```
## Object Oriented Programming(OOP's)
- Class
- Object
- Inheritance
- Polymorphism
- Data Abstraction
- Encapsulation
```

In [1]:

```
# class have properties(attributes) & Behaviour(methods--->Functions)
```

Class Synatax

class Classname:

```
    #statment1
    #statment2
```

In [2]:

```
# class Example-1
class Employee:
    id = 10       # Attribute -1
    name = "Devid"  # Attribute -2
    def display(self):
        print(self.id, self.name)
```

In [4]:

```
print(Employee.id)
```

```
10
```

In [5]:

```
print(Employee.name)
```

```
Devid
```

In [8]:

```
# Create Object to Employee Class
my_obj= Employee()
```

In [9]:

```
print(my_obj)
```

```
<__main__.Employee object at 0x000002019F4FA310>
```

In [10]:

```
del my_obj
```

In [11]:

```
my_obj
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-11-9665ddcf580b> in <module>
----> 1 my_obj

NameError: name 'my_obj' is not defined
```

In [12]:

```
emp_obj1 = Employee()
```

In [13]:

```
emp_obj1.display()
```

```
10 Devid
```

In [19]:

```
# Example-2
class Student:
    def show(self,roll,name):
        print("This is {} and I have {} RollNumber.".format(name,roll))

# create Object
stu_obj = Student()

# call method using object
stu_obj.show('001','ravi')

# Create 2nd obj
stu2 = Student()
stu2.show("005",'Kiran')
```

```
This is ravi and I have 001 RollNumber.
This is Kiran and I have 005 RollNumber.
```

In [21]:

```
# Every Class have constructor  __init__()
class Employee:
    # Constructor
    def __init__(self,name,id):

        self.id = id      #instance variable-1
        self.name = name   #instance variable-2
    # Method -1
    def display(self):
        print("ID: %d \n Name: %s"%(self.id,self.name))

#create object-1
emp1 = Employee("John",101)
emp2 = Employee("David",102)

# Accessing display() using objects
emp1.display()
```

```
ID: 101
 Name: John
```

In [22]:

```
emp2.display()
```

```
ID: 102
 Name: David
```

In [28]:

```
# Print count of objects
class Student:
    count = 0
    def __init__(self):
        Student.count = Student.count+1

# Create Multiple Objects
s1 = Student()
s2 = Student()
s2 = Student()
s4 = Student()
```

In [29]:

```
print("The Number of Students Count= ",Student.count)

```

```
The Number of Students Count=  4
```

In [ ]:

```
# Constructors can be of 2 types
# 1) Paramerized Constructor  # ex-> ___init__(self,name,...)
# 2) Non-Paramerized Constructor # ex-> ___init__(self)
```

In [30]:

```
# More than one constructor in single class
class ConEx:
    def __init__(self):
        print("This is First Constructor")
    def __init__(self):
        print("This is Second Constructor")

st = ConEx()
```

```
This is Second Constructor
```

In [32]:

```
# Built-in Class Functions
# getattr(obj,name,default)
```

In [33]:

```
class Student:
    def __init__(self,name,id,age):
        self.name= name
        self.id = id
        self.age =age

# create obj
s = Student('surya',101,26)

# Get attributes of s obj
print(getattr(s,'name'))
```

```
surya
```

In [34]:

```
# Reset Attribute Values using obj
setattr(s,'age',23)
```

In [35]:

```
print(getattr(s,'age'))
```

```
23
```

In [36]:

```
print(hasattr(s,'age'))
```

```
True
```

In [37]:

```
print(hasattr(s,'name'))
```

```
True
```

In [38]:

```
# delete Attribute
delattr(s,'id')
```

In [39]:

```
getattr(s,'id')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-39-51448ff2e0fe> in <module>
----> 1 getattr(s,'id')

AttributeError: 'Student' object has no attribute 'id'
```

In [40]:

```
getattr(s,'name')
```

Out[40]:

```
'surya'
```

In [ ]:

```
# getattr(obj,'attribute name')  # --> returns attribute value
# setattr(obj,'attribute name','attrib value')
# hasaatr(obj,'attribute name')
# delattr(obj, 'attribute name')
```

```
### Inheritance
- single Inheritance
- multilevel Inheritance
- Hierarchical Inheritance
- Multiple Inheritance
```

In [ ]:

```
'''
# Single Inheritance
class Parent:
    properties

class Chield(Parent):
    properties
'''
```

In [41]:

```python
# Ex for single Ineritance
class Animal:
    def speak(self):
        print("Animal Speaking")

class Dog(Animal):
    def bark(self):
        print("Dog barking")

d = Dog()
d.bark()
d.speak()

```

```
Dog barking
Animal Speaking
```

In [42]:

```python
# Multi-Level Inheritance Example
class Animal:
    def speak(self):
        print("Animal Speaking")

class Dog(Animal):
    def bark(self):
        print("Dog barking")

class DogChild(Dog):
    def eat(self):
        print("Eating Bread...")

dc = DogChild()
dc.speak()
dc.bark()
dc.eat()
```

```
Animal Speaking
Dog barking
Eating Bread...
```

In [ ]:

```
# Multiple Inheritance Example
'''
Syntax:-

class Base1:
    Base1 Attr

class Base2:
    Base2 Attr
  .
  .
  .
class BaseN:
    BaseN Attr

class Derived(Base1,Base2,...,BaseN):
    Derived Attr
'''
```

In [50]:

```
# Ex
class Cal1:
    def sum(self,a,b):
        return a+b

class Cal2:
    def mul(self,a,b):
        return a*b

class Cal3:
    def sub(self,a,b):
        return a-b

class Derived(Cal1,Cal2,Cal3):
    def div(self,a,b):
        return a/b

d = Derived()
print(d.sum(10,20))
print(d.mul(23,33))
print(d.sub(8,2))
print(d.div(5,1))
```

```
30
759
6
5.0
```

In [53]:

```
# Methods to check Relationship b/e 2 classes
# issubclass(class1,class2)
# Ex
print(issubclass(Derived,Cal1))
print(issubclass(Cal1,Derived))
```

```
True
False
```

In [56]:

```
# isinstance(obj,class) # -> check object is related to class or not
print(isinstance(d,Derived))
print(isinstance(d,Cal1))
print(isinstance(d,DogChild))
```

```
True
True
False
```

In [ ]:

```

```