DATA SCIENCE

DATA SCIENCE: - is a branch of computer science where we study how to store, use and Analyze data for deriving information from it.

INDEX

NUMPY

What is NumPy?

- NumPy is a python library used for working with array.
- It also has function for working in domain of linear algebra and matrices.
- NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
- NumPy stands for Numerical Python.

Why use NumPy?

- In Python we have lists that serve the purpose of arrays, but they are slow to process.
- NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.
- The array object in NumPy is called ndarray; it provides a lot of supporting functions that make working with ndarray very easy.
- Arrays are very frequently used in data science, where speed and resources are very important.

Install NumPy?

Install NumPy using following command: -

```
C:\Users\User_Name>pip install numpy
```

ARRAY:-

Assigning and declaring array.

Program:-

```
import numpy as np
a=np.array([[10,20,30,40,50],[1,2,3,4,5],[56,75,42,24,78]])

print(a)    #printing array.

print("using for loop")
for i in a:
   print (i,end="\t")  #prining array using for loop.

print(a.size)   #print size of array.
print(a.shape)  #print shape of array, return in array.

b=a.shape
print(b)    #print array elements using shape
print(b[0]) #and for loop
for i in range(b[0]):
   for j in range(b[1]):
      print(a[i][j],end=" ")
```

0D ARRAY: - or scalars are the elements in an array, single value in an array is a 0-D array.

Program:-

```
a=np.array(56)
print(a)
print(a.ndim)   #check dimension of array.
```

1D ARRAY: - An array that has 0D arrays as their element is called uni-dimensional or 1D array. These are the most common and basic arrays.

Program:-

```
a=np.array([23,45,64,69,41])
print(a)
print(a.ndim)   #check dimension of array.
```

program2:-

```
l=([34,54,23,56,76])    #list of elements
a=np.array(l)   #inserting list in array
print(a)
print(a.ndim)   #check dimension of array
```

2D ARRAY: - An array has 1D arrays as its elements is called 2D array. These are often used to Represent matrix or 2nd order tensors.

Program:-

```
l=[[1,2,3],[6,7,8]] #list of elements
a=np.array(l)   #inserting list in array
print(a)
print(a.ndim)   #check dimension of array
```

3D ARRAY: - An arrays that has 2D arrays (matrices) as its elements is called 3D array. These are often used to represent a 3rd order of tensor.

Program:-

```
a=np.array([[[23,45],[67,49]],[[65,21],[98,27]]])
print(a)
print(a.ndim)   #check dimension of array
```

HIGHER DIMENSIONAL ARRAYS: - An array can have any number of dimensions, when Array is created; you can define the number of dimensions by using the **ndmin** argument.

Program:-

```
a=np.array([1,2,3,4,5],ndmin=5)    #defining dimensions using ndmin
print(a)
print(a.ndim)   #check dimension of array
```

ARRAY INDEXING: -

1D array: -

```
import numpy as np

a=np.array([3,4,5,2,7])
print(a)
print(a[2]) #accessing single element
print("using for loop")
for i in range(len(a)): #print array
   print(a[i])
```

2D array: -

```
import numpy as np

#2d array
a=np.array([[3,4,5],[8,7,9]])
print(a)
print(a[1][1])  #accessing single element
```

NUMPY DATATYPES:-

NumPy offers a wider range of numerical data type that what is available in Python. Here the list of most commonly used numeric data types in NumPy:

1) Int8, int16, int64, int32:- signed integer types with different bit size.
2) Uint8, uint16, uint32, uint64:- unsigned integer.
3) Float32, float64:- floating-point types with different precision levels.
4) Complex64, complex128:- complex number types with different precision levels.

Program:-

```
import numpy as np

a=np.array([1,2,3,4,5],dtype='float32')
#assign data-type to array
print(a)
print(a.dtype)  #printing data-type of array
print(a.shape)  #shape of array
print(a.size)   #size of array
```

RESHAPING ARRAY:-

- Reshaping is a changing shape of array.
- The shape of array is the number of element in each dimension.

- By reshaping we can add or remove dimensions or change number of elements in each dimension. Reshape(row,column)
- To reshape array we can use reshape function.

