In [ ]:
```
#What is object oriented Programming?
Object oriented is a programming paradigm in which each and every code or program is treated as
objects.
Functional Programming --> we are prepraring the program that are based on functions(unreal entity)
Example: Student is  an object --> Student is having name , roll no ,marks, best student(whose grade
is greater than 90)
     Each and every object having properties(variables) and behaviour(actions or methods)
object of Human -->age , height , name  methods --> running , walking
Object as Dog --> legs ,eyes , mouth methods --> barking
```

In [ ]:
```
#Why do we need oops?
Object oriented programming is much secure as compare to functional programming
because in object oriented programming we can hide our data/variables.
In object orinted programming we are binding the variables and methods together and we can
use them by creating the object
```

In [ ]:
```
c
```

In [ ]:
```
#What is a class?
class is a blueprint or a model
class are basically use to represent behaviour(methods) and properties(variables)n
class contain both variable and methods
#FOR A SINGLE CLASS ANY NUMBER OF OBJECTS WE CAN CREATE .
```

In [1]:
```
class car:
    color ="red"
    def engine():
        pass
    def millage():
        pass
    def fuel_type():
        pass
```

In [ ]:
```
#What is an Object?
object is physical extension of a class
we can create any numebr of object for a class.
```

In [ ]:
```
Syntax of a class:
    class class_name:
```

In [ ]:
```
Syntax of creating an object:
    reference_variable =classname()
```

In [ ]:
```
Reference variables:
    the variable that are used to refer the object is known as reference variable
    by using reference variable we can access the properties and behaviour of the class
```

In [9]:
```
#demo
class Student:
    def __init__(self,name,rollno,marks):
        self.name=name
        self.rollno=rollno
        self.marks=marks
        print("Constructor is cALLED")
        print(self.name)
    def talks(self):
        print("Hello My name is ",self.name)
        print("My roll no is",self.rollno)
        print("MY marks is",self.marks)
s=Student("Pratyush",987,99)
s1=Student("Twarita",99,90)
s.talks()
s1.talks()
```

```
Constructor is cALLED
Pratyush
Constructor is cALLED
Twarita
Hello My name is  Pratyush
My roll no is 987
MY marks is 99
Hello My name is  Twarita
My roll no is 99
MY marks is 90
```

In [ ]:
```
#Constructor-->
it is a special method
constructor name will always be __init__
Constructor will automatically call when you created an object
if we are not giving ourn own contructor then pvm will execute default constructor
If you want to pass any data while object creation to our class for that we need to use
constructor.
for every object constructor will be called once.
constructor will take atleast one argument(self)
```

In [ ]:
```
self is a variable which will always pointing to the current object
```

In [19]:
```
class area:
    def circle(self,r):
        return 3.14*r*r
    def square(self,l):
        return l**2
    def rectangle(self,l,b):
        return l*b
x=area()
print(x.circle(12))
print(x.square(12))
print(x.rectangle(12,12))
y=area()
print(y.circle(23))
```

```
452.15999999999997
144
144
1661.06
```

In [ ]:
```
#Types of variable/atttributes
1.instance variable
2.static variable
3.local variable
```

In [ ]:
```
#Instance Variable --> if the value of the variable is varied from object to object
such type of variables are known as instance variable .
for every object a seperate copy of variable is created
```

In [26]:
```
#demo
class Student:
    def __init__(self,name,rollno,marks):
        self.name=name
        self.rollno=rollno
        self.marks=marks

    def talks(self):   #Instance method
        print("Hello My name is ",self.name)
        print("My roll no is",self.rollno)
        print("MY marks is",self.marks)
s=Student("Pratyush",987,99)
s1=Student("Twarita",99,90)
print(s.rollno)
print(s1.name)
#s1.talks()
#For declaring the instance variable you need to use  self
#If you want to access any instance variable inside the class then you need to use self variable
#If you want to access any instance variable outside the class then you need to use object reference
```

```
987
Twarita
```

In [ ]:
```
#static variable --> if the value of the variable is not varying with object to object
then such type of variables are known as static variable
a single copy is created for whole class
```

In [34]:
```
#demo
class Student:
    college_name="Indian Institute of Technology"
    def __init__(self,name,rollno,marks):
        self.name=name
        self.rollno=rollno
        self.marks=marks

    def talks(self):   #Instance method
        print("Hello My name is ",self.name)
        print("My roll no is",self.rollno)
        print("MY marks is",self.marks)
        print("My college name is",self.college_name)
        print("My college name is",Student.college_name)
print(Student.college_name)
#s1.talks()
#Static variable are declared inside the class outside the constructor
#If you want to access static variable inside the class then you can use either self or classname
#if you want to access static variable outside the class then you can use eithter object reference
#or classname. but it is recoommnded to use class name
dir(Student)
#Static variable are the only variable that we can access without creating the object
```

```
Indian Institute of Technology
```

Out[34]:
```
['__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 'college_name',
 'talks']
```

In [ ]:
```
local variable --> these variable are used to declare inside the method for the tempeerory requirement
```

In [36]:
```
class test:

    def m1(self):
        a=100
        print(a)
    def m2(self):
        b=200
        print(b)
t=test()
t.m1()
t.m2()

#Local variable are created at the time of function exectuion and once
#function execution is done it is destroyed
#Local variable cannot be accessed outside the function
```

```
100
200
```

In [ ]:
```

```