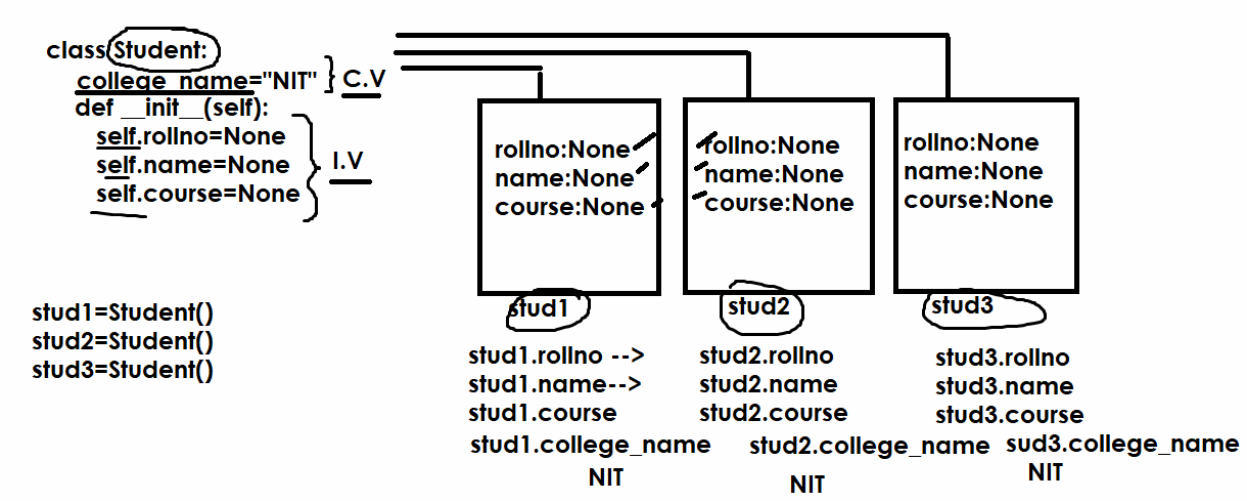
Class variable

The variable created inside class and outside method is called class variable. Inside and outside the class this variable binds with class name.



Why class level variable?

Class level variables are global variables which are global to more than one object.

Class level variable define the common property or variable.

Class level variable memory is allocated within class context and accessed with class name. Memory for this variable is allocated when this variable accessed first time.

Example:

```
class A:
    x=100 # class level variable


print(A.x)

class B:
    z=300 # class level
    def __init__(self):
        self.y=200 # Instance variable

objb1=B()
print(objb1.y)
```

```
print(B.z)
print(objb1.z)
objb2=B()
print(objb2.z)
B.z=900
print(objb1.z)
print(objb2.z)
```

Output

```
100
200
300
300
300
900
900
```

Example:

```
# Find Area of Circle
class Circle:
    pi=3.147
    def __init__(self,r):
        self.__radious=r
    def find_area(self):
        a=self.__radious*self.__radious*Circle.pi
        return a


c1=Circle(1.5)
c2=Circle(1.8)
area1=c1.find_area()
area2=c2.find_area()
print(f'Area of circle1 {area1:.2f}')
print(f'Area of circle2 {area2:.2f}')
```

Output

```
Area of circle1 7.08
Area of circle2 10.20
```

Example:

```
class Account:
```

```
    __min_balance=5000 # Class Level Varaible
    def __init__(self,a,cn,b):
        self.__accno=a # Instance Variable
        self.__cname=cn # Instance Variable
        self.__balance=b # Instance Variable
    def deposit(self,t):
        self.__balance+=t
    def withdraw(self,t):
        if (self.__balance-t)<Account.__min_balance:
            print("Insuff Balance")
        else:
            self.__balance-=t
    def print_account(self):
        print(f'''AccountNo {self.__accno}
CustomerName {self.__cname}
Balance {self.__balance}''')


acc1=Account(101,"naresh",50000)
acc2=Account(102,"suresh",65000)
acc1.print_account()
acc2.print_account()
acc1.deposit(50000)
acc1.print_account()
acc2.withdraw(90000)
acc2.withdraw(61000)
acc2.withdraw(50000)
acc2.print_account()
```

Output

```
AccountNo 101
CustomerName naresh
Balance 50000
AccountNo 102
CustomerName suresh
Balance 65000
AccountNo 101
CustomerName naresh
Balance 100000
Insuff Balance
```

Insuff Balance
AccountNo 102
CustomerName suresh
Balance 15000

What is difference between instance variables and class variables?

