S NO.	QUESTION	TEACHER SIGNATURE
1.	Write a program to implement data preprocessing on student's dataset. a. Handle missing data using different methods. b. Delete the row containing null values. c. Delete the row containing all null values. d. Replace the null values using forward method. e. Replace the null values using backward method. f. Replace with constant value. g. Replace with central tendency measures – mean, mode, medium.	
2.	Implement data preprocessing on the given e commerce dataset: a. Describe the data a. Head b. Tail c. Shape d. Info b. Check whether null values are present or not and display column wise also. c. Extract the independent and dependent variables. d. Handle the missing data. e. split into training and test data.	
3.	Write a program to implement line plots by comparing performances of computer science subject of different years. a. Draw a basic line plot using plot() b. Display a line plot using show() c. Display markers (initial and final endpoints) d. Take only y data. e. Display markers on every point. f. Set the marker properties : set the marker color, size, edge color, face color. g. Set the line properties: line style, line color, line width. h. Set the x axis label, y axis label and title of the graph. i. Set the font properties for title and labels : font family, color and size. j. Change the position of title. k. Add gridline to the plot(only x axis, only y axis, both). l. Set grid properties : grid color, grid linestyle, grid line width. m. Display multiple plots using subplot() . n. Display multiple lineplots in a single plot for comparison.	
4.	Write a program to plot a line chart of student's performance in 5 tests of minor1 and minor2 by setting the line style and marker style.	
5.	Write a program to implement a bar graph showing the no. of medals in commonwealth game. a. Create a basic bar graph using bar(). b. Draw a bar graph horizontally using barh(). c. change color of bar. d. change width of bar. e. change the alignment of the labels.	

6.	Draw a scatter plot that shows the stock trend for 5 years for TATA and Reliance company. Use the following properties:- a. Draw the basic scatter graph using scatter() b. Compare two plots in a single plot. c. Set the color of both plots. d. Set the different color of every dot. e. Set colormap and display it using colorbar(). f. Set size of the dot. g. Set the different size of every dot. h. Set the transparency of dot.	
7.	Draw a pie chart that shows the sales of a different fruit in a day for a shop: 30% banana, 20% other, 15% orange, 10% guava, 25% mango Use the following properties:- a. Draw basic pie chart using pie(). b. Add labels to wedges. c. Change the start angle of any wedge. d. Displace the centre of wedge. e. Add shadow to slice wedges. f. Change the colour of wedges. g. Add legend. h. Add legend with header.	
8.	Create a csv file displaying countries and no. Of companies in each country. Display a bar graph from this csv file and style the bar graph too.	
9.	Write a program to combine two dataframes (data contained in two csv files – f1.csv country, no. Of companies and f2.csv – country, no. Of employees) Using merge() , join(), concat().	
10.	Implement these graphs using seaborn library:- Lineplot, Distplot, Lmplot, Countplot, Relplot, Displot, Catplot	

Q QUESTION 1) Write a program to implement data preprocessing on student's dataset.

SOLUTION :-

```
[2]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
```

```
[4]  data=pd.read_csv('missing.csv')
     data
```

	Roll no	Name	Marks
0	11.0	abhi	34.0
1	12.0	amrit	54.0
2	13.0	NaN	54.0
3	14.0	ansh	NaN
4	15.0	nim	67.0
5	NaN	NaN	NaN
6	17.0	shu	54.0
7	18.0	NaN	65.0
8	19.0	charu	NaN
9	20.0	sham	67.0

a) Delete the row containing null values.

```
df_clear=data.dropna()
df_clear
```

	Roll no	Name	Marks
0	11.0	abhi	34.0
1	12.0	amrit	54.0
4	15.0	nim	67.0
6	17.0	shu	54.0
9	20.0	sham	67.0

b) Delete the row containing all null values.

```
df_clear=data.dropna(how='all')
df_clear
```

	Roll no	Name	Marks
0	11.0	abhi	34.0
1	12.0	amrit	54.0
2	13.0	NaN	54.0
3	14.0	ansh	NaN
4	15.0	nim	67.0
6	17.0	shu	54.0
7	18.0	NaN	65.0
8	19.0	charu	NaN
9	20.0	sham	67.0

c) Replace the null values using forward method.

```
[13] data.fillna(method='ffill')
```

	Roll no	Name	Marks
0	11.0	abhi	34.0
1	12.0	amrit	54.0
2	13.0	amrit	54.0
3	14.0	ansh	54.0
4	15.0	nim	67.0
5	15.0	nim	67.0
6	17.0	shu	54.0
7	18.0	shu	65.0
8	19.0	charu	65.0
9	20.0	sham	67.0

d) Replace the null values using backward method.

```
[13] data.fillna(method='ffill')
```

