In [81]:
```
#Answer 1

#CLASS
#class is a user defined prototype for which objects are made
#Class is like a skeleton of an object.........
#for example :- if i am asked to sit in the car, i know what is the car, but don't know in which car i have to get sitted
#car here is the skeleton

#OBJECTS
#class is like a blueprint and an object is an detailed description of the class
#object consists of the information related to the skeleton like:- what's the colour of the car, which compamy of the car is and etc.........
```

In [82]:
```
#FOR EXAMPLE

class SPS_Students:

    def welcome_msg(self):
        print("welcome to SPS batch 40")
```

In [83]:
```
#now we have a class "SPS_Students"
# so here this is a skeleton with undefined data
# to add data in it we'll create objects
```

In [84]:
```
Harshit = SPS_Students()
```

In [85]:
```
#we have an object named harshit, who is a student of SPS batch
```

In [86]:
```
#HERE WE HAVE A CLASS "SPS_Students" and an object "Harshit"
```

In [87]:
```
Harshit.welcome_msg()
```

```
welcome to SPS batch 40
```

In [ ]:

In [ ]:

In [88]:
```
#ANSWER 2

# 1. Abstraction
# 2. Encapsulation
# 3. Inheritance
# 4. Polymorphism
```

In [ ]:
```

```

In [89]:
```
#ANSWER 3

#__init__ function is used to add data in the class, it is an constructor in python CLASS
```

In [90]:
```
# for example:

class SPSStudents:

    def __init__(self, student_name, student_id):
        self.student_name = student_name
        self.student_id = student_id

    def return_details(self):
        return self.student_name, self.student_id
```

In [91]:
```
harshit = SPSStudents("harshit", 2241)
```

In [92]:
```
harshit.student_name
```

Out[92]:
```
'harshit'
```

In [93]:
```
harshit.student_id
```

Out[93]:
```
2241
```

In [94]:
```
harshit.return_details()
```

Out[94]:
```
('harshit', 2241)
```

In [95]:
```
#here we add an object "harshit", who is a student of SPS Batch, to the CLASS SPSStudents
```

In [ ]:
```

```

In [100]:
```
#ANSWER 4

#self is an instance of the class. we use self in OOPs python because python do not understand the @syntax
#self is used to bind the argumengts given to it, as well as to access the attributes given by us
```

In [ ]:
```

```

In [102]:
```
#ANSWER 5

# inheritance is a concept in OOPs..... inheritance allows us to inherit the arguments and methods of another class

#types of inheritance:-
    #1. single inheritance
    #2. multi inheritance
    #3. multilevel inheritance
    #4. hierarchical inheritance
    #5. hybrid inheritance
```