Program:-

```
import numpy as np

a=np.array([1,2,3,4,5,6])
print(a)
print(a.ndim)   #dimension of array a
print(a.shape)  #shape of array a
arr=a.reshape(3,2)  #reshaping array a
print(arr)  #print new array
print(arr.shape)    #shape of new array
```

ITERATING ARRAY USING NDITER (): -

- The function nditer () is a helping function that can be used from very basic to very advanced iterations. It solves some basic issue which we face in iteration.
- In basic for loops, iterating through each scalar of an array we need to use n for loops which can be difficult to write for array with very high dimensionality.

Program:-

```
import numpy as np

a=np.array([[[16,24],[33,47]],[[59,62],[74,88]]])
print(a.shape)
for x in np.nditer(a):  #print elements using nditer
   print(x)
```

NUMPY JOINING ARRAY:-

Program:-

```
import numpy as np

a=np.array([[1,2,3],[4,5,6]])
b=np.array([[10,21,32],[43,54,67]])
arr=np.concatenate((a,b))   #join array a&b
print(arr)
print(arr.shape)
arr2=np.concatenate((a,b),axis=1)   #join array a&b in axis
print(arr2)
```

SPECIAL FUNCTION FOR CREATING ARRAY: -

Zeros function: - return new array of given shape and type filled with zero.

Program:-

```
import numpy as np

a=np.zeros((5,5))   #creating array with zero
print (a)
```

Ones function: - it creates a numpy array return a new array of given shape and type.

Program:-

```
a=np.ones((5,5))   #creating array with one
print (a)
```

Empty array: -return a new array of given shape and type, without initializing entries.

Program:-

```
a=np.empty((3,2),dtype='int8') #creating empty array
print (a)
print(a.dtype)
```

fill method: - it replace array elements with number that passed as argument.

```
a.fill(number)
print(a)
```

Arange: - return evenly spaced values within a given interval arrange can be called with a varying number of positional arguments.

Program:-

```
a=np.arange(1,15,3) #arrange array element
print (a)
```

Generator: - class implementing the entire random number distributions, default constructor for generator.

Program:-

```
a=np.random.rand(5) #random elements
print(a)
```

Linspace: - return evenly spaced number over a specified interval. Returns numbers evenly spaced samples calculated over the interval. [start, stop].

Program:-

```
a=np.linspace(1,20,10)  #evenly space number
print(a)
```

MEAN: - find mean of array.

```
a=np.array([23,45,12,45,67])
Mean1=a.mean()
print(Mean1)
```

MATPLOTLIB

WHAT IS:-

- Matplotlib is low level graph plotting library in python that serves as a visualization utility.
- Matplotlib was created by John D. Hunter.
- Matplotlib is open source and we can use it freely.

Install Matplotlib: -

Install Matplotlib using following command

```
C:\Users\Your Name>pip install matplotlib
```

PYPLOT:-

Program:-

```
import matplotlib.pyplot as plt
import numpy as np

x=np.array([10,20,30,40,50])
y=np.array([1,2,3,4,5])
plt.plot(x,y)   #input arrays
plt.show()  #show graph
```

PLOTTING X & Y POINTS: -

- The plot() function is used to draw points(marks) in diagram.
- By default this function draws lines from point to point.
- The function takes parameters for specifying points in the diagram.
- Parameter 1 is an array containing the points on the X-axis.
- Parameter 2 is an array containing the points on the Y-axis.

Program:-

```
import matplotlib.pyplot as plt
import numpy as np
x=np.array([10,20,30,40,50])
y=np.array([1,2,3,4,5])
plt.xlabel('x label')   #name on x axis
plt.ylabel('y label')   #name on y axis
plt.title("graph name") #show graph name
plt.plot(x,y,'-')   #input arrays
plt.show()  #show graph
```

symbol	Line	symbol	Line	symbol	Line	symbol	Line
‘-’	Straight line	‘d’	Diamond thin	‘x’	X marks	‘v’	Triangle down

‘--’	Dashed line	‘h’	Hexagon	‘X’	X fill	‘<’	Triangle left
‘o’	Disk	‘^’	Triangle up	‘+’	Plus	‘1’	Tri down
‘*’	Star	‘>’	Triangle right	‘P’	Plus fill	‘3’	Tri left
‘.’	Point	‘2’	Tri up	‘D’	Diamond	‘|’	Vertical line
‘s’	Square	‘4’	Tri right	‘p’	pentagon		

MARKER:-

```
import matplotlib.pyplot as plt
import numpy as np

x=np.array([10,20,30,40,50])
y=np.array([1,2,3,4,5])

plt.plot(x,y,'--',marker='h')   #input marker
plt.show()  #show graph
```

Marker size: - you can use keyword argument markersize or shorter version as ms to set size of marker.

Example:-

```
plt.plot(x,y,marker='D',ms=15)  #marker size
plt.show()  #show graph
```

Marker color: -

Marker edge color: - you can use keyword argument markeredgecolor or shorter mec to set the color of edge of the marker.

Example:-

```
plt.plot(x,y,marker='D',mec='r')    #marker edge color
plt.show()  #show graph
```

Marker face color: - you can use keyword argument markerfacecolor or the shorter mfc to set color of face.

Example:-

```
plt.plot(x,y,marker='D',mfc='r',ms=10)    #marker face color
plt.show()
```

LINES:-

Line style: - To plot the line you can use linestyle keyword argument or shorter ls, to change the style of line.

Program:-

```
import matplotlib.pyplot as plt
import numpy as np

y=np.array([5,3,7,2,8])
plt.plot(y,ls='-.') #line style
plt.show()
```

Symbol	Name	Symbol	Name	Symbol	Name
‘-’	Solid	‘:’	Dotted	‘ ’	none
‘--’	Dashed	‘-.’	Dashdot		

Line color and width:- color keyword used for line color and linewidth keyword shorter version lw use for width of line.

Program:-

```
import matplotlib.pyplot as plt
import numpy as np
a=np.array([10,20,30,40,50])
a2=np.array([10,80,30,70,20])
b=np.array([5,10,15,20,25])
plt.plot(b,a,ls='-',color='b',lw='5')   #color and line width
plt.plot(b,a2,ls='-',color='r',lw='5')  #color and line width
plt.show()
```

GRID: - use this function for plot grids

```
plt.plot([1,2,3,4,5],[23,10,8,4,12])
plt.grid()
plt.show()
```

Axis plotting: -

plt.grid(axis=’x’) plot grid on x axis.
plt.grid(axis=’y’) plot grid on y axis.

Line properties: -

```
plt.grid(color="red",linestyle="--",linewidth=1)
```

BAR CHART: - with the help of barchart we can plot a bar chart. For that purpose bar function is used.

Program:-

```
import matplotlib.pyplot as plt
import numpy as np
x=np.array(['hp','lenovo','dell','tesla','apple'])
y=np.array([1234,4322,3456,7544,4556])
plt.xlabel("company")   #name of x label
plt.ylabel("profit")    #name of y label
plt.title("profit 2023")    #chart title
plt.bar(x,y,color='g')  #plot bar
plt.show()
```

Multi color bar:-

```
import matplotlib.pyplot as plt
fruits=['apple','blueberry','cherry','orange']
count=[40,100,30,55]
bar_colors=['maroon','blue','red','orange'] #color list
plt.title('fruits supply')  #title
plt.bar(fruits,count,color=bar_colors)  #assign color list
plt.show()
```

PIE CHARTS:-

```
import matplotlib.pyplot as plt
l=['parle','monaco','sunfeast','good day']  #label list
c=['gold','hotpink','green','orange']   #color list
s=[300,400,60,30]   #data
plt.pie(s,labels=l,colors=c)    #pie chart
plt.legend()    #show inventory
plt.show()
```

LEGEND: - legend for bars and plot

```
plt.plot([1,2,3,4,5],[23,45,12,56,45])
plt.plot([1,2,3,4,5],[15,35,1,34,5])
plt.legend(["name1", "name2"],title="data speed",loc="upper left", fontsize=20,title_fontsize=30)
plt.show()
```

SUBPLOTTING: - it is used for plot multiple graph in same method. It first divide figure in row and column then add plotting number in graph.

```
plt.subplot(2,1,1)
plt.plot([1,2,3,4,5],[23,10,8,4,12],color="red")

plt.subplot(2,1,2)
plt.plot([1,2,3,4,5],[12,34,5,6,3],color="green")

plt.show()
```

Syntax: - plot figure in row, column

plt.subplot(figure row, figure column, graph plotting number)

example: -

```
plt.subplot(2,2,1)
plt.plot([1,2,3,4,5],[23,10,8,4,12],color="red")

plt.subplot(2,2,2)
plt.plot([1,2,3,4,5],[12,34,5,6,3],color="green")
plt.subplot(2,2,3)
plt.plot([1,2,3,4,5],[23,10,8,4,12],color="red")

plt.subplot(2,2,4)
plt.bar([1,2,3,4,5],[12,45,8,56,5])

plt.show()
```

PANDAS

WHAT IS PANDAS:-

- Pandas is python library used for working with data set.
- It has functions for analyzing, cleaning, exploring and manipulating data.
- The name “pandas” has a reference to both “panel data” and “python data analysis” and was created by Wes McKinney in 2008.

Install pandas: -

Install pandas using following command
C:\Users\Your Name>pip install pandas

PANDAS SERIES:-

- A pandas series is like a column in a table.
- It is a one-dimensional array holding data type of any type.

Program:-

```
import pandas as pd

l=[1,2,3,4,5]
indx=['l','m','n','o','p']  #index list
s=pd.Series(l,index=indx)   #list in series with index

print(s)
for i in s:
   print(i)
```

Example:-

```
d={'name':'xyz','mob':3234546432,'dob':'24-8-2002'}
s=pd.Series(d)  #dictionary in series
print(s)
```

Pandas with numpy and Matplotlib:-

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

a=np.array([1,2,3,4,5])
b=np.array([10,20,30,40,50])
s=pd.Series(a)
print(s)
plt.bar(s,b)
plt.show()
```

PANDAS DATAFRAMES:-

A pandas dataframe is a two-dimensional data structure like a 2d array, or table with row and column.

Program:-

```
import pandas as pd

l=[[1,2,3],[11,12,13],[21,22,23]]
df=pd.DataFrame(l)
print(df)   #list in dataframe
df2=pd.DataFrame(l,columns=['A','B','C'])
print(df2)  #naming columns
df3=pd.DataFrame(l,index=['X','Y','Z'])
print(df3)  #naming index
df4=pd.DataFrame(l,index=['X','Y','Z'],columns=['A','B','C'])
print(df4)  #naming index and columns
```

Operations with dataframes:-

```
import numpy as np
import pandas as pd

a=np.array([[1,2,3],[11,12,13],[21,22,23]])
df=pd.DataFrame(a,columns=['1','2','3'],index=['A','B','C'])
print(df)

'''Print Data column wise'''
print(df['1'])
print(df['2'])
print(df['3'])
'''Print Data row wise'''
print(df.loc['A'])
print(df.loc['B'])
print(df.loc['C'])

df2=pd.DataFrame(df,dtype=float)
print(df2)  #data type float
df3=pd.DataFrame(df,dtype=complex)
print(df3)  #data type complex

print(df)
df4=df+(df*50/100)  #show 50%+ value
print(df4)

print(df)
print(df2)
```

```
df5=df+df2
print(df5)  #addition of dataframes

df6=df*df2
print(df6)  #multiplication of dataframes
```

Example:-

```
d={'name':['ram','shyam','vishnu','krishna'],
  'age':[12,13,14,15],
  'sal':[25,27,29,31]}
df=pd.DataFrame(d,index=[1,2,3,4])
print(df)   #dictionary in dataframe
```

Example2:-

```
l=[[12,13,14,15],list("abcd"),[55,66,77,88]]
df=pd.DataFrame(l,columns=['tata','mahindra','bajaj','yamaha'],
          index=['Sr.No','company grade','income'])
print(df)   #list in dataframe
print(df['tata'])   #print column
print(df.loc['Sr.No'])  #print row
```

Example:-

