object Level Methods.
=>Instance Methods always Takes "self" as First Positional Parameters for obtaining id /memory address of Current Class object.
=>Syntax:-

```
def    InstanceMethodName(self, list of formal params):
        ----------------------------------------
       -------Specific Operations on objects-------
       ----------------------------------------
```

=>Instance Methods of a Class must be accessed w.r.t object name or self

```
objectname.InstanceMethodName()
              (or)
self.InstanceMethodName()
```

What is self:

=>self is one of the implicit object used as a First formal parameter in the definition of Instance Method
=>The self contains Id or memory address or reference of Current Class object.
=>self is applicable for objects only.
=>self can be accessed inside of corresponding Instance Method definition only but not possible to access other part of the program.

OOP'S

Class Level Method

=>Class level Methods are used for Performing Class Level Operatons Such as Specifying Class Level Data Members and Performs operations on them (if required).
=>Class Level Methods always Takes "cls" as First Positional Parameters for obtaining Current Class Name.
=>Every Class Level Method must be preceded with a pre-defined decorator called @classmethod
=>Syntax:-

```
@classmethod
def    ClassLevelMethodName(cls, list of formal params):
        ----------------------------------------
       -------Common Operations-------
       ----------------------------------------
```

=>Every Class Level Method can be accessed w.r.t to Class Name or cls or object name or self

```
ClassName.Class Level method Name()
                (OR)
cls.Class Level method Name()
                (OR)
objectname.Class Level method Name()
                (OR)
self.Class Level method Name()
```

What is cls :

=>cls is one of the implicit object used as a First formal parameter in the definition of Class Level Method
=>The cls contains Name of Current Class
=>cls is applicable for Class Level Data Members and Class Level Methods only.
=>cls can be accessed inside of corresponding Class Level Method definition only but not possible to access other part of the program

---

Static Method

=>Static Methods are those which are used for Performing utility Operation or Universal Operation
=>Static Methods Definition neither takes "self" nor takes "cls" but it takes an object as parameter (if req) which belongs other classes
=>Syntax:

```
@staticmethod
def  staticmethodname(list of formal params if any):
       ----------------------------------------
       ----------------------------------------
       ----Utility or Universal Operations----
       ----------------------------------------
```

=>Static Methods Must be accessed w.r.t class name or object name

```
classname.Static Method name()
                (OR)
objectname.Static Method Name()
```

OOP'S

==========

objects in Python

==========

=>When we define a class, memory space is not created for Data Members and Methods but whose memory is created when we create an object w.r.t class name.
=>To do any Data Processing, It is mandatory to create an object.
=>To create an object, there must exists a class Definition otherwise we get NameError.

Definition of object:

=>Instance of a class is called object (Instance is nothing but allocating sufficient memory space for the Data Members and Methods of a class).

Syntax for creating an object

```
varname=classname()
```

Examples: create an object of Student

```
so=Student()
```

Example:- create an object Employee

```
eo=Employee()
```

Differences Betwwen Classes and Objects

Class:

---

1) A class is a collection of Data Members and Methods
2) When we define a class, memory space is not created for Data Members and Methods and it can be treated as specification / model for real time application.
3) Definition of a perticular exists only once
4) When we develop any Program with OOPs principles, Class Definition Loaded First in main memory only once.

Objects:

---

1) Instance of a class is called Object
2) When we create an object, we get the memory space for Data members and Methods of Class.
3)w.r.t One class Definition, we can create multiple objects.
4)we can crate an object after loading the class definition otherwise we get NameError

==========

Constructors in OOPs

==========

=>The purpose of Constructors is that "To Initlize the object".
=>Initlizing the object is nothing but placing our own values without leaving object empty.
=>Before performing Operation on objects, First, we must initlize the object.

Definition of Constructor:

=>A Constructor is one of the Special Method which is automatically or implicitly called by PVM during object creation and whose role is to Initlize the object without leaving object empty.

Syntax for defining Constructor:

```
       def     __init__(self, list of formal params if any):
                      ------------------------
                      ------------------------
                      Block of Statements---Initlization
                      ------------------------
                      ------------------------
```

Rules for Using Constructors in Python

---

1. The name of the constructor is always def __init__(....).
2. Programmatically, Constructors called by PVM automatically / Implicitly during Object Creation and it initlizes the object
3. Constructor can't return any value (It can return only None)
4. Constructors can be inherited
5. Constructor can be Overridden but not possible to Overload

==========

Destructors in Python
and
Garbage Collector

==========

=>We know that Garbage Collector is one of the in-built program in python, which is running behind of every python program and whose is role is to collect un-used memory space and it improves the performnace of python based applications.

=>Every Garbage Collector Program is internally calling its Destructor Functions.

=>The destructor function name in python is def __del__(self).