	Roll no	Name	Marks
0	11.0	abhi	34.0
1	12.0	amrit	54.0
2	13.0	amrit	54.0
3	14.0	ansh	54.0
4	15.0	nim	67.0
5	15.0	nim	67.0
6	17.0	shu	54.0
7	18.0	shu	65.0
8	19.0	charu	65.0
9	20.0	sham	67.0

e) Replace with constant value.

```
data.fillna(1)
```

	Roll no	Name	Marks
0	11.0	abhi	34.0
1	12.0	amrit	54.0
2	13.0	1	54.0
3	14.0	ansh	1.0
4	15.0	nim	67.0
5	1.0	1	1.0
6	17.0	shu	54.0
7	18.0	1	65.0
8	19.0	charu	1.0
9	20.0	sham	67.0

f) Replace with central tendency measures – mean, mode, medium.

```
mean_value = data['Marks'].mean()
data['Marks'] = data['Marks'].fillna(mean_value)
data['Marks']
```

```
0    34.000000
1    54.000000
2    54.000000
3    56.428571
4    67.000000
5    56.428571
6    54.000000
7    65.000000
8    56.428571
9    67.000000
Name: Marks, dtype: float64
```

```
median_value = data['Marks'].median()
data['Marks'] = data['Marks'].fillna(median_value)
data['Marks']
```

```
0    34.000000
1    54.000000
2    54.000000
3    56.428571
4    67.000000
5    56.428571
6    54.000000
7    65.000000
8    56.428571
9    67.000000
Name: Marks, dtype: float64
```

```
data['Marks'].fillna(data['Marks'].mode()[0])
```

```
0    34.000000
1    54.000000
2    54.000000
3    56.428571
4    67.000000
5    56.428571
6    54.000000
7    65.000000
8    56.428571
9    67.000000
Name: Marks, dtype: float64
```

QUESTION2) Implement data preprocessing on the given e commerce dataset:

SOLUTION :-

+ Code + Text

```
1  import pandas as pd
2  import numpy as np
3
4  # Read the CSV file into a DataFrame
5  dataset = pd.read_csv('data.csv')
6
7  # Display the dataset
8  dataset
9
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

a. Describe the data.

a. Head

```
1 dataset.head()
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes

b. Tail

```
1 dataset.tail()
```

	Country	Age	Salary	Purchased
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

c. Shape

```
1 dataset.shape
```

```
(10, 4)
```

d. Info

```
[4]  1  dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Country    10 non-null     object
 1   Age        9 non-null      float64
 2   Salary     9 non-null      float64
 3   Purchased  10 non-null     object
dtypes: float64(2), object(2)
memory usage: 448.0+ bytes
```

b. Check whether null values are present or not and display column wise also.

```
1  dataset.isnull()
```

	Country	Age	Salary	Purchased
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	True	False
5	False	False	False	False
6	False	True	False	False
7	False	False	False	False
8	False	False	False	False
9	False	False	False	False

c. Extract the independent and dependent variables.

```
[ ]  1  x =dataset.iloc[ :,:-1].values
```

```
[ ]  1  x
```

```
array([['France', 44.0, 72000.0],
       ['Spain', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain', 38.0, 61000.0],
       ['Germany', 40.0, nan],
       ['France', 35.0, 58000.0],
       ['Spain', nan, 52000.0],
       ['France', 48.0, 79000.0],
       ['Germany', 50.0, 83000.0],
       ['France', 37.0, 67000.0]], dtype=object)
```

```
[ ]  1  y =dataset.iloc[ :,-1: ].values
```

```
[ ]  1  y
```

```
array([['No'],
       ['Yes'],
       ['No'],
       ['No'],
       ['Yes'],
       ['Yes'],
       ['No'],
       ['Yes'],
       ['No'],
       ['Yes']], dtype=object)
```

d. Handle the missing data.

```
1  dataset.dropna(how='any')
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
5	France	35.0	58000.0	Yes
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
1  dataset.dropna(how='all')
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
[ ]  1  dataset.fillna(method='bfill')
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	58000.0	Yes
5	France	35.0	58000.0	Yes
6	Spain	48.0	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
[ ]  1  dataset.fillna(method="ffill")
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	61000.0	Yes
5	France	35.0	58000.0	Yes
6	Spain	35.0	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
[ ]  1  dataset.fillna(0)
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	0.0	Yes
5	France	35.0	58000.0	Yes
6	Spain	0.0	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
1  dataset.fillna(dataset.mean(numeric_only=True))
```

	Country	Age	Salary	Purchased
0	France	44.000000	72000.000000	No
1	Spain	27.000000	48000.000000	Yes
2	Germany	30.000000	54000.000000	No
3	Spain	38.000000	61000.000000	No
4	Germany	40.000000	63777.777778	Yes
5	France	35.000000	58000.000000	Yes
6	Spain	38.777778	52000.000000	No
7	France	48.000000	79000.000000	Yes
8	Germany	50.000000	83000.000000	No
9	France	37.000000	67000.000000	Yes

```
1  dataset.fillna(dataset.mode())
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	61000.0	Yes
5	France	35.0	58000.0	Yes
6	Spain	44.0	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
[ ]  1  dataset.fillna
     2  (dataset.median(numeric_only=True))
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	61000.0	Yes
5	France	35.0	58000.0	Yes
6	Spain	38.0	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

e. Split into training and test data.

