oops

In [1]:
```
class Book:
    numberOfPages = 145
    author = "anish"
    scope = "To be sold in india"

    def __init__(self,model):
        self.model = model
mybook = Book()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [1], in <cell line: 8>()
      6     def __init__(self,model):
      7         self.model = model
----> 8 mybook = Book()

TypeError: __init__() missing 1 required positional argument: 'model'
```

In [2]:
```
# in above model = instance attribute, author and other 2 are class attributes
```

In [3]:
```
mybook = Book("M1")
print(mybook.numberOfPages)
```

```
145
```

In [4]:
```
print(mybook.model)
```

```
M1
```

In [5]:
```
print(mybook.zone)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [5], in <cell line: 1>()
----> 1 print(mybook.zone)

AttributeError: 'Book' object has no attribute 'zone'
```

In [6]:
```
mybook = Book("Mystery")
print(mybook.model)
```

```
Mystery
```

In [7]:
```
# constructor will be executed just before the creation of an object
```

In [8]:
```
# destructor will be executed just before the destruction of an object
```

oop pillars

inheritance

In [11]:
```
# child classes will have access to parent class
```

In [12]:
```
# and parent class does not access to parent classes
```

In [13]:
```
class School:
    name = "VJHS"

    def __init__(self, age):
        this.age = age

    def getAge():
        return self.age

class SubSchool(School):
    def getParentName():
        return "VJHS"
    def getName():
        return "Anish"

myschool = SubSchool()
print(myschool.getParentName())
print(myschool.getName())
print(myschool.getAge())
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [13], in <cell line: 16>()
     13     def getName():
     14         return "Anish"
---> 16 myschool = SubSchool()
     17 print(myschool.getParentName())
     18 print(myschool.getName())

TypeError: __init__() missing 1 required positional argument: 'age'
```

In [14]:

```
class School:
    name = "VJHS"

    def __init__(self, age):
        this.age = age

    def getAge():
        return self.age

class SubSchool(School):
    def getParentName(self):
        return "VJHS"
    def getName(self):
        return "Anish"

myschool = SubSchool()
print(myschool.getParentName())
print(myschool.getName())
print(myschool.getAge())
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [14], in <cell line: 16>()
     13     def getName(self):
     14         return "Anish"
---> 16 myschool = SubSchool()
     17 print(myschool.getParentName())
     18 print(myschool.getName())

TypeError: __init__() missing 1 required positional argument: 'age'
```

In [22]:
```
class School:
    name = "VJHS"
    def __init__(self, age):
        self.age = age

    def getAge(self):
        return self.age

class SubSchool(School):
#     def __init__(self):
#         print("Child Constructor")
    def getParentName(self):
        return "VJHS"
    def getName(self):
        return "Anish"

myschool = SubSchool(20)
print(myschool.getParentName())
print(myschool.getName())
print(myschool.getAge())
```

```
VJHS
Anish
20
```

In [ ]:
```

```

In [27]:
```
class Parent1:
    def getParentName(self):
        return "Parent1"

class Parent2:
    def getParentName():
        return "Parent2"

class Parent3:
    def getParentName():
        return "Parent3"

class Child(Parent1, Parent2, Parent3):
    def __init__(self):
        self.getParentName()
    def getAllParent(self):
        print("trying to get all parents of this class")
mychild = Child()
print(mychild.getParentName())
```

```
Parent1
```

In [28]:
```
print(mychild.__bases__)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [28], in <cell line: 1>()
----> 1 print(mychild.__bases__)

AttributeError: 'Child' object has no attribute '__bases__'
```

In [29]:
```
print(mychild.__base__)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [29], in <cell line: 1>()
----> 1 print(mychild.__base__)

AttributeError: 'Child' object has no attribute '__base__'
```

In [30]:
```
print(mychild.__Bases__)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [30], in <cell line: 1>()
----> 1 print(mychild.__Bases__)

AttributeError: 'Child' object has no attribute '__Bases__'
```

In [31]:
```
print(mychild.__Base__)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [31], in <cell line: 1>()
----> 1 print(mychild.__Base__)

AttributeError: 'Child' object has no attribute '__Base__'
```

In [33]:
```
for base in Child.__bases__:
    print(base,end=" ")
```

```
<class '__main__.Parent1'> <class '__main__.Parent2'> <class '__main__.Paren
t3'>
```

polymorphism

In [34]:
```
#polymorphism is defined as phenomenon having same function names with differ
```

In [35]:
```
def add(a,b):
    print(a+b)
def add(a,b,c=2):
    print(a+b+c)

add(2,3)
```

```
7
```

In [36]:
```
add(10,20,30)
```

```
60
```

In [37]:
```
def addn(n1,n2,*n3):
    print(n1,n2,n3)

addn(10,20,30,40,50,60)
```

```
10 20 (30, 40, 50, 60)
```

