To transform method to class level, @classmethod decorator is used.

Syntax:

```
class <class-name>:
        @classmethod
        def <method-name>(cls,arg1,arg2,…):
                statement-1
                statement-2
```

Example:

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


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

Output:

```
instance method
class method
```

Instance method is able access instance variables and class level variables.
Class level method is able access only class level variables but cannot access instance variables.

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

```
    @classmethod
    def m2(cls):
        print(cls.x)


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

Output:

```
100
200
100
```

Example:

```
class Account:
    __minBal=5000

    def __init__(self,a,c,b):
        self.__accno=a
        self.__cname=c
        self.__bal=b
    @classmethod
    def printMinBal(cls):
        print(cls.__minBal)
    def printAccount(self):
        print(f'AccountNo {self.__accno}')
        print(f'CustomerName {self.__cname}')
        print(f'Balance {self.__bal}')
        print(f'Minimum Balance {Account.__minBal}')


Account.printMinBal()
acc1=Account(101,"naresh",50000)
acc2=Account(102,"ramesh",45000)
acc1.printAccount()
acc2.printAccount(
```

Output:

```
5000
AccountNo 101
CustomerName naresh
Balance 50000
Minimum Balance 5000
AccountNo 102
CustomerName ramesh
Balance 45000
Minimum Balance 5000
```

Example:

```
import datetime
class Person:
    def __init__(self,n,a):
        self.__name=n
        self.__age=a
    def getName(self):
        return self.__name
    def getAge(self):
        return self.__age

    @classmethod
    def createPerson(cls,n,dob):  # Factory Methods
        year=datetime.date.today().year
        a=year-dob
        p=Person(n,a)
        return p


p1=Person("naresh",50)
p2=Person("suresh",40)
print(p1.getName(),p1.getAge())
print(p2.getName(),p2.getAge())
p3=Person.createPerson("kishore",2000)
print(p3.getName(),p3.getAge())
```

Output:

naresh 50
suresh 40
kishore 23

What is difference between instance method and class method?

Instance method	Class method
A method defined with first argument as a “self” is called instance method	A method defined with first argument as “cls” is called class method, to transform method to class use @classmethod decorator
Instance method is able access instance variables and class level variables	Class level method is able to access class level variables
Instance method is bind with object name. This method cannot invoked without creating object	Class method is bind with class name. This method can be invoked without creating object
This method performs object level operations.	This method performs class level operations.

Static method
Static methods are global methods; these methods are used to perform global operations.
This method does not have implicit first argument.
To declare a method as static we use @staticmethod decorator.

Syntax:
@staticmethod
def <method-name>(arg1,arg2,arg3,…):
statement-1
statement-2

static method is bind with class name, this method can invoked without creating object.

Example:

```
class Math:
    @staticmethod
    def power(num,p):
        return num**p
```

```
    @staticmethod
    def factorial(num):
        fact=1
        for i in range(1,num+1):
            fact=fact*i
        return fact


res1=Math.factorial(5)
res2=Math.power(4,3)
print(res1,res2)
```

Output:
120 64

Class Reusability
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) → Special type of composition
3. Inheritance (IS-A)