```
[ ]  1  x_train, x_test, y_train, y_test = train_test_split (x,y, test_size=0.2, random_state=0)
```

```
1  x_train
```

```
array([['Germany', 40.0, nan],
       ['France', 37.0, 67000.0],
       ['Spain', 27.0, 48000.0],
       ['Spain', nan, 52000.0],
       ['France', 48.0, 79000.0],
       ['Spain', 38.0, 61000.0],
       ['France', 44.0, 72000.0],
       ['France', 35.0, 58000.0]], dtype=object)
```

```
[ ]  1  x_test
```

```
array([['Germany', 30.0, 54000.0],
       ['Germany', 50.0, 83000.0]], dtype=object)
```

```
[ ]  1  y_train
```

```
array([['Yes'],
       ['Yes'],
       ['Yes'],
       ['No'],
       ['Yes'],
       ['No'],
       ['No'],
       ['Yes']], dtype=object)
```

```
[ ]  1  y_test
```

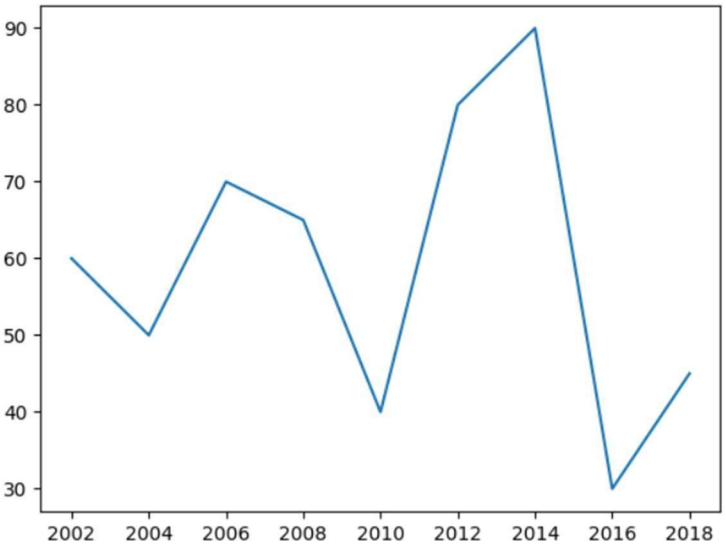
```
array([['No'],
       ['No']], dtype=object)
```

QUESTION3) Write a program to implement line plots by comparing performances of computer science subject of different years.

a. Draw a basic line plot using plot()

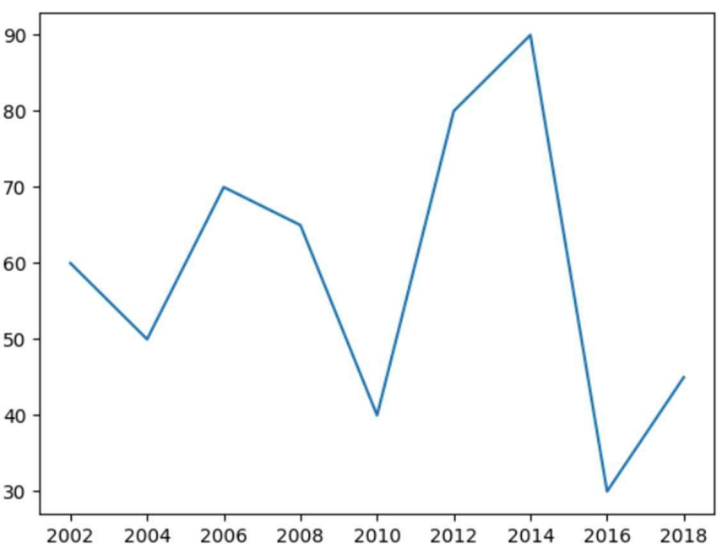
```
x=np.array([2002,2004,2006,2008,2010,2012,2014,2016,2018])
y=np.array([60,50,70,65,40,80,90,30,45])
plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x7a74eeeb4850>]
```



b. Display a line plot using show()

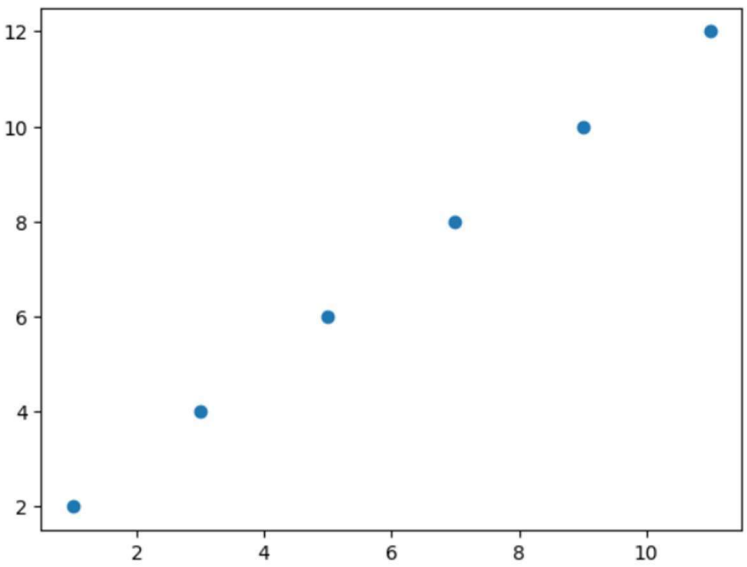
```
x = np.array([2002, 2004, 2006, 2008, 2010, 2012, 2014, 2016, 2018])
y = np.array([60, 50, 70, 65, 40, 80, 90, 30, 45])

plt.plot(x, y)
plt.show()
```



c. Display markers (initial and final endpoints)

