```python
# oops

class Car:
    pass
```

What is a Method?

Method is a set of code and can be invoked at any time.

```python
class Car:
    def start(self):
        print("Car started")
    def stop(self):
        print("car stopped")
    def engine_type(self, engine):
        print(f"This car run on {engine} engine")

mercedes=Car() # created a object mercedes
audi=Car()     #audi is an object of class car

mercedes.start()
```

```
Car started
```

```python
mercedes.stop()
```

```
car stopped
```

```python
mercedes.engine_type('petrol')
```

```
This car run on petrol engine
```

```python
# Task 1
class Calculator:
    def add(self,num1,num2):
        print(f"addition of num1 and num2 = {num1+num2}")

    def subtract(self,num1,num2):
        print(f"subtraction of num1 and num2 = {num1-num2}")

    def multiply(self,num1,num2):
        print(f"multiplication of num1 and num2 = {num1*num2}")

    def divide(self,num1,num2):
        print(f"division of num1 and num2 = {num1/num2}")


calculator=Calculator() #here i am creating object of class #object in
lower case and class name in upper class

calculator.add(2,4)
```

```
addition of num1 and num2 = 6
```

```
calculator.subtract(8,2)

subtraction of num1 and num2 = 6

calculator.multiply(2,4)

multiplication of num1 and num2 = 8

calculator.divide(2,4)

division of num1 and num2 = 0.5

# constructor
class Car:
    # constructor is a special type of method because it always has
the following syntax
    def __init__(self, color,engine):  # properties i want to attach
with car
        self.color_of_car=color
        self.engine_type=engine

    def print_car_info(self):
        print(f"color of car={self.color_of_car}")
        print(f"type of engine={self.engine_type}")

mercedes=Car('silver','petrol')
audi=Car('black','diesel')

audi.print_car_info()

color of car=black
type of engine=diesel



# Task 2

class pizza_makers:  # class is pizza_makers
    def __init__ (self,size,crust): #created a constructor that takes
two parameters
        self.size_of_pizza=size
        self.crust=crust
    def pizza_info(self):  # created a method in a class, which print
details of pizza
        print(f"pizza is {self.size_of_pizza} inches in size with
{self.crust} crust")

small_pizza=pizza_makers(8,"thin")  # object created small_pizza and
large_pizza
large_pizza=pizza_makers(14,"regular crust")

small_pizza.pizza_info()
large_pizza.pizza_info()
```

```
pizza is 8 inches in size with thin crust
pizza is 14 inches in size with regular crust crust



# static keyword (number_of_steering_wheels)

# constructor
class Car:
    # constructor is a special type of method because it always has
the following syntax
    def __init__(self, color,engine):  # properties i want to attach
with car
        self.color_of_car=color
        self.engine_type=engine

    def print_car_info(self):
        print(f"color of car={self.color_of_car}")
        print(f"type of engine={self.engine_type}")

        #@static method
    def number_of_steering_wheels(self):
        print("The car has 1 steering wheel")

mercedes=Car('silver','petrol')
mercedes.print_car_info()
mercedes.number_of_steering_wheels()

audi=Car('black','diesel')
audi.print_car_info()
audi.number_of_steering_wheels()   # this is static common to both
object

color of car=silver
type of engine=petrol
The car has 1 steering wheel
color of car=black
type of engine=diesel
The car has 1 steering wheel

# calculator.py [file name]
# Task 1
class Calculator:
    def add(self,num1,num2):
        print(f"addition of num1 and num2 = {num1+num2}")

    def subtract(self,num1,num2):
        print(f"subtraction of num1 and num2 = {num1-num2}")

    def multiply(self,num1,num2):
        print(f"multiplication of num1 and num2 = {num1*num2}")
```

```
    def divide(self,num1,num2):
        print(f"division of num1 and num2 = {num1/num2}")


from calculator import Calculator  # from module importinga class
(imp)
class Test:
    obj=Calculator()
    obj.add(25,30)

addition of num1 and num2 = 55

# Task

# Question :- create a package demo 1 and inside this package create a
class -weather_info.
# in this class ,create a method-current weather which gives output-it
is very hot outside.
# create another package demo 2 and inside this package ,create a
class - weather _report.
# call the current_weather() method in this class and print the
output.

# demo1.py
class weather_info:
    def current_weather(self,temp):
        print(f"it is very hot outside tempreature is {temp}")

# demo2.py
class weather_report:
    obj=weather_info()
    obj.current_weather(24)

it is very hot outside tempreature is 24

# when demo1.py is different python file

from demo1 import weather_info
class weather_report:
    obj=weather_info()
    obj.current_weather(24)

# static method

class Math:
    @staticmethod
    def Multiply(one, two):
        return one * two

math = Math()
if(12*72 == math.Multiply(12, 72)):
    print("Equal")
```

```
else:
    print("Not Equal")
```

```
Equal
```