CYCLE 1

Results and Observations

```
Initial array
 [1 2 3 4 5 6 7 8 9]

After dimension change
 [[1 2 3]
 [4 5 6]
 [7 8 9]]

Array into list
 [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

CHANGING DIMENSION OF AN ARRAY

Program No: 1 **Date:**29-08-22

AIM: Write a NumPy program to change dimension of an array and convert the numpy array into a list.

Theoretical Support

NumPy arrays have an attribute called shape that returns a tuple with each index having the number of corresponding elements.

To convert a NumPy array (ndarray) to a Python list use ndarray.tolist() function, this doesn't take any parameters and returns a python list for an array. While converting to a list, it converts the items to the nearest compatible built-in Python type

Code

```
import numpy as np
x=np.array([1, 2, 3, 4, 5, 6, 7, 8,9])
print("Initial array\n",x)
x.shape=[3,3]
print("\nAfter dimension change\n",x)
y=x.tolist()
print("\nArray into list\n",y))
```

Inference

We can change the dimension of an array using array attribute shape.Dimension can be changed according to the number of elements in the array.Array can be converted into list using tolist().

Results and Observations

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
Update sixth value to 11
[ 0.  0.  0.  0.  0.  0. 11.  0.  0.  0.]
```

CREATE AND UPDATE A NULL NUMPY ARRAY

Program No: 2 **Date:**29-08-22

AIM: Write a NumPy program to create and update a null numpy array.

Theoretical Support

Numpy.zeros return a new array of given shape and type,filled with zeros.

Code

```
import numpy as np
x = np.zeros(10)
print(x)
print("Update sixth value to 11")
x[6] = 11
print(x)
```

Inference

np.zeros() used for creating null array.Parameter specifies the number of elements that we want to create.Element of an array can be updated using the index value.index start from zero.

Results and Observations

```
comparing array x with y

Array x
 [[12 16]
 [ 5 10]]

Array y
 [[ 8 23]
 [14  8]]

Greater
[[ True False]
 [False  True]]

Greater_Equal
[[ True False]
 [False  True]]

Less
[[False  True]
 [ True False]]

Less_Equal
[[False  True]
 [ True False]]
```

EXERCISE 1: INTRODUCTION TO NUMPY CO1

ELEMENT-WISE COMPARISON OF ARRAYS

Program No: 3 **Date:**29-08-22

AIM: Write a NumPy program to create an element-wise comparison(greater, greater_equal,less and less_equal) of two given arrays.

Theoretical Support

numpy.greater()-To compare and return True if an array is greater than another array.

numpy.greater_equal()-checks whether the elements in a given array (first argument) is greater than or equal to a specified number(second argument).

numpy.less()-function in Python is used to check, one by one, if the elements of the array x1 are less than the elements of another array x2 that is of the same shape.

numpy.less_equal()-function is used to return the truth value of (x1 =< x2) element-wise.

Code

```
import numpy as np
x = np.array([[12,16],[5,10]])
y = np.array([[8,23],[14,8]])
print("comparing array x with y")
print("\nArray x\n",x)
print("\nArray y\n",y)
print("\nGreater")
print(np.greater(x, y))
print("\nGreater_Equal")
print(np.greater_equal(x, y))
print("\nLess")
print(np.less(x, y))
print("\nLess_Equal")
print(np.less_equal(x, y))
```

Inference:

numpy comparison functions helps to find the element-wise greater,greater than,lesser,lesser than values of two arrays.Standard mathematical functions for fast operations on entire arrays of data without having to write loops.

Results and Observations

```
Array of all the even integers from 30 to 70

[30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70]
```

ARRAY CREATION

Program No: 4 **Date:**29-08-22

AIM: Write a NumPy program to create an array of all the even integers from 30 to 70.

Theoretical Support

NumPy arange() is one of the array creation routines based on numerical ranges. It creates an instance of ndarray with evenly spaced values and returns the reference to it.

Code

```
import numpy as np
x=np.arange(30,71,2)
print("Array of all the even integers from 30 to 70\n")
print(x)
```

Inference

Array of even numbers are created easily using numpy.arrange().we can create array of any range of values according to the parameter specified inside the arrange() function.

Results and Observations

```
3 x 3 Identity Matrix

[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

IDENTITY MATRIX

Program No: 5 **Date:**02-09-22

AIM: Write a NumPy program to create a 3x3 identity matrix

Theoretical Support

numpy.identity(n, dtype = None) : Return a identity matrix i.e. a square matrix with ones on the main diagonal.

Code

```
import numpy as np
x=np.identity(5)
print('3 x 3 Identity Matrix\n')
print(x)
```

Inference

identity matrix are created using identity() function according to the parameter.
Identity(5)-creates a matrix with 5 rows and 5 columns with main diagonal elements as 1 and other elements as 0.

Results and Observations

```
Original array:
[[0 1]
 [2 3]]
After loading, content of the text file:
[[0. 1.]
 [2. 3.]]
```

ARRAY TO TEXT FILE

Program No: 6 **Date:**02-09-22

AIM: Write a NumPy program to save a given array to a text file and load it

Theoretical Support

numpy savetxt enables you to save a Numpy array to a text file.

Python **numpy loadtxt()** function is used to load the data from a text file and store them in a ndarray.

Code

```
import numpy as np
import os
x = np.arange(4).reshape(2,2)
print("Original array:")
print(x)
header = 'col1 col2'
np.savetxt('temp.txt', x, fmt="%d", header=header)
print("After loading")
result = np.loadtxt('temp.txt')
print(result)
```

temp.txt

