OOPS Concept: (object oriented programming language) class object abstraction encapsulation inheritance polymorphism

Class is known as the blueprint 1. Member Variable 2. Member Method

object - gives result to the class

In [2]:

```
class Sret:
    course = "Python"
    def display(self):
        print("Welcome")
#obj name = class name()
#obj name.function_name()
obj = Sret()
obj.display()
```

```
Welcome
```

In [1]:

```
#single inheritance
class Sriher:
    def display(self):
        print("Welcome to SRIHER")
class Sret(Sriher):
    def show(self):
        print("Welcome to SRET")
obj = Sret()
obj.display()
obj.show()
```

```
Welcome to SRIHER
Welcome to SRET
```

In [7]:

```python
#Multilevel Inheritance
class University():                     #super class
    def text(self):
        print("First year class will start on 1st September\n")
class Sriher(University):               #derived class
    def text1(self):
        print("First year class will start on 10th October\n")
class Sret(Sriher):                     #child class
    def text2(self):
        print("First year class will start on 30th October\n")
obj = Sret()
obj.text()
obj.text1()
obj.text2()
```

```
First year class will start on 1st September

First year class will start on 10th October

First year class will start on 30th October
```

In [10]:

```python
#multiple inheritance
class University:
    def text(self):
        print("I am new to OOPS concept")
class Sriher:
    def text1(self):
        print("The topic is taught thru online mode")
class Sret(University,Sriher):
    def text2(self):
        print("My specialization is CYS and IOT")
obj = Sret()
obj.text()
obj.text1()
obj.text2()
```

```
I am new to OOPS concept
The topic is taught thru online mode
My specialization is CYS and IOT
```

In [14]:

```
#op 1
class Addition:
    def add(self,a,b):
        return a+b
#op2
class Subtraction:
    def sub(self,a,b):
        return a-b
#op 3
class Division:
    def div(self,a,b):
        return a/b
#op 4
class Multiplication:
    def mult(self,a,b):
        return a*b
class calculator(Addition,Subtraction,Division,Multiplication):
    def modulo(self,a,b):
        return a%b
obj = calculator()
inp_val1 = int(input("Enter Value for num1\n"))
inp_val2 = int(input("Enter value for num2\n"))

print("Addition : ",obj.add(inp_val1,inp_val2))
print("Subtraction : ",obj.sub(inp_val1,inp_val2))
print("Division : ",obj.div(inp_val1,inp_val2))
print("Multiplication : ",obj.mult(inp_val1,inp_val2))
print("Modulo : ",obj.modulo(inp_val1,inp_val2))
```

```
Enter Value for num1
5
Enter value for num2
6
Addition :  11
Subtraction :  -1
Division :  0.8333333333333334
Multiplication :  30
Modulo :  5
```

In [18]:

```
#Hierarchical Clustering
class University:
    def text1(self):
        print ("This is a class named University ")
class Sriher(University):
    def text2(self):
        print ("Inside class University, Class Sriher is available")
class Sret(University):
    def text3(self):
        print ("Inside class University, There is another class named Sret")
obj_1 = Sriher()
obj_2 = Sret()
obj_1.text1()
obj_1.text2()
obj_2.text1()
obj_2.text3()
```

```
This is a class named University
Inside class University, Class Sriher is available
This is a class named University
Inside class University, There is another class named Sret
```

In [20]:

```
#Abstraction - Providing only the essential info
# A method where it doesnt have any implementation is known as abstarct method and the
from abc import ABC,abstractmethod
class Car(ABC):
    def mileage(self):
        pass
class Honda(Car):
    def mileage(self):
        print("Honda Car Mileage is 20 Kmph")
class Tesla(Car):
    def mileage(self):
        print("Tesla Car Mileage is 30 kmph")
class Lambo(Car):
    def mileage(self):
        print("Lambo Car Mileage is 100 kmph")
obj1 = Honda()
obj1.mileage()
obj2 = Tesla()
obj2.mileage()
obj3 = Lambo()
obj3.mileage()



```

```
Honda Car Mileage is 20 Kmph
Tesla Car Mileage is 30 kmph
Lambo Car Mileage is 100 kmph
```

In [23]:

```python
#method Overriding
#Same function name and same number of parameter
class Bank:
    def getROI(self):
        return 10
class SBI(Bank):
    def getROI (self):
        return 7
class ICICI(Bank):
    def getROI(self):
        return 9
obj = SBI()
print("SBI Bank:",obj.getROI())
obj1 = ICICI()
print("ICICI Bank:",obj1.getROI())

```

```
SBI Bank: 7
ICICI Bank: 9
```