```
x = np.array([1,3,5,7,9,11])
y = np.array([2,4,6,8,10,12])
plt.plot(x,y,'o')
```

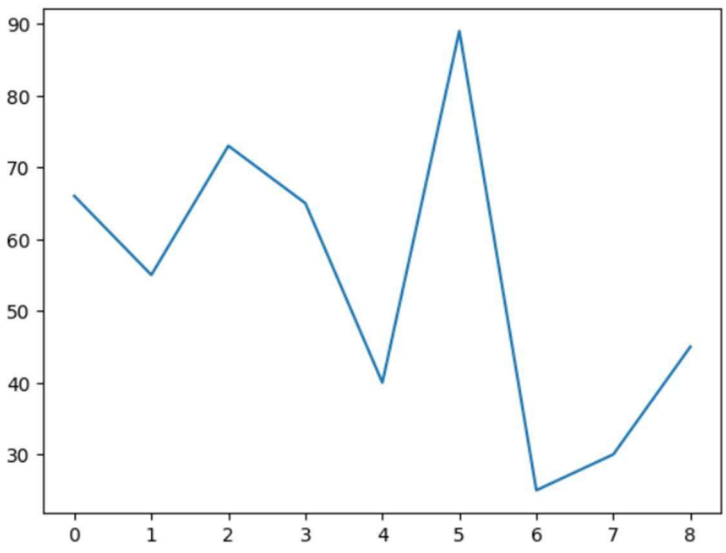
```
[<matplotlib.lines.Line2D at 0x7f53c890a800>]
```



d. Take only y data.

```
x=np.array([2002,2004,2006,2008,2010,2012,2014,2016,2018])
y=np.array([66,55,73,65,40,89,25,30,45])
plt.plot(y)
```

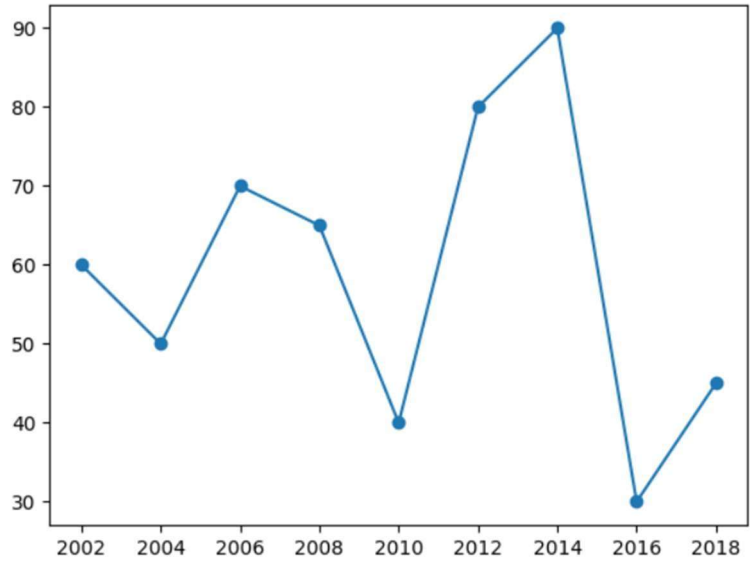
```
[<matplotlib.lines.Line2D at 0x7f53c7c16110>]
```



e. Display markers on every point.

