In [ ]:
```
#Agend of the Day:
        1. Object Oriented Programming in Python
```

In [ ]:
```
#Introduction Abouts Oops?

     --in 1960s oops was initiated by alan kan
     --with help of c++ which is available in the market
    ---furtherly it was adopted by many programming languages.
#Where we can use oops?
    1. real-Time systems
    2. Artificial Intelligence
    3. Expert Systems
    4. Client-Server Systems
    5. Object-Oriented Databases...etc
```

In [ ]:
```
#Examples of oops languages:
  1. c++
  2. java
  3. Javascript
  4. python
```

In [ ]:
```
#What is object oriented program?
  - Its a different method of structuring a software program
    by bundling the properties and behaviours into invidual
    objects.
    or
    its deals with classes and objects.
-It used to structure a software program into
simple,resuable pieces of code.
```

In [ ]:
```
#Contents of oops?
1. Class
2. object
3. Method
4. Inheritance
   - Single Level
   - Multi Level
   - Hierarchical
   - Multiple
5. Polymorphism
   - Compile Time    - method overloading()
   - Run Time        - method overriding()
6. Data Abstraction
7. Data Encapusulation & Data Hiding
```

In [ ]:
```
#Class:  What is a class?
    A class is a collection of objects.
       or
    A class is blueprint of the object and object must follows that
    class rules.
```

In [ ]:
```
#How to create class? (Logical Entity)
class class_name:
    #vairables declaration
    #construtor method
    #methods
```

In [ ]:
```
#Object: (Physical Entity)
 An Object (instance) of a class that follows the class logic.
```

In [ ]:
```
#How to create objects?
class car:
     #Varibles
     #methods.

#objectvariable = classname()
obj = car()
```

In [18]:
```
#Example: (class,object,classmethod,construtormethod(),class variables,
  #instance variable,object creation, accessing data members of class)
class Myclass:
    sums = 0   #class variable
    c= 50
    d= 60  #self keyword used to access the members of class
    def __init__(self,a,b):    #construtor method or instance method
        self.sums=a+b          #a,b -instance vairables
        print("Construtor is invoked")

    def printsum(self):                #class method
        print("Sum of the a and b is:",self.sums)


#creating objects for above class

a = int(input("enter a value"))
b = int(input("enter b value"))
obj = Myclass(a,b)    #object creation
print(obj.c)
print(obj.d)
print(obj.printsum())
```

```
enter a value10
enter b value20
Construtor is invoked
50
60
Sum of the a and b is: 30
None
```

In [21]:
```
#Example2:
class Student:            #class
    roll = "501"
    height = 5.10                      #class variables
    weight = 50
    color = "white"
    def run(self):                      #class methods
        print("He is run as fast")
    def eat(self):
        print("He is best foodie in our class")

#object creation

dhanu = Student()
print(dhanu.color,dhanu.height,dhanu.weight,dhanu.roll)
         #accessing the values of variables
dhanu.run()
dhanu.eat()  #accessing the class methods.
```

```
white 5.1 50 501
He is run as fast
He is best foodie in our class
```

In [26]:
```
class Student:            #class
    #roll = "501"
    #height = 5.10                      #class variables
    #weight = 50
    def __init__(self,c,h,r,w):
        self.color=c
        self.height=h
        self.roll = r
        self.weight=w
    def run(self):                      #class methods
        print("He is run as fast")
    def eat(self):
        print("He is best foodie in our class")
dhanu=Student("white",5.11,501,50)
print("dhanu Details=",dhanu.color,dhanu.height,dhanu.roll,dhanu.weight)
raghu= Student("Fair",5.10,503,65)
print("raghu Details=",raghu.color,raghu.height,raghu.roll,raghu.weight)
dhanu.run()
```

```
dhanu Details= white 5.11 501 50
raghu Details= Fair 5.1 503 65
He is run as fast
```

In [ ]:
```

```