=>The destructor always called by Garbage Collector for de-allocating the memory space when the program execution completed(Automatic Garbage Collection). Where as constructor called By PVM implicitly when object is created for initlizing the object.

=>When the program execution is completed, GC calls its own destructor to de-allocate the memory space of objects present in program and it is called automatic Garbage Collection.

=>Hence , We have THREE programming conditions for calling GC and to make the garbage collector to call destructor Functions.

a) By default (or) automatically GC calls destructor, when the program execution is completed.

b) Make the object refereence as None by Forcefully

Syntax : objname=None

c) delete the object by using del by Forcefully

Syntax:- del objname

=>Syntax:

```
              def       __del__(self):
                         ----------------------
                        ----------------------
```

==========

OOP'S

Garbage Collector

=>Garbage Collector contains a pre-defined module called "gc"
=>Here gc contains the following Functions.

1) isenabled()----returns True provided GC is running otherwise returns False
2) enable()-------making GC enabled
3) disable()------Making GC disable
=>GC is not under control Programmer but it always maintained and mangaged by OS and PVM

==========

Data Encapsulation and Data Abstraction

==========

Data Encapsulation:

=>The Process of Hiding the confidential Information / Data / Methods from external Programmers / end users is called Data Encapsulation.
=>The Purpose of Encapsulation concept is that "To Hide Confidental Information / Features of Class (Data Members and Methods) ".
=>Data Encapsulation can be applied in two levels. They are

a) At Data Members Level
b) At Methods Level

=>To implement Data Encapsulation in python programming, The Data Members , Methods must be preceded with double under score (_ _)

Syntax1:- (Data member Lavel)

```
class <ClassName>:
        def   methodname(self):
                self.__Data MemberName1=Value1
                self.__Data MemberName2=Value2
                --------------------------------
                self.__Data MemberName-n=Value-n

                (OR)
```

Syntax1:- (Data member Lavel)

```
class <ClassName>:
        def   __init__(self):
                self.__Data MemberName1=Value1
                self.__Data MemberName2=Value2
                --------------------------------
                self.__Data MemberName-n=Value-n
```

Syntax2:- (Method Level)

```
class <ClassName>:
        def   __methodname(self):
                self.Data MemberName1=Value1
                self.Data MemberName2=Value2
                --------------------------------
                self.Data MemberName-n=Value-n
```

OOP'S

Example1:

```
#account.py----file name and treated as module name
class Account:
       def  getaccountdet(self):
              self.__acno=34567
              self.cname="Rossum"
              self.__bal=34.56
              self.bname="SBI"
              self.__pin=1234
              self.pincode=4444444
              #here  acno,bal and pin are encapsulated
```

Example2:

```
#account1.py----file name and treated as module name
class Account1:
       def   __getaccountdet(self):   # here   __getaccountdet() is made is encapsulated
              self.acno=34567
              self.cname="Rossum"
              self.bal=34.56
              self.bname="SBI"
              self.pin=1234
              self.pincode=4444444
```

==========

Data Abstraction:

=>The Process of retrieving / extracting Essential Details without considering Hidden Details is called Data Abstraction.

==========

Inheritance.

==========

=>The main purpose of Inheritance is that " To Build Re-Usable Applications in OOPs".

=>Definition of Inheritance:

=>The process of obtaining Data Members, Methods and Constructors from One Class into Another Class is called Inheritance.

=>The Class Which is giving The Data Members , Methods and Constructors (Features of Class) is called Base or Super or Parent Class

=>The Class Which is Taking The Data Members , Methods and Constructors (Features of Class) is called Derived or Sub or Child Class.

=>The Inheritance principle always follows Logical Memory Management or Virtual Memory Management. This Memory management says that "Neither we write source code nor takes physical memory space ".

=>In Otherwords, Logical Memory Management or Virtual Memory Management makes us to understand, all the features of Base Class can be inherited or avialble in derived class witout taking Physical Memory space and Not writing physical Source Code."

Advantages of Inheritance

=>When we develop any inheritance based application, we get the following Advantages.

1. Application Development time is Less.
2. Application Memory space is Less.
3. Application Execution Time is Less.
4. Application Performance is Enhanced (Improved)
5. Redundency of the code is Minimized.

OOP'S

==========

Type of Inheritances.

==========

=>Type of Inheritance of Inheritance always makes us to understand How the features are inherited from Base Class into derived Class.

=>In Python Programming, we have 5 types of Inheritances. They are

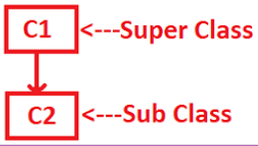
1. Single Inheritance
2. Multi-Level Inheritance
3. Hierarchical Inheritance
4. Multiple Inheritance
5. Hybrid Inheritance.

1) Single Inheritance

Definition:

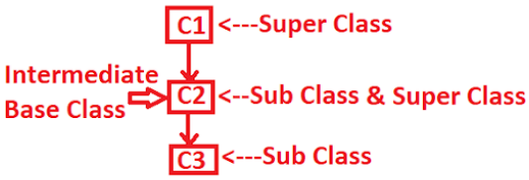
This Inheritance contains Single Base class and Single Derived Class

Diagram:



2. Multi Level Inheritance

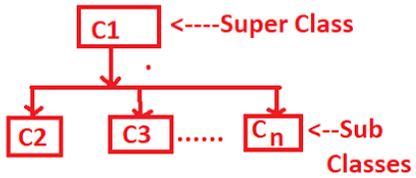
Definition: This Inheritance contains single base class , single derived class and Intermediate base class(es).



3. Hierarchical Inheritance

Definition: This Inheritance Contains Single Super Class and Multiple Sub Classes

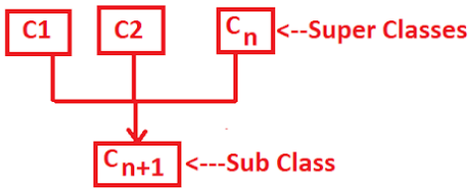
Diagram:



4. Multiple Inheritance

Definition: This Inheritance contains multiple Super Classes and single sub class.

Diagram:



5. Hybrid Inheritance:

Definition:

Hybdrid Inheritance= Combination any available Inheritance Types.

Diagram1:

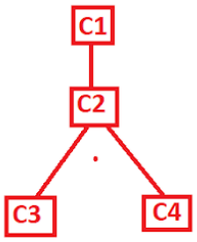
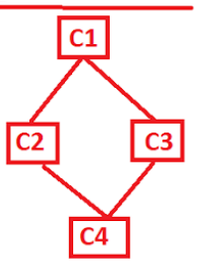


Diagram2:



OOP'S

==========

Inheriting the features of base class into derived class

==========

=>To inherit the features of Base class into Derived class , we use the following Syntax.

```
class  <classname-1>:
      ---------------------
      ---------------------

class  <classname-2>:
      ---------------------
      ---------------------

class  <classname-n>:
      ---------------------
      ---------------------

class  <classname-n+1>(ClassName-1,ClassName-2....ClassName-n>):
      ---------------------
      ---------------------
```

Explanation:

=>Here ,ClassName-1,ClassName-2 ...ClassName-n are called Base Classes
=>Here classname-n+1 is called Derived Class
=>This Syntax Makes us to understand , All the features of Base Class(es) are available in Derived class Logically or Virtually.
=>When we develop any Inheritance Based Application it is always recommended to create an object of Bottom Most Derived Class bcoz It inherites the features of Intermediate Base Classes and Top Most Base Class.

=>For Every Class in Python, Implicitly there exist a pre-defined super class called "object" and it provides Garbage Collection Facility to its sub classes for collecting Un-used Memory space.

==========

Polymorphism in Python

==========

=>Polymorphism is one of the distinct features of OOPs
=>The purpose of Polymorphism is that "Efficient Utilization Memory Space (OR) Less Memory space is achieved".

=>Def. of Polymorphism:

=>The Process of Representing "One Form in multiple Forms " is called Polymorphism.
=>The Polymorphism Principle is implemented(Bring into action) by Using "Method Overriding" feature of all OO Programming Languages.
=>In The definition of polymorphism, "One Form" represents "Original Method" and multiple forms represents Overridden Methods.
=>A "Form" is nothing but existence of a Method. if the method is existing in base class then it is called "Original Method(one form)" and if the method existing derived class(es) then it is called "Overridden Method(multiple Forms)".

OOP'S

==========

Method Overriding in Python

==========

=>Method Overriding=Method Heading is same + Method Body is Different

(OR)

=>The process of re-defining the original method of base class into various derived classes for performing different operations is called Method Overriding.

=>To use Method Overriding in python program we must apply Inheritance Principle.

=>Method Overriding used for implementing Polymorphism Principle.

Examples:

```
-----------------------
#methodoverex1.py
class Circle:
       def  draw(self):  # original Method
              print("Drawing Circle")

class Rect(Circle):
       def draw(self):    # overridden Method
              print("Drawing Rect:")
              super().draw()

class Square(Rect):
       def  draw(self): # overridden Method
              print("Drawing Square:")
              super().draw()

#main program
so=Square()
so.draw()
-------------------------------------------------------------------------
#teacher.py
class Teacher:
       def   readsub(self):
              print("Teacher advises to read 2 hours")

class LazyStudent(Teacher):
       def readsub(self):
              print("LazyStudent never read at all")
class PerfectStudent(Teacher):
       def readsub(self):
              print(" Perfect Student 2hrs reading and practicing")

ls=LazyStudent()
ls.readsub()
ps=PerfectStudent()
ps.readsub()
```