```
l=[[55,57,58,51,59],[42,58,67,52,47],[31,72,58,88,95],
  [72,68,78,54,83],[92,75,87,95,98],[87,54,23,46,78]]
df=pd.DataFrame(l,columns=['s1','s2','s3','s4','s5'],
          index=[1,2,3,4,5,6])
print(df)   #print dataframe with index and columns
```

PANDAS STATISTICS:-

Performing various complex statistical operations in python can be easily reduced to single line commands using pandas functions for statistics.

1. **Mean: -** calculating the mean or average value by using DataFrame / Series mean() method.
   Example: -
   ```
   n=np.array([1,2,3,40,5,6,7,10,8,9,10,10])
   df=pd.DataFrame(n)
   print('mean')
   mean=df.mean()
   print(mean)
   ```

2. **Median: -** calculates the median value by using DataFrame / series median() method.
   Example: -
   ```
   print('median')
   median=df.median()  #median function
   print(median)
   ```

3. **Mode: -** calculates the mode or most frequent value by using DataFrame / series mode() method.

Example: -

```
print('mode')
mode=df.mode()  #mode function
print(mode)
```

4. **Count: -** calculates the count or frequency of non-null values by using DataFrame / series count() method.

Example:-

```
print('count')
count=df.count()  #count function
print(count)
```

5. **Standard Deviation: -** calculates the standard deviation of values by using DataFrame / series std() method.

Example:-

```
print('standard deviation')
std=df.std()  #standard deviation function
print(std)
```

6. **Max: -** calculates the maximum value by using DataFrame / series max() method.

Example:-

```
print('max')
mx=df.max()  #max function
print(mx)
```

7. **Min: -** calculates the minimum value by using DataFrame / series min() method.

Example:-

```
print('min')
mn=df.min()  #min function
print(mn)
```

8. **Describe: -** summarizes general descriptive statistics using DataFrame / Series describe() method.

Example:-

```
print('describe')
des=df.describe()   #describe function
print(des)
```

READ FILES

READ CSV FILE:-

```
import pandas as pd
import matplotlib.pyplot as plt

df=pd.read_csv("files/income.csv")

print(df)   #print csv data
print(df.head())   #print first 5 value
print(df.tail())   #print last 5 value

print('mean of income')
m=df['income'].mean()
print(m)    #print mean of income

l=[]
for i in df['income']:
   r=i-m
   l.append(r)
print(l)

below=0;
above=0;
for i in l:
   if(i<=0):
      below+=1
   if(i>0):
      above+=1

print("person below avarage income ",below)
print("person above avarage income ",above)

plt.bar(df['sr'],df['income'])
plt.show()
```

READ EXCEL FILE:- to read excel file need to install openpyxl

```
import pandas as pd
import openpyxl

df=pd.read_excel('files\incomebook2023.xlsx')
print(df)
```

CONVERT DATAFRAME TO CSV & EXCEL FILE:-

```
import pandas as pd
import numpy as np
import openpyxl

a=np.array([['rose','red',30],
          ['lotus','pink',50],
          ['mogra','white',10],
          ['chafa','yellow',40]])

df=pd.DataFrame(a,columns=['name','colors','price'])
print(df)

df.to_csv('flowerprice.csv')    #convert to csv
df.to_excel('priceflower.xlsx') #convert to xlsx
```

MISSING DATA

CHECKING MISSING VALUES:-

In order to check missing values in Pandas Dataframe / Series, isnull() and notnull() function is used. Both functions are used to check whether value is NaN or not. Isnull() function return true if value is null, notnull() function return true if value is not null.

Example:-

```
import numpy as np
import pandas as pd

dt={'first score':[100,90,np.nan,95],
     'second score':[30,40,56,np.nan],
     'third score':[np.nan,70,80,56]}

df=pd.DataFrame(dt)
print(df)

a=df.isnull()   #print true and false
print(a)

b=df.notnull()  #print true and false
print(b)

c=df.dropna()   #drop rows that have null value
print(c)

r=df.fillna(1)  #fill null with value
print(r)
```

Checking missing values from file:-

```
import pandas as pd

a=pd.read_csv('book1.csv')
print(a)

s=a['gender'].isnull()
print(s)
```