```
x=np.array([2002,2004,2006,2008,2010,2012,2014,2016,2018])
y=np.array([60,50,70,65,40,80,90,30,45])
plt.plot(x,y,marker='o')
```

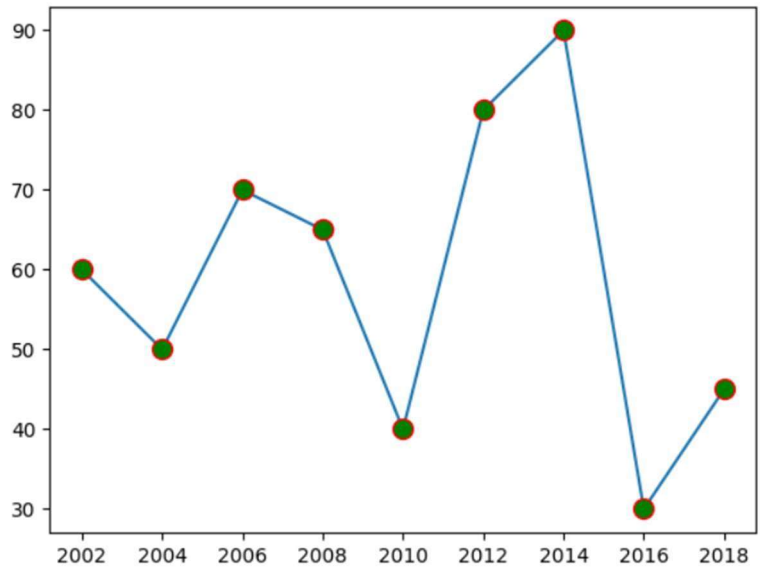
```
[<matplotlib.lines.Line2D at 0x7f53c7a9a620>]
```



f. Set the marker properties : set the marker color, size, edge color, face color.

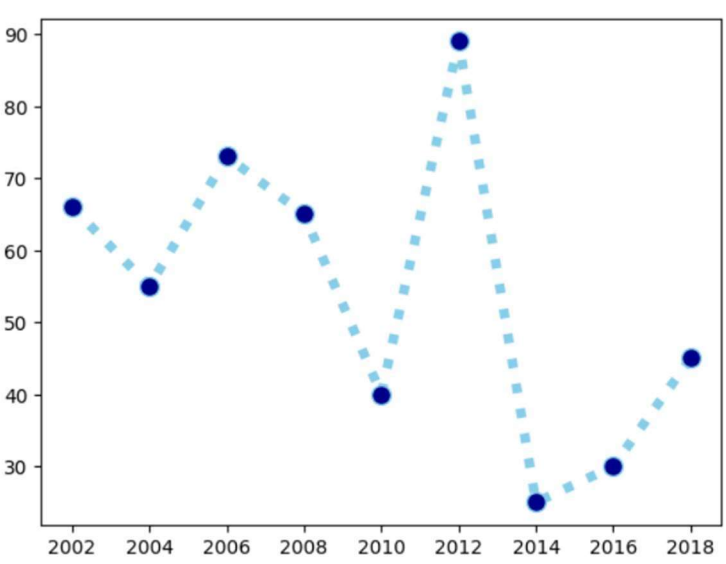
```
x=np.array([2002,2004,2006,2008,2010,2012,2014,2016,2018])
y=np.array([60,50,70,65,40,80,90,30,45])
plt.plot(x,y,marker='o',ms='10',mec='r',mfc='g')
```

```
[<matplotlib.lines.Line2D at 0x7f53c795f430>]
```



g. Set the line properties: line style, line color, line width.

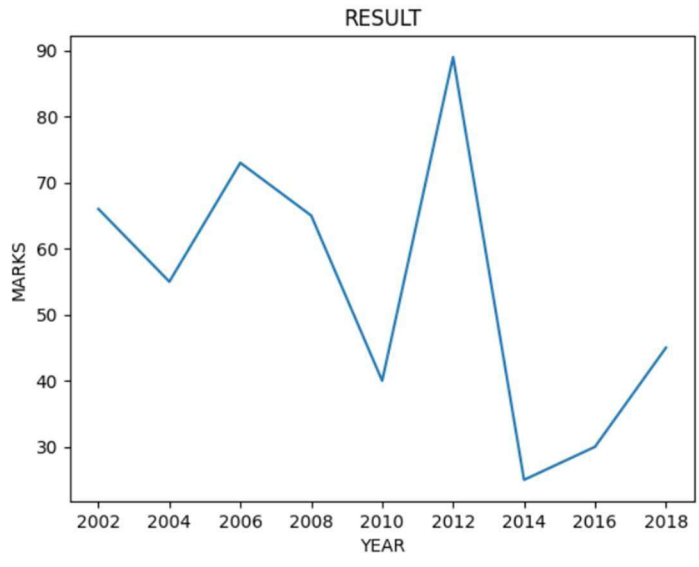
```
x=np.array([2002,2004,2006,2008,2010,2012,2014,2016,2018])
y=np.array([66,55,73,65,40,89,25,30,45])
plt.plot(x, y, ls='dotted', c='skyblue', lw='5', marker='o', ms='10',mfc='darkblue')
plt.show()
```



h. Set the x axis label, y axis label and title of the graph.

```
plt.plot(x,y)
plt.title("RESULT")
plt.xlabel("YEAR")
plt.ylabel("MARKS")
```

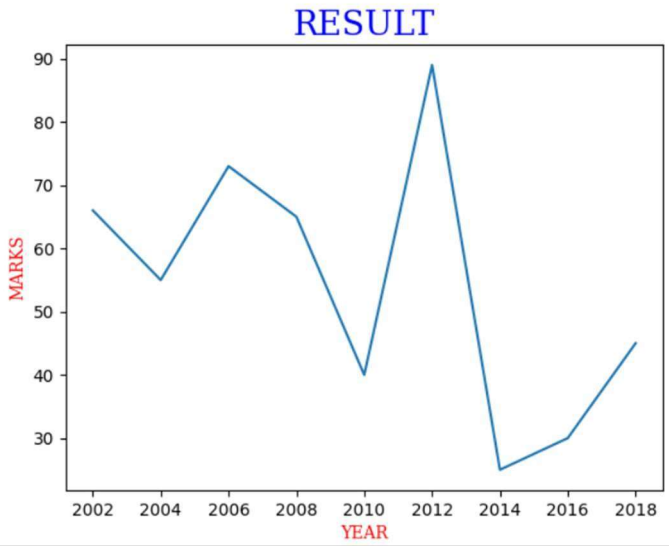
```
Text(0, 0.5, 'MARKS')
```



i. Set the font properties for title and labels : font family, color and size.

```
font1 = {'family': 'serif', 'color': 'blue', 'size': 20}
font2 = {'family': 'serif', 'color': 'red', 'size': 10}
plt.plot(x,y)
plt.title("RESULT",fontdict=font1)
plt.xlabel("YEAR",fontdict=font2)
plt.ylabel("MARKS", fontdict=font2)
```

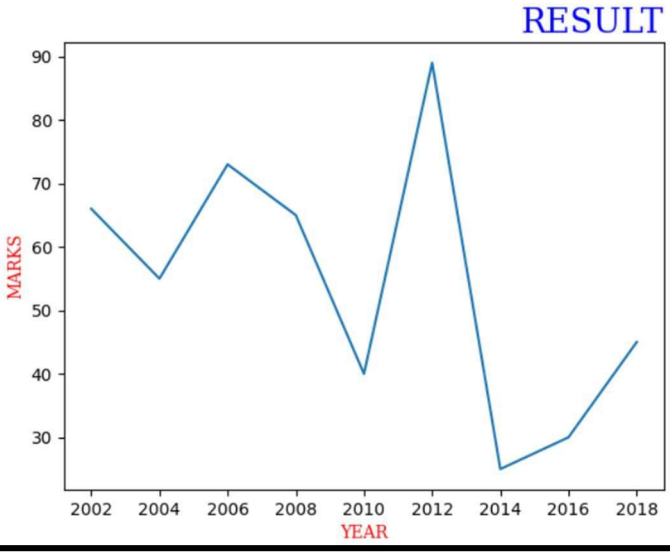
```
Text(0, 0.5, 'MARKS')
```



j. Change the position of title.

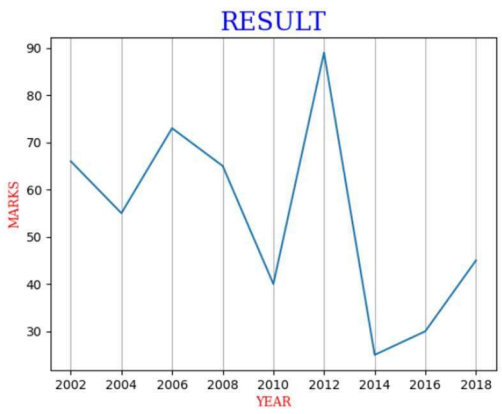
```
font1 = {'family': 'serif', 'color': 'blue', 'size': 20}
font2 = {'family': 'serif', 'color': 'red', 'size': 10}
plt.plot(x,y)
plt.title("RESULT", loc="right",fontdict=font1)
plt.xlabel("YEAR",fontdict=font2)
plt.ylabel("MARKS", fontdict=font2)
```

```
Text(0, 0.5, 'MARKS')
```

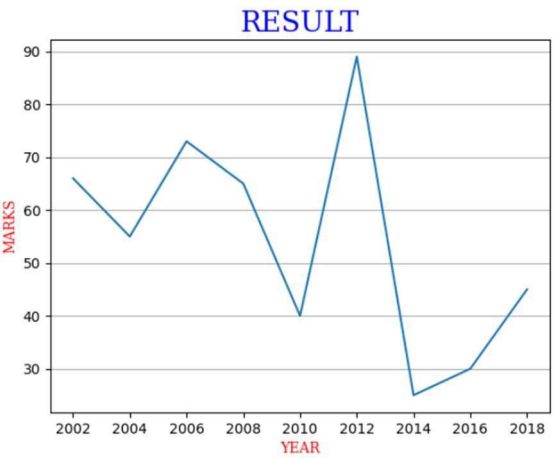


k. Add gridline to the plot(only x axis, only y axis, both).

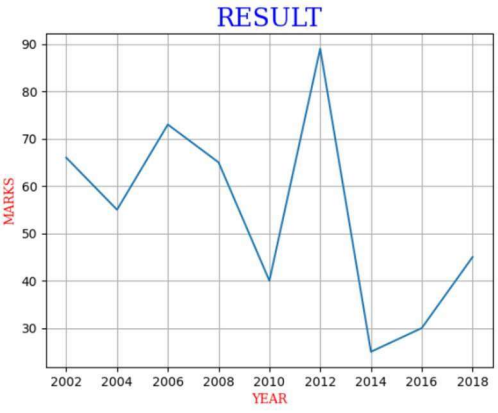
```
plt.title("RESULT",fontdict=font1)
plt.xlabel("YEAR",fontdict=font2)
plt.ylabel("MARKS", fontdict=font2)
plt.plot(x, y)
# Alternatively, you can turn on grid for specific axes:
plt.grid(True, axis='x')  # Turn on grid for x-axis only
plt.show()
```




```
plt.title("RESULT",fontdict=font1)
plt.xlabel("YEAR",fontdict=font2)
plt.ylabel("MARKS", fontdict=font2)
plt.plot(x, y)
plt.grid(True, axis='y')  # Turn on grid for y-axis only
```

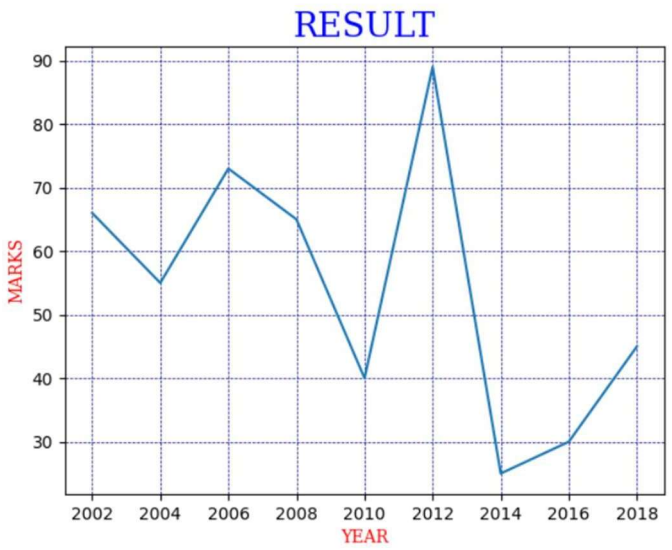



```
plt.title("RESULT",fontdict=font1)
plt.xlabel("YEAR",fontdict=font2)
plt.ylabel("MARKS", fontdict=font2)
plt.plot(x, y)
plt.grid()
```



l. Set grid properties : grid color, grid linestyle, grid line width.

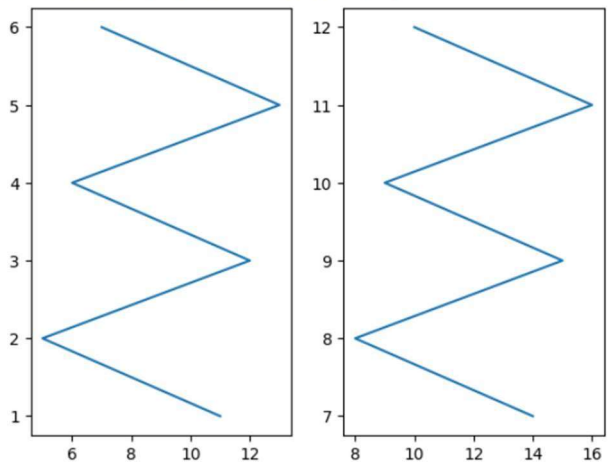
```
plt.title("RESULT",fontdict=font1)
plt.xlabel("YEAR",fontdict=font2)
plt.ylabel("MARKS", fontdict=font2)
plt.plot(x, y)
plt.grid(color = 'darkblue', linestyle ='--', linewidth = 0.5)

```



m. Display multiple plots using subplot() .

```
x = np.array([11,5,12,6,13,7])
y = np.array([1,2,3,4,5,6])
plt.subplot(1,2,1)
plt.plot(x,y)

x = np.array([14,8,15,9,16,10])
y = np.array([7,8,9,10,11,12])
plt.subplot(1,2,2)
plt.plot(x,y)
```

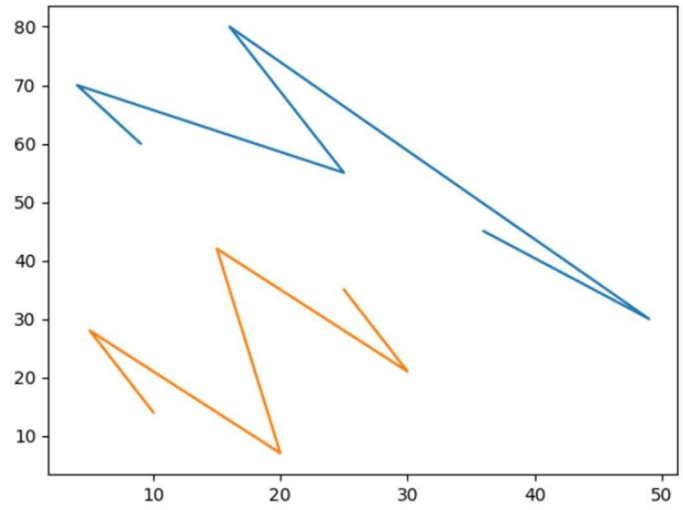
```
[<matplotlib.lines.Line2D at 0x7badca825a80>]
```



n. Display multiple lineplots in a single plot for comparison.

