In [3]:
```
import pandas as pd
```

class method

In [4]:
```
class Employee:
    company="Apple"
    def show(self):
        print(f"The name is {self.name} and the company is {self.company}")
    def changeCompany(cls,newCompany):
        cls.company=newCompany
e1=Employee()
e1.name="Usama"
e1.show()
```

```
The name is Usama and the company is Apple
```

In [5]:
```
class Employee:
    company="Apple"
    def show(self):
        print(f"The name is {self.name} and the company is {self.company}")
    @classmethod
    def changeCompany(cls,newCompany):
        cls.company=newCompany
e1=Employee()
e1.name="Usama"
e1.show()
e1.changeCompany("Tesla")
e1.show()
```

```
The name is Usama and the company is Apple
The name is Usama and the company is Tesla
```

In [6]:
```
print(Employee.company)
```

```
Tesla
```

class method as alternative constructor

In [7]:
```
class Employee:
    def __init__(self,name,salery):
        self.name=name
        self.salery=salery
    @classmethod
    def fromStr(cls,string):
        return cls(string.split("-")[0],string.split("-")[1])
string="Usama-2000"
e=Employee.fromStr(string)
print(e.name)
print(e.salery)
```

```
Usama
2000
```

In [9]:
```
x=[1,2,3,4]
dir(x)
```

Out[9]:
```
['__add__',
 '__class__',
 '__class_getitem__',
 '__contains__',
 '__delattr__',
 '__delitem__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getstate__',
 '__gt__',
 '__hash__',
 '__iadd__',
 '__imul__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__reversed__',
 '__rmul__',
 '__setattr__',
 '__setitem__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'append',
 'clear',
 'copy',
 'count',
 'extend',
 'index',
 'insert',
 'pop',
 'remove',
 'reverse',
 'sort']
```

In [12]:
```
x.__add__
```

Out[12]:
```
<method-wrapper '__add__' of list object at 0x000001876F2B0780>
```

In [21]:
```
class person:
    def __init__(self,name,age):
        self.name=name
        self.age=age
p=person("Usama",22)
print(p.__dict__)
```

```
{'name': 'Usama', 'age': 22}
```

In [24]:
```
print(help(person))
```

```
Help on class person in module __main__:

class person(builtins.object)
 |  person(name, age)
 |
 |  Methods defined here:
 |
 |  __init__(self, name, age)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)

None
```

In [ ]:
```

```