Single Inheritance

In [1]:
```
class Apple:

    manufacturer = "Apple Inc."
    contactWebsite = 'www.apple.com'

    def contactDetails(self):
        print('To contact, please visit ',self.contactWebsite)

#Apple is the base class and Macbook is the derived class
class Macbook(Apple):

    def __init__(self):
        self.yearOfManufacture = 2017

    def manufacturingDetails(self):
        print('This was manufactured by {} in the year {}'.format(self.manufacturer,sel
```

In [2]:
```
macbook = Macbook()
```

In [3]:
```
macbook.manufacturingDetails()
```

```
This was manufactured by Apple Inc. in the year 2017
```

In [4]:
```
macbook.contactDetails()
```

```
To contact, please visit  www.apple.com
```

Multiple Inheritance

In [6]:
```
class OperatingSystem:

    multitasking = True
    name = 'Mac Os'

class Apple:

    manufacturer = 'Apple Inc'
    name = 'Apple'

class Macbook(OperatingSystem,Apple):

    def __init__(self):

        if self.multitasking is True:
            print('This is manufactured by {}'.format(self.manufacturer))
            print('Name is : ',self.name)
```

In [7]:
```
macbook = Macbook()
```

```
This is manufactured by Apple Inc
Name is :  Mac Os
```

In [8]:
```
class OperatingSystem:

    multitasking = True
    name = 'Mac Os'

class Apple:

    manufacturer = 'Apple Inc'
    name = 'Apple'

class Macbook(Apple,OperatingSystem):

    def __init__(self):

        if self.multitasking is True:
            print('This is manufactured by {}'.format(self.manufacturer))
            print('Name is : ',self.name)
```

In [9]:
```
mac = Macbook()
```

```
This is manufactured by Apple Inc
Name is :  Apple
```

In [10]:
```
#When a conflict comes with respect to attribute being same, the order depends on the o
```

Multi-Level Inheritance

In [12]:
```
class MusicalInstruments:
    noOfMajorKeys = 12

class StringInstruments(MusicalInstruments):
    typeOfWood = 'Teakwood'

class Guitar(StringInstruments):

    def __init__(self):
        self.noOfStrings = 6

        print('This instrument has {} keys and the wood is {} and no of strings in {}'.
```

In [13]:
```
music = Guitar()
```

```
This instrument has 12 keys and the wood is Teakwood and no of strings in 6
```

In [14]:
```
#Public access specifier can be accessed by base class and all other derived classes, a
#Protected access specifier can be accessed by base class and all other derived classes
#Private access specifier can be accessed only by your class and not derived classes
```

In [15]:
```
#Python access specifier conventions:
#Public -> memberName
#Protected -> _memberName
#Private -> __memberName
```

In [20]:
```
class Car:
    noOfWheels = 4
    _color = 'Black'
    __yearOfManufacture = 2017
```

```
class BMW(Car):

    def __init__(self):

        print('The no of wheels is {} and the color is {}'.format(self.noOfWheels,self.
        print('Year of manufacture is : ', self._Car__yearOfManufacture)
        #print('Year of manufacture is : ', self.__yearOfManufacture)
        #AttributeError: 'BMW' object has no attribute '_BMW__yearOfManufacture'
```

In [21]:
```
bm = BMW()
```

```
The no of wheels is 4 and the color is Black
Year of manufacture is :  2017
```

In [22]:
```
#Name Mangling on private members:
#__varibaleName->_ClassNAme__variableName
```

In [ ]:
```

```