```
0 1
2 3
```

Inference

Read and wirte contents of a file without using normal file functions.

Results and Observations

```
Array x
 [12 16]

Array y
 [12 23]

After comparison
 [ True False]
```

ARRAY EQUALITY

Program No: 7 **Date:**02-09-22

AIM: Write a NumPy program to check whether two arrays are equal (element wise) or not.

Theoretical Support

The **equal()** function is used to return (x1 == x2) element-wise. Input arrays of the same shape

Code

```
import numpy as np
x=np.array([12,16])
y=np.array([12,23])
print("Array x\n",x)
print("\nArray y\n",y)
print("\nAfter comparison\n",np.equal(x,y))
```

Inference

We can compare(element-wise) every elements of a array with another array without using loops.

Results and Observations

```
Enter the elements of first matrix- 2 X 2
2 7 8 3
Enter the elements of second matrix- 2 X 2
2 7 2 5

First Matrix
 [[2 7]
 [8 3]]
Second matrix
 [[2 7]
 [2 5]]

Dot product of first and second matrix

array([[18, 49],
       [22, 71]])
```

EXERCISE 2:MATRIX OPERATIONS(USING VECTOIZATION) AND TRANSFORMATIONS **CO1**

DOT PRODUCT OF MATRIX

Program No: 8 **Date:**05-09-22

AIM: Write a python program to create two matrices(read values from user) and find the dot product.

Theoretical Support

numpy.dot(vector_a, vector_b, out = None) returns the dot product of vectors a and b. It can handle 2D arrays but considers them as matrix and will perform matrix multiplication.

Code

```
import numpy as np

print("Enter the elements of first matrix- 2 X 2 ");
elements = list(map(int, input().split()));
a= np.array(elements).reshape(2,2);
print("Enter the elements of second matrix- 2 X 2");
elements = list(map(int, input().split()));
b= np.array(elements).reshape(2,2);
print("\nFirst Matrix\n",a)
print("Second matrix\n",b)
print("\nDot product of first and second matrix");
np.dot(a, b)
```

Inference

We can find the dot product of the 2 matrix using numpy.dot() function.During matrix multiplication,we multiply the values of the rows of matrix a with the values of the columns of matrix b and sum them up.

Results and Observations

```
Matrix A
 [[0 1 2]
 [3 4 5]]
Transpose of Matrix A
 [[0 3]
 [1 4]
 [2 5]]
```

TRANSPOSE OF MATRIX

Program No: 9 **Date:**05-09-22

AIM: Write a python program to find the transpose of the matrix.

Theoretical Support

With the help of **Numpy numpy.transpose()**, We can perform the simple function of transpose within one line by using **numpy.transpose()** method of Numpy. It can transpose the 2-D arrays on the other hand it has no effect on 1-D arrays. This method transpose the 2-D numpy array.

Code

```
import numpy as np

a= np.arange(6).reshape((2,3))
print("Matrix A\n",a)
b=np.transpose(a)
print("Transpose of Matrix A\n",b)
```

Inference

Transpose() converts the row data to the column data and column data to the row data.

Results and Observations

```
Matrix A
 [[0 1 2]
 [3 4 5]
 [6 7 8]]

Trace of Matrix A
 12
```

TRACE OF MATRIX

Program No: 10 **Date:**09-09-22

AIM: Write a python program to find the trace of the matrix.

Theoretical Support

Trace of the matrix is calculated using **numpy.trace()**.

Code

```
import numpy as np
a= np.arange(9).reshape((3,3))
print("Matrix A\n",a)
b=np.trace(a)
print("\nTrace of Matrix A\n",b)
```

Inference

Numpy.trace() calculate the sum of its digonal elements from the upper left to lower right of matrix.

Results and Observations

```
Matrix A
 [[0 1 2]
 [3 4 5]
 [6 7 8]]
Rank of Matrix A: 2
```

RANK OF MATRIX

Program No: 11 **Date:**09-09-22

AIM: Write a python program to find the rank of the matrix.

Theoretical Support

Rank of the matrix is calculated using matrix_rank() of numpy.

Code

```
import numpy as np
a= np.arange(9).reshape((3,3))
print("Matrix A\n",a)
b=np.linalg.matrix_rank(a)
print("Rank of Matrix A:",b)
```

Inference

Matrix_rank() returns the number of linearly independent columns present in the matrix.

Results and Observations

```
determinant of the matrix:
1330.0000000000002
```

DETERMINANT OF MATRIX

Program No: 12 **Date:**09-09-22

AIM: Write a python program to find the determinant of the matrix.

Theoretical Support

Determinant of the matrix is calculated using det() of numpy.

Code

```
import numpy as np
a = np.array([[50,29], [30,44]])
print("\ndeterminant of the matrix:")
print(np.linalg.det(a))
```

Inference

det(a) will calculate the sum of products of the elements of of any row or column and their corresponding co-factor.

Results and Observations

```
Matrix x
 [[4 3]
 [3 2]]
Inverse of Matrix x
 [[-2.  3.]
 [ 3. -4.]]
```

INVERSE OF MATRIX

Program No: 13 **Date:**12-09-22

AIM: Write a python program to find the inverse of the matrix.

Theoretical Support

Inverse of the matrix can be calculated using inv().

Code

```
import numpy as np
x = np.array([[4,3],[3,2]])
y = np.linalg.inv(x)
print("Matrix x\n",x)
print("Inverse of Matrix x\n",y)
```

Inference

Inv() easily find the inverse of the matrix by finding determinant and adjoint.

Results and Observations

```
Matrix
 [[0 2]
 [2 3]]

Eigen value: [-1.  4.]