In [38]:
```
def adds(*args):
    sum = 0
    for i in args:
        sum += i
    print(sum)
adds(10,20,30)
```

```
60
```

In [39]:
```
def dojob(a1,a2,a3):
    print("anish")
def dojob(a1):
    print("Mahammed Anish")
dojob(10)
dojob(10,20,30)
```

```
Mahammed Anish
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [39], in <cell line: 6>()
      4     print("Mahammed Anish")
      5 dojob(10)
----> 6 dojob(10,20,30)

TypeError: dojob() takes 1 positional argument but 3 were given
```

In [40]:
```
# by using above method we cannot implement polymorphism
```

In [43]:
```
class Animal:
    def printSomething():
        print("Animals are good by nature")
    def canFly(self,name):
        if(name == "man"):
            print("man cannot fly")
        else:
            print("no animals can fly")

class Cat(Animal):
    def canFly():
        print("cats cannot fly")

class Martan(Animal):
    def canFly():
        print("martan can fly")

myone = Martan()
myone.canFly()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [43], in <cell line: 19>()
     16         print("martan can fly")
     18 myone = Martan()
---> 19 myone.canFly()

TypeError: canFly() takes 0 positional arguments but 1 was given
```

In [44]:
```
myone.canFly("man")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [44], in <cell line: 1>()
----> 1 myone.canFly("man")

TypeError: canFly() takes 0 positional arguments but 2 were given
```

In [45]:
```
class Animal:
    def printSomething():
        print("Animals are good by nature")
    def canFly(self,name):
        if(name == "man"):
            print("man cannot fly")
        else:
            print("no animals can fly")

class Cat(Animal):
    def canFly():
        print("cats cannot fly")

class Martan(Animal):
    def canFly():
        print("martan can fly")

myone = Martan()
myone.canFly("man")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [45], in <cell line: 19>()
     16         print("martan can fly")
     18 myone = Martan()
---> 19 myone.canFly("man")

TypeError: canFly() takes 0 positional arguments but 2 were given
```

In [46]:
```
Cat.canFly()
```

```
cats cannot fly
```

In [47]:
```
# we cannot call parent class method form child class object when the same me
```

Encapsulation

In [49]:
```
class Library:
    book = []
    my_book = {}
    def getAllBooks():
        pass
    def addBook():
        pass
    def getBookByAuthorName():
        pass
    def sortBooksInTopologicalOrder():
        pass
    def getAllSubscribers():
        pass
    def getAllEarnings():
        pass
    def getUnavailableBooks():
        pass
    def mapAuthorWithBook():
        pass

class MyLibrary(Library):
#     It cannot have it's own data structure to store book, authors, subscribe
    def getFavouriteBook():
        pass
    def getFavouriteCustomer():
        pass

lib = MyLibrary()
print(lib)
```

```
<__main__.MyLibrary object at 0x000001E2FB6C3730>
```

In [ ]:

In [103]:

```
class Library:
    book = []
    my_book = {}
    def getAllBooks(self):
        print("The books in library are : ")
        for bk in self.book:
            print(bk,end=" ")

    def addBook(self,bknm):
        self.book.append(bknm)

    def getBookByAuthorName(self,nm):
        if nm not in self.my_book.keys():
            print("no such author found known as :",nm)
        else:
            print("Book =",nm,",and Author = ",self.my_book[nm])

    def sortBooksInTopologicalOrder(self):
        for key in sorted(self.my_book.keys()):
            print(key,"->",self.my_book[key])

    def getAllSubscribers():
        pass
    def getAllEarnings():
        pass
    def getUnavailableBooks():
        pass

    def mapAuthorWithBook(self,atr,bk):
        if atr not in self.my_book.keys():
            self.my_book[atr] = bk
        else:
            self.my_book[atr] += (bk)

class MyLibrary(Library):
#     It cannot have it's own data structure to store book, authors, subscribe
    def getFavouriteBook():
        pass
    def getFavouriteCustomer():
        pass

lib = MyLibrary()
lib.addBook("anish")
lib.addBook("mahammed")
lib.mapAuthorWithBook("anish","The World Of Coding ")
lib.mapAuthorWithBook("anish","The World Of narrator ")
lib.mapAuthorWithBook("magicman","The World Of magic")
lib.mapAuthorWithBook("warner","wanted")
lib.getBookByAuthorName("The World Of Coding")
lib.getBookByAuthorName("The World")
lib.getAllBooks()
print()
print()
print("The books in Topological Sorting Order are: ")
```

```
lib.sortBooksInTopologicalOrder()
```

```
no such author found known as : The World Of Coding
no such author found known as : The World
The books in library are :
anish mahammed

The books in Topological Sorting Order are:
anish -> The World Of Coding The World Of narrator
magicman -> The World Of magic
warner -> wanted
```

In [ ]: