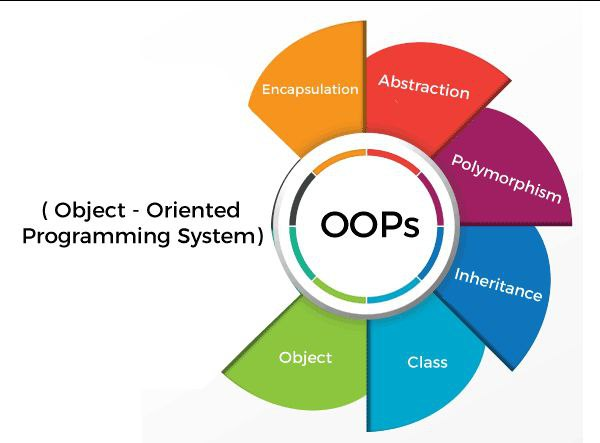
**OOPS part 1**

Oops- object oriented programming

**What is oops?**

Object oriented programming is a programming based on the concept of “object” the object contains both data and code : data in the form of properties (often knows an attribute)and code,in the form of methonds (actions object can perform).

**Types of oops**



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

**Oops terminology**

Attributes also known as Data member or variable behavior also know as member function or method.

**what is class?**

class is Defined with keyword class

The class is a user-defined data structure That binds the data members & Method into a single unit. Class is blueprint or code template or object creation using you can creates as many objects as you want.

class Ex Mercedes, BMW, Toyota

**class Class Name:**

**syntax**

**class greetings():**

**what is object?**

An object is simply a collection of data variables and Methods (functions) that act on those data - Similar a class is a blueprint for that object,The object is the instance of a class. The process of Creating an object can be called instantiation

There is no memory allocation until we created As object. The objector instance contains real data information.

Object: chair ,bike ,pen car .

object

physical entity(real)

1.an instance of a class

2.memory is created when it declared

Ex: apple,orange,mango

attribute (variable) data members

age=20

color='blue’

Self keyword

We can access the atteibutes methods of the class

Class classname :

Class Karthik ( ) : # class definition

Def output(self)

Print(“this is class”)

Output()

* class Shiva():

    a=10

    def show(self):

        print("this is class")

# obj name=class name()

kiran=Shiva()

print(kiran.a)

kiran.show()

* class Srilekha(): # class defination

    a=10

    def ClassMethod(self):

        print("this is class")

# Obj name=Class name()

kiranobj1=Srilekha()

kiranobj=Srilekha() # object declaration

kiranobj.ClassMethod()

print(kiranobj.a)

class Kiran(): # class keyword class name

    a=10 # data member

    def Output(self): # method(self)

        print(self.a)

obj=Kiran()  # declaring of object

obj.Output() # using object we can call the methods

class Naveen():

    a=10 #attribute

    def display(self):

        print(self.a)

ram=Naveen()  # object declaration

# object name . method name ()

syam=Naveen()

ram.display()

syam.display()

**IN it**

Constructors are generally used for instantiating an object. The task of constructors is to initialize (assign values)to the data members of the class when an object of the class is created.

class Prakash():

    def \_\_init\_\_(self,a,b,c):

        self.ff=a

        self.vv=b

        self.dd=c

        print(a)

    def Output(self):

        print(self.ff,self.vv,self.dd)

p=Prakash(1,2,3)

p.Output()

class Mobiles():

    def \_\_init\_\_(self,Mobile\_name,Mobile\_Ram,Mobile\_battery,Mobile\_Price):

        self.a=Mobile\_name

        self.c=Mobile\_Ram

        self.d=Mobile\_battery

        self.e=Mobile\_Price

    def Mobile\_Data(self):

        print("Mobile Name:",self.a)

        print("Mobile Ram:",self.c)

        print("Mobile Battery:",self.d)

        print("Mobile Price:",self.e)

while True:

    name=input("Enter the Mobile Name:")

    ram=input("Enter the Mobile Ram:")

    bat=input("Enter the Mobile Battery:")

    Price=float(input("Enter the Mobile Price:"))

    Mobile\_obj=Mobiles(name,ram,bat,Price)

    Mobile\_obj.Mobile\_Data()

Self.dd=c

Print(a)

Def output(self):

Print(self.ff,self.vv,self.dd)

P=Prakash(1,2,3)

p.output ( )

**inheritance**

inheritance allows us to define a class that inherits all the methods and properties from another class.

Parent class is the class being inherited from also called child class is the inherits from another class also called derived class.

**Polymorphism**

The literal meaning of polymorphism is the condition of the occurrence in different forms.

There are two types

1)method overloading 2) method overriding

**Encapsulation**

Encapsulation is a mechanism of wrapping the data and code acting on the data together as a single unit.

Encapsulation is using by access modifier.

**Data abstraction**

Data abstraction in python is a programming concept that hide complex implementation details while exposing only essential information and functionalities to uses.