Eigen vector
 [[-0.89442719 -0.4472136 ]
 [ 0.4472136  -0.89442719]]
```

EIGEN VALUES AND EIGEN VECTORS

Program No:14 **Date:**12-09-22

AIM: Write a python program to find the eigen values and eigen vectors..

Theoretical Support

Eigen value and eigen vectors are calculated using eig() of numpy.

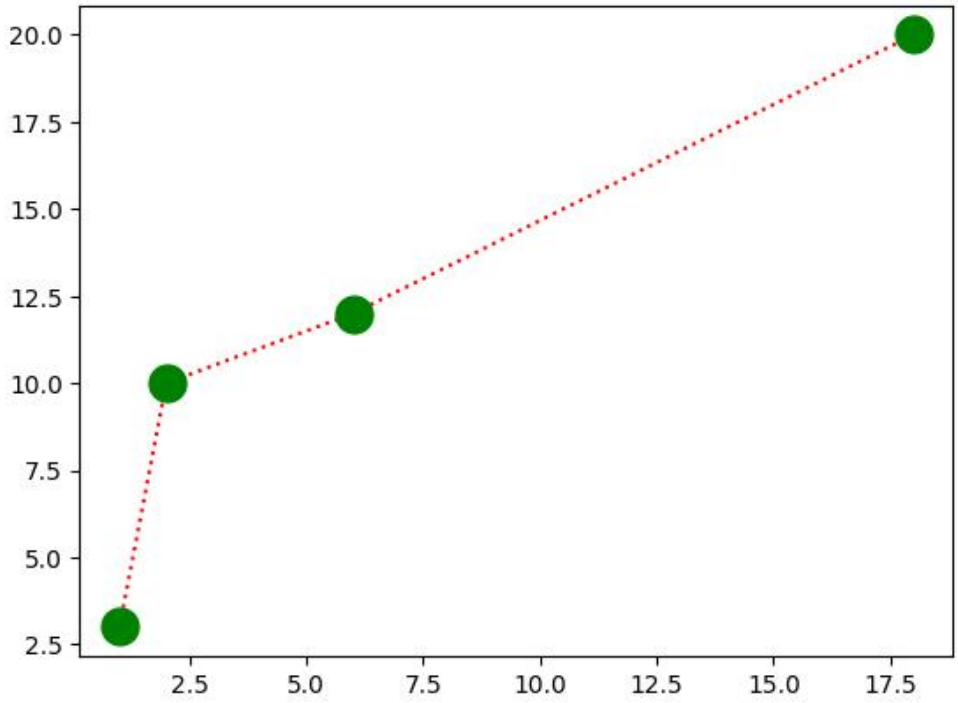
Code

```
  import numpy as np
from numpy.linalg import eig
a = np.array([[0, 2], [2, 3]])
print("Matrix\n",a)
val,vec=eig(a)
print('\nEigen value:', val)
print('\nEigen vector\n', vec)
```

Inference

eig(a) returns two values,first is eigen values and second is eigen vectors.

Results and Observations



EXERCISE 3:PROGRAM USING MATPLOTLIB

LINE DIAGRAM

Program No: 15 **Date:**12-09-22

AIM: Draw a line in a diagram from position (1, 3) to (2, 10) then to (6, 12) and finally to position (18, 20). (Mark each point with a beautiful green colour and set line colour to red and line style dotted)

Theoretical Support

Line diagram is plotted using plot() of malplotlib.pyplot

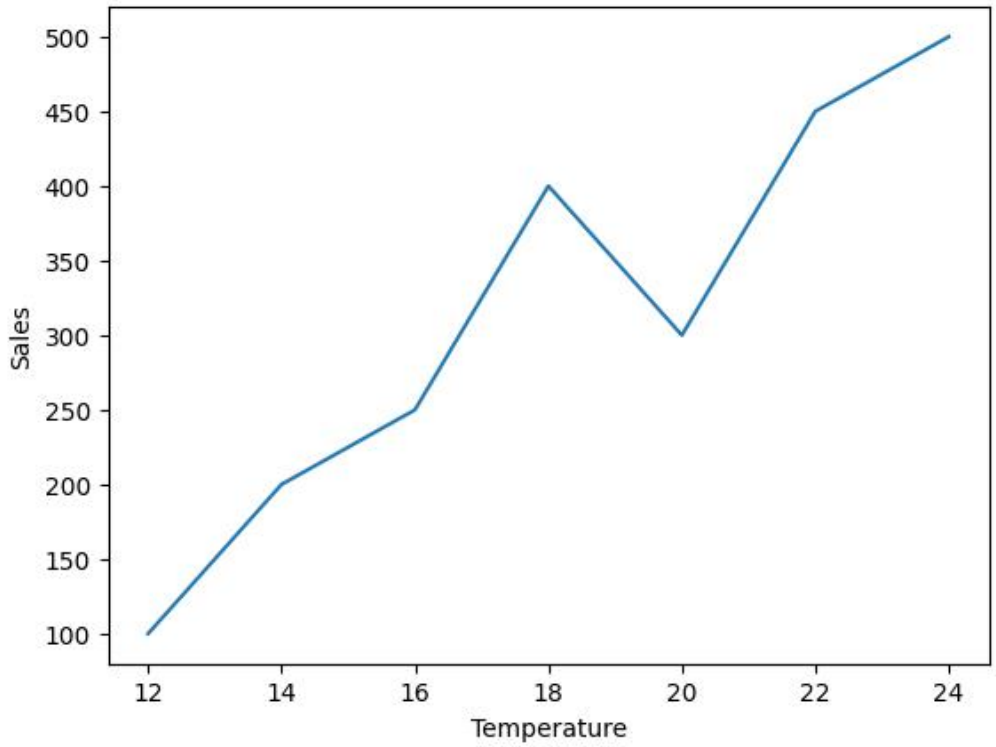
Code

```
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([1, 2, 6, 18])
ypoints = np.array([3, 10, 12, 20])
plt.plot(xpoints, ypoints,marker='o',color='r',linestyle=':',mfc='g',mec='g',ms='15')
plt.show()
```

Inference

Plt.plot() will plot the xpoints and ypoints in the xy plane with customized line color and size.

Results and Observations



PLOT FOR THE GIVEN DATA

Program No: 16 **Date:**16-09-22

AIM: Draw a plot for the following data:

Temperature in degree Celsius	Sales
12	100
14	200
16	250
18	400
20	300
22	450
24	500

Theoretical Support

Line diagram is plotted using plot() of malplotlib.pyplot

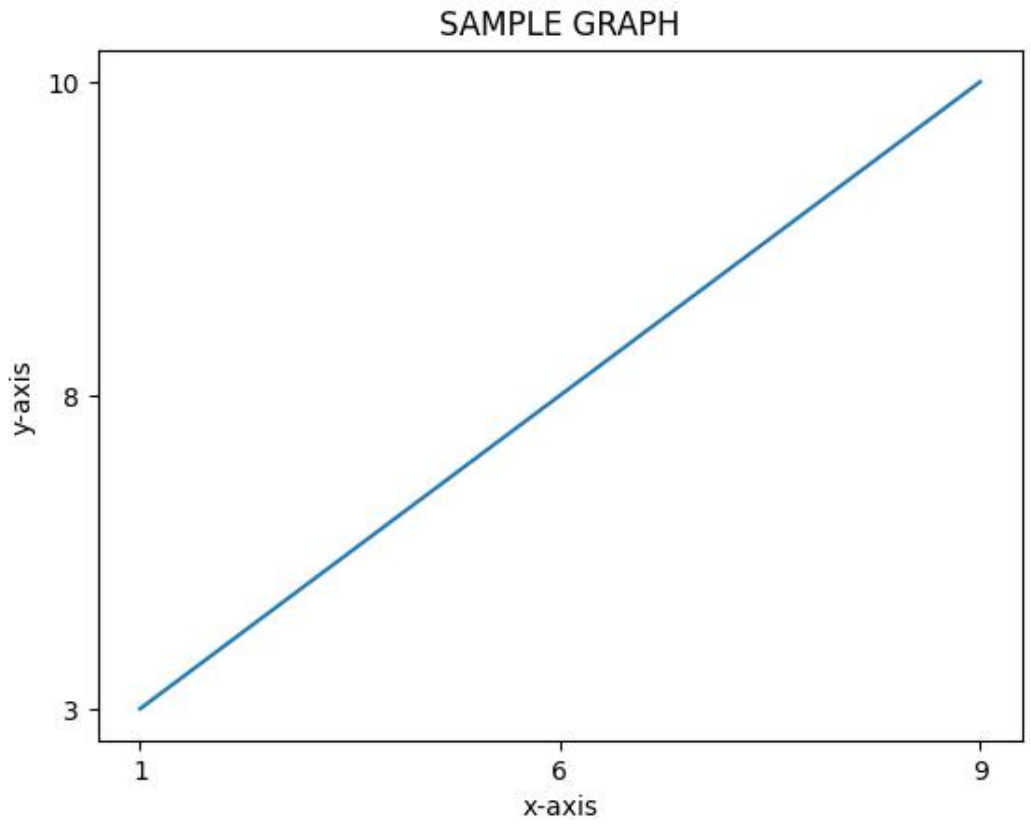
Code

```
import matplotlib.pyplot as plt
import numpy as np
xpoints = [12,14,16,18,20,22,24]
ypoints = [100,200,250,400,300,450,500]
plt.plot(xpoints,ypoints)
plt.xlabel("Temperature")
plt.ylabel("Sales")
plt.show()
```

Inference

Plt.plot(xpoints,ypoints) plots xpoints on x-axis and ypoints on y-axis.

Results and Observations



LINE DIAGRAM USING TEXT FILE

Program No: 17 **Date:**16-09-22

AIM: Write a Python program to draw a line using given axis values taken from a text file, with suitable label in the x axis, y axis and a title

Theoretical Support

Data from file is readed using read().Data is splited using split().

Code

