In [ ]:
```
#Agenda of Today :
                      1. Continue to Oops
                          Data Encapusulation (Data Hiding ) in python
                          Ploymorphism   (Method overloading and Method Overriding)
                          Data Abstraction.
                     2.  Data Science Packages
                          i.  Numpy
                          ii. Pandas
                         iii. Matplotlib
```

In [ ]:
```
#Difference b/w Data Encapsulation and Data Hiding:
   Encapsulation means wraping the implementation of data member and
   methods inside the class.

   Its just enveloping the complexity of data

   Data Hiding means protecting the members of a class
   from an illegal and unauthorized users.
```

In [ ]:
```
#Data hiding:
  Its means making the data private so that data will not be
  accessble by the others.
  It can be accessed only in the class where it is decalred.
#Note: to make class variable as private by using double underscore.
```

In [3]:
```
#Example:
class Myclass:
    __a= 10              #class variable or private variable
    def add(self,b):
        sum = self.__a+b
        print(sum)

ob = Myclass()
print(ob.add(20))  #its just accessing the method of class
print(ob.__a)
```

```
30
None

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-3-de3dfca6c185> in <module>
      8 ob = Myclass()
      9 print(ob.add(20))  #its just accessing the method of class
---> 10 print(ob.__a)

AttributeError: 'Myclass' object has no attribute '__a'
```

In [ ]:
```
#Ploymorphism in oops:
   - Its a Greek word.
      Ploy - means many
      morph - means having many forms.
- It refers to the functions having the same names but carrying out different funtional
```

In [7]:
```
#Example:
class audi:
    def desc(self):
        print("This is desc of AUDI CAR")
class BMW:
    def desc(self):
        print("this is desc of BMW CAR")
a = audi()
b = BMW()
for car in(a,b):
    car.desc()
```

```
This is desc of AUDI CAR
this is desc of BMW CAR
```

In [ ]:

```
#Data Abstraction:
     We use abstraction in oops for hiding the interal details or implementions of func
     and functinalaties only.
#Note:
     Any class with atleast one abstract function is called as abstract class.
     To create an abstract class
          we need to import ABC class from abc module.
          ABS stands for Abstact Base Class.
    #Data Abstraction can be done by using inheritance.
```

In [ ]:

```
#syntax for how to import ABC class
from abc import ABC
class Abs_class(ABC):
    #body of the class
    @abstact method
```

In [9]:

```
#example:
from abc import ABC,abstractmethod

class car(ABC):                           #abstract class
    def __init__(self,name):
        self.name = name

    def desc(self):                    #normal class method
        print("this is desc of class car")


    @abstractmethod                  #abstract method
    def price(self,x):
        pass

class new(car):
    def price(self,x):
        print(f"The {self.name}'s price is {x} lakhs")

ob = new("BMW")   #child class object
ob.desc()
ob.price(50)
```

```
this is desc of class car
The BMW's price is 50 lakhs
```

In [ ]:

```
#Data Science Packages
                       1.  Numpy (Num+py)
                           Numrical data + python

                      2. Pandas- Its used packages used
                         for
    (i)   Data Manipulation (modification,updation)
    (ii)  Data analysis (To Know the Complete knowledge about data including the theme
    (iii) Data Cleaning  (To remove unneccessary data from data sets.)

3. Matplotlib - To represent complex data with 2d- Graphics tools.
         used tools:
                  1. plots
                  2. bar graphs
                  3. pie charts
                  4. Histograms
                  5. Scatter plots
                  6. area graphs.
```

In [ ]:

```
#Numpys in Python:
Numpy is open source library package used for scientific computing.
   Its meanly deals  with
      (i) mathematical
      (ii) scientific
       (iii) engineering data
        (iv) statical data
        (v) multi-dimesional arrays and matric multiplication.
```

In [ ]:

```
#Note:
Numpy contains a powerful n-dimensional array object.
```

In [ ]:

```
#what is numpy array?
   - numpy array is powerful n-dimensional  array object which is in the
     form of rows and columns
   - We can initiliaze numpy arrays from nested python lists.