Instance variables/object level variable	**Class variables**
Instance variables inside class bind with self and outside the class bind with object name	Class variables bind with class name
Memory is allocated within object context	Memory is allocated within class context
Memory is allocated for every object	Memory is allocated only once and accessed by all the objects

Class Method

A method defined inside class with first parameter “cls” is called class method. To declare a method as class method @classmethod decorator is used.

Syntax:

```
class class-name:
    def method-name(self,param,param,…):
        statement-1
        statement-2

    @classmethod
    def  method-name(cls,param,param,…):
         statement-1
         statement-2
```

Class method is bind with class name and can be invoked without creating object.
Class method is used to access or manipulate class variables but cannot manipulate instance variables (we cannot access instance variables within class method)

Example:

```
class A:
    def m1(self):
        print("Instance method")
    @classmethod
    def m2(cls):
        print("Class method")


A.m2()
obj1=A()
obj1.m1()
```

Output

```
Class method
Instance method
```

Example:

```
class A:
    x=100 #class variable
    def __init__(self):
        self.y=200 # instance variable
    def m1(self):
        print(A.x)
        print(self.y)
    @classmethod
    def m2(cls):
        print(A.x)

A.m2()
obj1=A()
obj1.m1()
```

Output

```
100
100
200
```

Example:

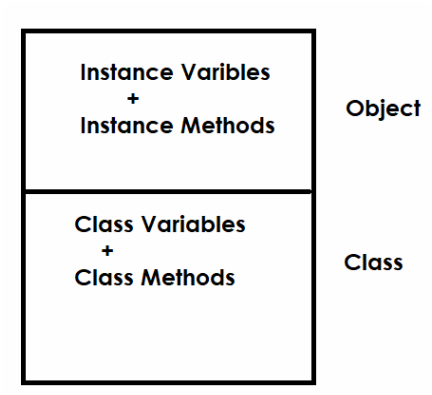
```
class Product:
    __count=0
```

```
    def __init__(self,n,p):
        self.__name=n
        self.__price=p
        print("Product Created...")
        Product.__count+=1
    @classmethod
    def getProductCount(cls):
        return Product.__count

print(Product.getProductCount())
prod1=Product("mouse",100)
print(Product.getProductCount())
prod2=Product("keyboard",1500)
print(Product.getProductCount())
```

Output

```
0
Product Created...
1
Product Created...
2
```



What is difference between instance method and class method?

Instance method	Class method
A method defined inside class with first parameter “self” is called instance method	A method defined inside class with first parameter “cls” with @classmethod decorator is called class method
This method is bind with object	This method is bind with class name

name	or object name
This method can access instance variables and class variables	This method can be access class level variables
This method cannot called without creating object	This method can be called without creating object with class name.
The defines object level operator	This defines class level operation

Static method

A method defined inside class with @staticmethod decorator is called static method.
Static method is global method, which is used to perform global operation.
This method cannot access instance variables or class variables.
This method is bind with class name and can be invoked without creating object.

Syntax:

```
class class-name:
    @staticmethod
    def method-name(param-name,param-name,…):
        statement-1
        statement-1
```

Example

```
class A:
   @staticmethod
   def m1():
      print("static method")


A.m1()
```

Output

static method

Example:

```
class Math:
   @staticmethod
```

```
    def power(n,p):
        return n**p
    @staticmethod
    def factorial(num):
        fact=1
        for i in range(1,num+1):
            fact=fact*i
        return fact
    @staticmethod
    def isprime(num):
        c=0
        for i in range(1,num+1):
            if num%i==0:
                c+=1
        return c==2

res1=Math.power(5,2)
res2=Math.isprime(9)
res3=Math.factorial(4)
print(res1,res2,res3)
```

Output
25 False 24

Class Reusability
An object oriented application is not developed by defining everything in one class.
Object oriented application is a collection of classes. The content of one class can be used inside another class in different ways

1. Composition (Has-A)
2. Aggregation (Use-A)
3. Inheritance   (IS-A)