```
import matplotlib.pyplot as plt
with open("plot.txt") as f:
    data = f.read()
data = data.split('\n')
x = [row.split(' ')[0] for row in data]
y = [row.split(' ')[1] for row in data]
plt.plot(x,y)
plt.title("SAMPLE GRAPH")
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.show()
```

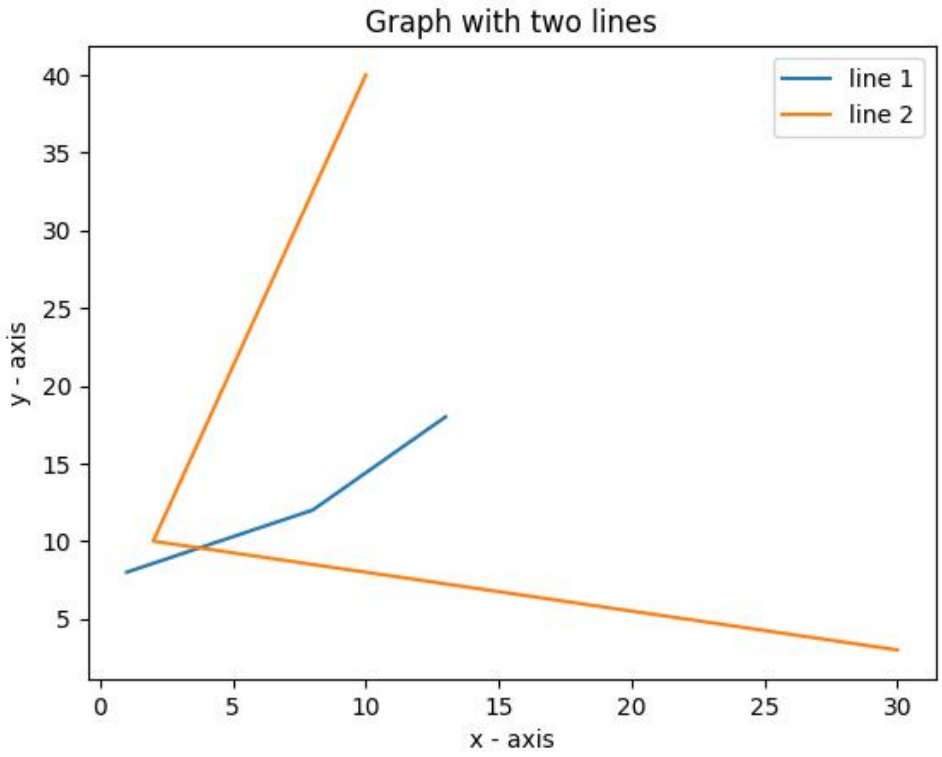
Plot.txt

```
1 3
6 8
9 10
```

Inference

Data points from plot.txt is readed and splited into 2 columns using split() and ploted the x values on x-axis and y values on y-axis.

Results and Observations



MULTIPLE LINES ON SAME PLOT

Program No: 18 **Date:**16-09-22

AIM: Write a Python program to plot two or more lines on same plot with suitable legends of each line

Theoretical Support

Plt.plot() used for creating plots.

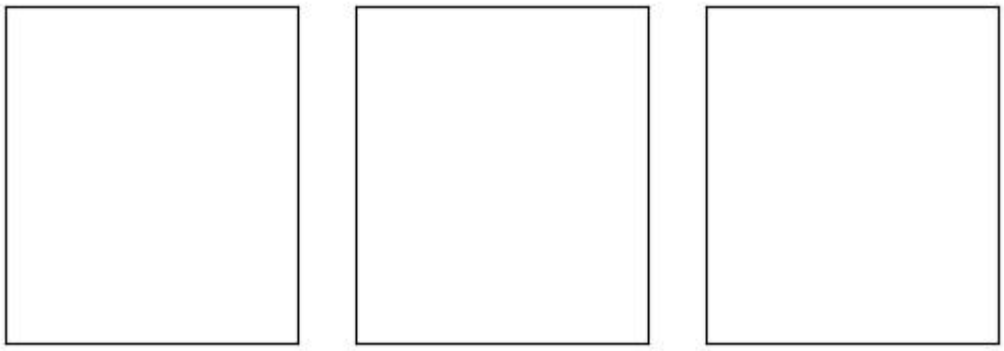
Code

```
import matplotlib.pyplot as plt
x1 = [1,8,13]
y1 = [8,12,18]
plt.plot(x1, y1, label = "line 1")
x2 = [10,2,30]
y2 = [40,10,3]
plt.plot(x2, y2, label = "line 2")
plt.title('Graph with two lines')
plt.xlabel('x - axis')
plt.ylabel('y - axis')
plt.legend()
plt.show()
```

Inference

Multiple lines on the same plot can be ploted using plt.plot().first line with x1,y1 points and second line with x2,y2 are plotted.

Results and Observations



MULTIPLE PLOTS

Program No: 19 **Date:**19-09-22

AIM: Write a Python program to create multiple plots.

Theoretical Support

Multiple plots are created using plt.subplot().

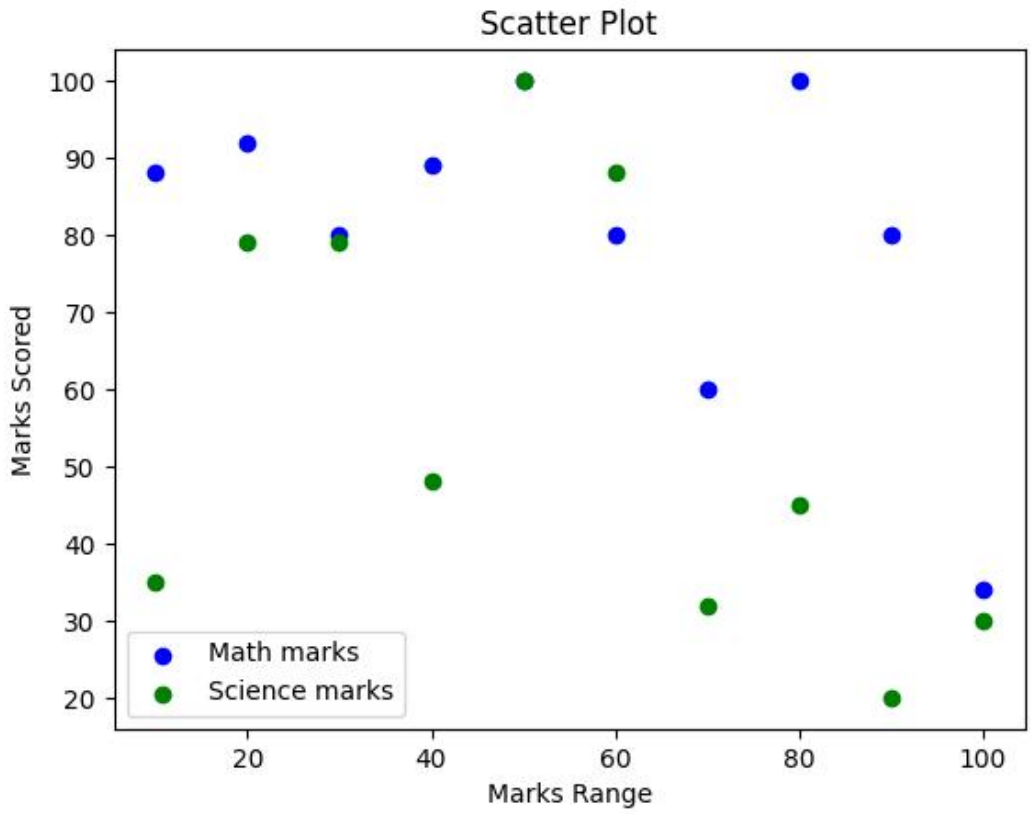
Code

