In [31]:
```
class student:
    def __init__(self, roll_no, standard, section):  #Fixing attributes that this class
 can hold

        #now initialising the attributes

        self.roll_no= roll_no
        self.standard= standard
        self.section= section
        #self.decision= decision
        #defining class-methods

    def topper(self, decision):
        print('Roll_no. {} is {}'.format(self.roll_no, decision ))
```

In [32]:
```
student1=student(1,6,'a')
```

In [33]:
```
student2= student(2,6,'b')
```

In [34]:
```
print(student2.section)
```

```
b
```

In [36]:
```
student1.topper("non-Topper")
student2.topper("Topper")
```

```
Roll_no. 1 is non-Topper
Roll_no. 2 is Topper
```

```
In [4]: dir(student1)
```

```
Out[4]: ['__class__',
         '__delattr__',
         '__dict__',
         '__dir__',
         '__doc__',
         '__eq__',
         '__format__',
         '__ge__',
         '__getattribute__',
         '__gt__',
         '__hash__',
         '__init__',
         '__init_subclass__',
         '__le__',
         '__lt__',
         '__module__',
         '__ne__',
         '__new__',
         '__reduce__',
         '__reduce_ex__',
         '__repr__',
         '__setattr__',
         '__sizeof__',
         '__str__',
         '__subclasshook__',
         '__weakref__']
```

we can use the **init** function to initialise our attributes and therefore it acts basically like a constructor

```
In [ ]: 
```