```

In [ ]:

```
#How to install numpy package?
pip install numpy
```

In [10]:

```
#how to import and use numpy package ?
import numpy as np
```

In [14]:

```
#how to create numpy arrays?
import numpy as np
a=np.array([1,2,3])     #single-dimensional numpy array
a
```

Out[14]:

```
array([1, 2, 3])
```

In [15]:

```
#to know the shape of array
a.shape
```

Out[15]:

```
(3,)
```

In [17]:

```
#How to 2D- multi dimensional arrays:
a=np.array([(1,2,3),(4,5,6)])
a.shape
```

Out[17]:

```
(2, 3)
```

In [22]:

```
#Numpy array attributes:
a1 = np.array([1,2,3,4,5,6,7,8,9])
print(a1)
```

```
a2 = np.array([2.3,6,9,3.5,9.0,1.3])
print(a2)
print(a1.dtype)
print(a2.dtype)
```

```
[1 2 3 4 5 6 7 8 9]
[2.3 6.  9.  3.5 9.  1.3]
int32
float64
```

In [25]:

```
#shape of numpy array:
a1 = np.array([1,2,3,4,5,6,7,8,9])   #1-d array
a2 = np.array([[1,2,3],[4,5,6]])     #2-D array
a3 = np.array([[[1,3,5],[5,6,7]],[[8,9,3],[6,7,8]]]) #3-d array
print(a1.shape)
print(a2.shape)
print(a3.shape)
```

```
(9,)
(2, 3)
(2, 2, 3)
```

In [29]:

```
#dimension of an numpy:
#ndim():
print(a1.ndim)
print(a2.ndim)
print(a3.ndim)
```

```
1
2
3
```

In [44]:

```
#reshape of array:
a1 = np.array([1,2,3,4,5,6,7,8])
print(a1.shape)
print("before reshape",a1)
a2 = a1.reshape(4,2)
print("after reshape",a2)
print(a2.shape)
```

```
(8,)
before reshape [1 2 3 4 5 6 7 8]
after reshape [[1 2]
 [3 4]
 [5 6]
 [7 8]]
(4, 2)
```

In [48]:

```
#resize of array: (its modifies existing shape and array itself.)
a1 = np.array([1,2,3,4,5,6,7,8])
print(a1)
print(a1.shape)
a1.resize(3)   #resize the no of elements from original array
print(a1)
print(a1.shape)
a1.resize(2,4)
print(a1)
print(a1.shape)
a1.resize(3,3)
print(a1)
print(a1.shape)
```

```
[1 2 3 4 5 6 7 8]
```

```
(8,)
[1 2 3]
(3,)
[[1 2 3 0]
 [0 0 0 0]]
(2, 4)
[[1 2 3]
 [0 0 0]
 [0 0 0]]
(3, 3)
```

In [56]:

```
#zero array: ones array: full array:  eye array
import numpy as np
a1=np.zeros(3)
a2=np.zeros((2,4),dtype="int64")
print(a1.dtype)
print(a2.dtype)
print(a1)
print(a2)
```

```
float64
int64
[0. 0. 0.]
[[0 0 0 0]
 [0 0 0 0]]
```

In [59]:

```
#ones array:
np.ones(3,dtype="int64")
np.ones((6,5),dtype="int64")
```

Out[59]:

```
array([[1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1]], dtype=int64)
```

In [63]:

```
#full array: full(dimension,specified number)
np.full(3,100)
np.full((3,9),500)
np.full((3,9),5)
```

Out[63]:

```
array([[5, 5, 5, 5, 5, 5, 5, 5, 5],
       [5, 5, 5, 5, 5, 5, 5, 5, 5],
       [5, 5, 5, 5, 5, 5, 5, 5, 5]])
```

In [76]:

```
#eye array:
np.eye(6,dtype="int64")
n=np.eye(5,k=0,dtype="int64")
n
```

Out[76]:

```
array([[1, 0, 0, 0, 0],
       [0, 1, 0, 0, 0],
       [0, 0, 1, 0, 0],
       [0, 0, 0, 1, 0],
       [0, 0, 0, 0, 1]], dtype=int64)
```

In [ ]:

```

```