```
import matplotlib.pyplot as plt
plt.subplot(2, 3, 4)
plt.xticks(())
plt.yticks(())
plt.subplot(2, 3, 5)
plt.xticks(())
plt.yticks(())
plt.subplot(2, 3, 6)
plt.xticks(())
plt.yticks(())
plt.show()
```

Inference

Each plt.subplot() will create subplots in the main plot.

Results and Observations



SCATTER PLOT

Program No: 20 **Date:**19-09-22

AIM: Write a Python program to draw a scatter plot comparing two subject marks of Mathematics and Science. Use marks of 10 students.

Sample data:

math_marks = [88, 92, 80, 89, 100, 80, 60, 100, 80, 34]
science_marks = [35, 79, 79, 48, 100, 88, 32, 45, 20, 30]
marks_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

Theoretical Support

The scatter diagram graphs numerical data pairs, with one variable on each axis, show their relationship

Code

```
import matplotlib.pyplot as plt
math_marks = [88, 92, 80, 89, 100, 80, 60, 100, 80, 34]
science_marks = [35, 79, 79, 48, 100, 88, 32, 45, 20, 30]
marks_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
plt.scatter(marks_range, math_marks, label='Math marks', color='b')
plt.scatter(marks_range, science_marks, label='Science marks', color='g')
plt.title('Scatter Plot')
plt.xlabel('Marks Range')
plt.ylabel('Marks Scored')
plt.legend()
plt.show()
```

Inference

Plt.scatter will plot the maths_marks and science mark with marks_range with custom color.

CYCLE 2

Results and Observations

```
Original list:
[11, 33, 55, 77, 99]

After conversion
0    11
1    33
2    55
3    77
4    99
dtype: int64
```

EXERCISE 4:INTRODUCTION TO PANDAS

LIST TO SERIES CONVERSION

Program No: 21 **Date:**23-09-22

AIM: Write a python program to implement List-to-Series Conversion.

Theoretical Support

List is converted to series using pandas.series().

Code

```
import pandas as pd


list = [11, 33, 55, 77, 99]
print("Original list:")
print(list)
print("\nAfter conversion")
print(pd.Series(list))
```

Inference

Pd.series(list) will convert the list data to series.

Results and Observations

```
DatetimeIndex(['2021-05-01', '2021-05-02', '2021-05-03', '2021-05-04',
               '2021-05-05', '2021-05-06', '2021-05-07', '2021-05-08',
               '2021-05-09', '2021-05-10', '2021-05-11', '2021-05-12'],
              dtype='datetime64[ns]', freq='D')
```

GENERATING SERIES OF DATES

Program No: 22 **Date:**23-09-22

AIM: Write a python program to Generate the series of dates from 1st May, 2021 to 12th May, 2021 (both inclusive).

Theoretical Support

Pandas.date_range() will create series of date between two ranges.

Code

```
import pandas as pd
result = pd.date_range(start = '05-01-2021', end = '05-12-2021')
print(result)
```

Inference

pd.date_range(start = '05-01-2021', end = '05-12-2021') will create series of date between '05-01-2021' to '05-12-2021'.

Results and Observations

	slno	name
0	1	albin
1	2	ajay
2	3	binu
3	4	sam

DICTIONARY TO DATA FRAME

Program No: 23 **Date:**23-09-22

AIM: Given a dictionary, convert it into corresponding dataframe and display it.

Theoretical Support

Pandas.dataframe() will convert the data into dataframe.

Code

```
import pandas as pd
data={'slno':[1,2,3,4],'name':['albin','ajay','binu','sam']}
df=pd.DataFrame(data)
print(df)
```

Inference

Pd.dataframe(data) will convert the dictionary to dataframe.

Results and Observations

	0	1	2
0	java	2000	200
1	cpp	3000	300
2	c	4000	400

LIST TO DATA FRAME

Program No: 24 **Date:**30-09-22

AIM: Given a 2D List, convert it into corresponding dataframe and display it.

Theoretical Support

Pandas.dataframe() will convert the list to dataframe.

Code

```
import pandas as pd
data = [['java', '2000', '200'],
        ['cpp','3000','300'],
        ['c', '4000','400']]
data = pd.DataFrame(data)
print(data)
```

Inference

Pd.DataFrame(data) will convert the list data to dataframe.

Results and Observations

	0	1	2
0	html	css	javascript
1	c	cpp	java
2	100	200	300

CSV FILE TO DATA FRAME

Program No: 25 **Date:**30-09-22

AIM: Given a CSV file, read it into a dataframe and display it.

Theoretical Support

Csv file is readed using read_csv() of pandas.

Code

```
import pandas as pd
data = pd.read_csv ('sample.csv',header=None)
df=pd.DataFrame(data)
print(df)
```

Sample.csv

```
html  html  html
c     cpp   java
100   200   300
```

Inference

Pd.DataFrame(data) splits the data into data frames.

Results and Observations

	0	1	2
0	java	2000	200
1	cpp	3000	300

DISPLAY FIRST 2 ROWS OF DATA FRAME

Program No: 26 **Date:**10-10-22

AIM: Given a dataframe, select first 2 rows and output them.

Theoretical Support

Dataframe() of pandas used for creating data frames.

Code

```
import pandas as pd
data = [['java', '2000', '200'],
        ['cpp','3000','300'],
        ['c', '4000','400']]
df = pd.DataFrame(data)
print(df.head(2))
```

Inference

df.head(2) will display first 2 rows of the dataframe.

Results and Observations

```
data
       name occupation  salary
0    sneha   engineer   70000
1  pradeep   mechanic   40000
2    Alvin   engineer   65000
3    Rahul   mechanic   38000
4   Ramesh   mechanic   35000

Average salary per occupation

 occupation
engineer    67500.000000
mechanic    37666.666667
Name: salary, dtype: float64
```

AVERAGE SALARY PER OCCUPATION

Program No: 27 **Date:**10-10-22

AIM: Given is a dataframe showing name, occupation, salary of people. Find the average salary per occupation.

Theoretical Support

read_csv() used for reading csv file.

Code

```
import pandas as pd
data=pd.read_csv('people.csv')
print("data\n",data)
print("\nAverage salary per occupation\n\n",data.groupby('occupation')['salary'].mean())
```

People.csv

name	occupation	salary
sneha	engineer	70000
pradeep	mechanic	40000
Alvin	engineer	65000
Rahul	mechanic	38000
Ramesh	mechanic	35000

Inference

Groupby(occupation) will group the data according to the occupation.

mean() will calculate the average value.

Results and Observations

```
Data frame 1
    eid     ename  stipend
0  101  ajay dev    10000
1  102      binu    15000
2  104    rajeev     8000
3  107   sandhya     7000

Data frame 2
    eid designation
0  101  programmer
1  102     analyst
2  104   developer
3  107  programmer

Combined
    eid     ename  stipend designation
0  101  ajay dev    10000  programmer
1  102      binu    15000     analyst
2  104    rajeev     8000   developer
3  107   sandhya     7000  programmer
```

DISPLAYING EMPLOYEE DETAILS FROM DATA FRAME

Program No: 28 **Date:**17-10-22

AIM: Given are 2 dataframes, with one dataframe containing Employee ID (eid), Employee Name (ename) and Stipend (stipend) and the other dataframe containing Employee ID (eid) and designation of the employee (designation). Output the Dataframe containing Employee ID (eid), Employee Name (ename), Stipend (stipend) and Position (position).

Theoretical Support

Pandas.merge() used for merging 2 data frames.

Code

```
import pandas as pd
df1=pd.read_csv('employee1.csv')
df2=pd.read_csv('employee2.csv')
print("Data frame 1\n",df1)
print("\nData frame 2\n",df2)
print("\nCombined\n",pd.merge(df1,df2, how = 'inner', on = 'eid'))
```

employee1.csv

eid	ename	stipend
101	ajay dev	10000
102	binu	15000
104	rajeev	8000
107	sandhya	7000

employee2.csv

eid	designation
101	programmer
102	analyst
104	developer
107	programmer

Inference

Pd.merge(df1,df2) will merge the data frames df1 and df2.

CYCLE 3

Results and Observations

```
Accuracy
0.9666666666666667
```

EXERCISE 5

K-NN CLASSIFICATION

Program No: 29 **Date:**21-10-22

AIM: Program to implement **k-NN classification** using any standard dataset available in the public domain and find the accuracy of the algorithm

Theoretical Support

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

Dataset used:iris

Code

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
irisData=load_iris()
x=irisData.data
y=irisData.target
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state=42)
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(X_train, y_train)
print("Accuracy")
print(knn.score(X_test, y_test))
```

Inference

Dataset is splited into training(80%) and testing(20%) sets. knn.fit() will create the model using training data.

Knn.score() will find the accuracy of the model and a accuracy of 0.966667

Results and Observations

```
[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0
 1 1 1 2 0 2 0 0]
CM [[16  0  0]
 [ 0 18  0]
 [ 0  0 11]]


predicted output for [[5,5,4,4]]: [2]
Naive bayes score : 1.0
```

EXERCISE 6

NAIVE BAYES ALGORITHM

Program No: 30 **Date:**28-10-22

AIM: Program to implement **Naïve Bayes Algorithm** using any standard dataset available in the public domain and find the accuracy of the algorithm.

Theoretical Support

Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.It is mainly used in text classification that includes a high-dimensional training dataset.

Dataset used:iris

Code

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
X,y=load_iris(return_X_y=True)
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.30,random_state=0)
gnb=GaussianNB()
y_pred=gnb.fit(X_train,y_train).predict(X_test)
print(y_pred)
x_new=[[5,5,4,4]]
y_new=gnb.fit(X_train,y_train).predict(x_new)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print ("CM", cm)
print("\n")
print("predicted output for [[5,5,4,4]]:",y_new)
print("Naive bayes score :",gnb.score(X_test,y_test))
```

Inference

Data set is splitted into training(70%) and testing sets(30%).gnb.fit() will create the model using training data.Accuracy is obtained using gnb.score() and a accuracy of 1.0 is obtained.

Results and Observations

```
Accuracy:  0.9555555555555556
```

EXERCISE 7

DECISION TREE

Program No: 31 **Date:**8-11-22

AIM: Program to implement **decision trees** using any standard dataset available in the public domain and find the accuracy of the algorithm

Theoretical Support

Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**

Dataset used:iris

Code

```
from sklearn.datasets import load_iris
from sklearn import metrics
from sklearn import tree
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
iris=load_iris()
x=iris.data
y=iris.target
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=1)
clf=DecisionTreeClassifier()
clf=clf.fit(x_train,y_train)
y_pred=clf.predict(x_test)
print("Accuracy: ",metrics.accuracy_score(y_test,y_pred))
```

Inference

Data set is splitted into training(70%) and testing sets(30%).clf.fit() will create the model using training data.Accuracy is obtained using metrics.accuracy_score() and a accuracy of 0.9555 is obtained

Results and Observations

```
Coefficients:
 [938.23786125]
Mean squared error: 2548.07
Coefficient of determination: 0.47
```

EXERCISE 8

REGRESSION

Program No: 32 **Date:**18-11-22

AIM: Program to implement linear and multiple **regression** techniques using any standard dataset available in the public domain and evaluate its performance

Theoretical Support

Regression is **a technique for investigating the relationship between independent variables or features and a dependent variable or outcome**. It's used as a method for predictive modelling in machine learning, in which an algorithm is used to predict continuous outcomes

Dataset used:diabetes

Code

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
df = datasets.load_diabetes()
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
diabetes_X = diabetes_X[:, np.newaxis, 2]
diabetes_X_train = diabetes_X[:-20]
diabetes_X_test = diabetes_X[-20:]
diabetes_y_train = diabetes_y[:-20]
diabetes_y_test = diabetes_y[-20:]
regr = linear_model.LinearRegression()
regr.fit(diabetes_X_train, diabetes_y_train)
diabetes_y_pred = regr.predict(diabetes_X_test)
print("Coefficients: \n", regr.coef_)
print("Mean squared error: %.2f" % mean_squared_error(diabetes_y_test, diabetes_y_pred))
print("Coefficient of determination: %.2f" % r2_score(diabetes_y_test, diabetes_y_pred))
```

Inference

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data.Multiple regression is a statistical technique that can be used to analyze the relationship between a single dependent variable and several independent variables.

Results and Observations

```
Accuracy :  1.0
['setosa']
```

EXERCISE 9

SUPPORT VECTOR MACHINE

Program No: 33 **Date:**24-11-22

AIM: Program to implement text classification using a **Support vector machine**.

Theoretical Support

support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

Dataset used:iris

Code

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.svm import SVC
iris = load_iris()
x = iris.data
y = iris.target
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=1)
classifier = SVC(kernel='linear', random_state=0)
classifier.fit(x_train, y_train)
y_pred= classifier.predict(x_test)
print("Accuracy : ",metrics.accuracy_score(y_test,y_pred))
sample = [[1,1,1,2]]
pred = classifier.predict(sample)
pred_v = [iris.target_names[p] for p in pred]
print(pred_v)
```

Inference

Data set is splitted into training(70%) and testing(30%) sets.

metrics.accuracy_score() is used for obtaining accuracy and a accuracy of 1.0 is obtained.

Results and Observations

```
[[5.77358491 2.69245283]
 [5.006      3.428     ]
 [6.81276596 3.07446809]]
[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 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 0 2 0 2 0 2 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0
 2 2 2 2 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2
 2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 0 2 2 2 0 2 2 2 0 2 2 2 0 2
 2 0]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

EXERCISE 10

K-MEANS CLUSTERING

Program No: 34 **Date:**02-12-22

AIM: Program to implement **k-means clustering** technique using any standard dataset available in the public domain

Theoretical Support

Clustering is a type of unsupervised learning where the references need to be drawn from unlabelled datasets. Generally, it is used to capture meaningful structure, underlying processes, and grouping inherent in a dataset. In clustering, the task is to divide the population into several groups in such a way that the data points in the same groups are more similar to each other than the data points in other groups. In short, it is a collection of objects based on their similarities and dissimilarities.

Dataset used:iris

Code

```
from sklearn import datasets
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.cluster import KMeans
iris = datasets.load_iris()
X = iris.data[:, :2]
y = iris.target
km = KMeans(n_clusters = 3, init='k-means++', n_init=10, max_iter=300, tol=0.0001, verbose=0, random_state=21, copy_x=True, algorithm="auto")
km.fit(X)
centers = km.cluster_centers_
print(centers)
new_labels = km.labels_
print(new_labels)
print(y)
```

Inference

K-means algorithm identifies *k* number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.The *'means'* in the K-means refers to averaging of the data; that is, finding the centroid.