```
x = np.array([9,4,25,16,49,36])
y = np.array([60,70,55,80,30,45])
x1 = np.array([10,5,20,15,30,25])
y1 = np.array([14,28,7,42,21,35])
plt.plot(x,y,x1,y1)
```

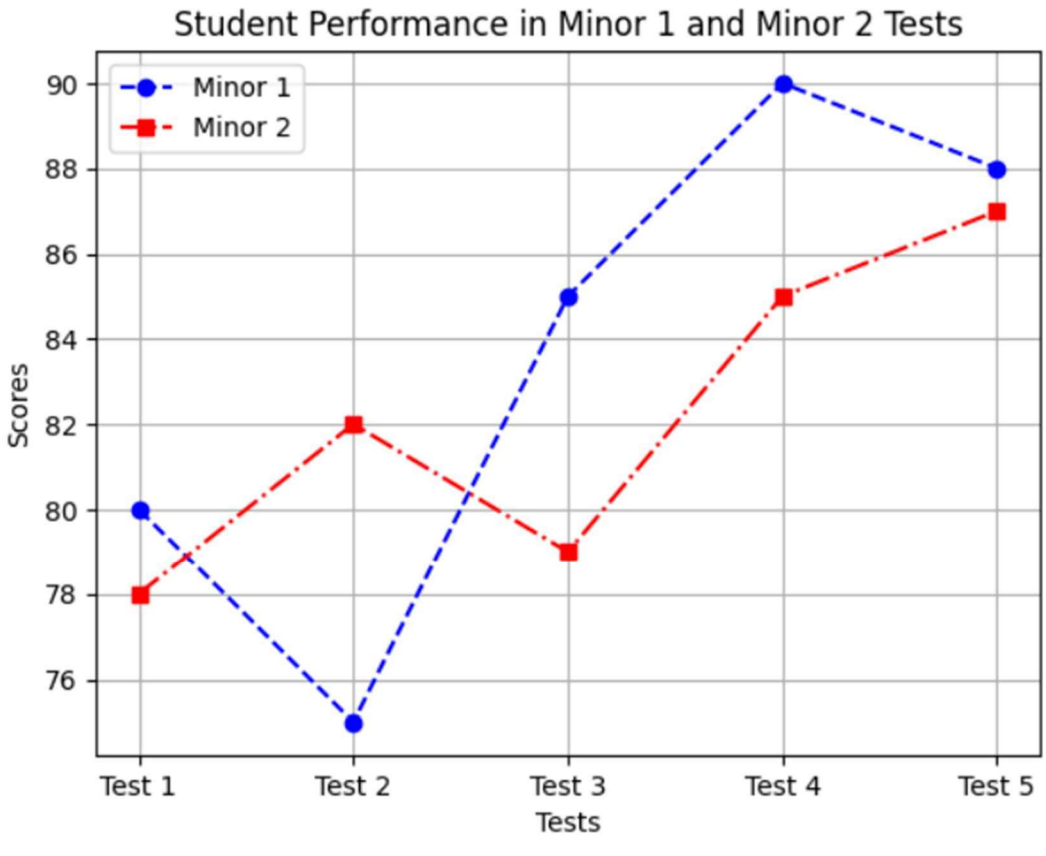
```
[<matplotlib.lines.Line2D at 0x7badc6437640>,
 <matplotlib.lines.Line2D at 0x7badc64341f0>]
```



QUESTION 4) Write a program to plot a line chart of student's performance in 5 tests of minor1 and minor2 by setting the line style and marker style.

SOLUTION :-

```python
import matplotlib.pyplot as plt

# Sample student performance data for minor1 and minor2 tests
tests = ['Test 1', 'Test 2', 'Test 3', 'Test 4', 'Test 5']
minor1_scores = [80, 75, 85, 90, 88]
minor2_scores = [78, 82, 79, 85, 87]

# Plotting the line chart with customized line and marker styles
plt.plot(tests, minor1_scores, linestyle='--', marker='o', color='blue', label='Minor 1')
plt.plot(tests, minor2_scores, linestyle='-.', marker='s', color='red', label='Minor 2')

# Adding labels and title
plt.xlabel('Tests')
plt.ylabel('Scores')
plt.title('Student Performance in Minor 1 and Minor 2 Tests')
plt.legend()

# Display the plot
plt.grid()  # Adding grid
plt.show()

```



QUESTION5) Write a program to implement a bar graph showing the no. of medals in Commonwealth game.

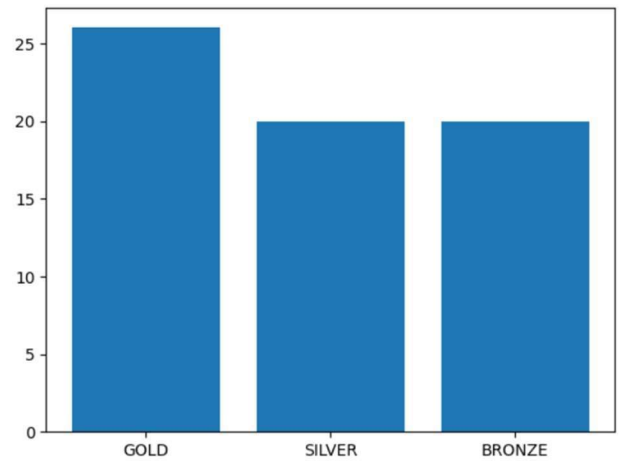
SOLUTION:

```
#WAP to implement a bar graph showing the no. of medels in common wealth game.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

a. Create a basic bar graph using bar().

```
x = np.array(['GOLD','SILVER','BRONZE'])
y = np.array ([26,20,20])
plt.bar(x,y)
```

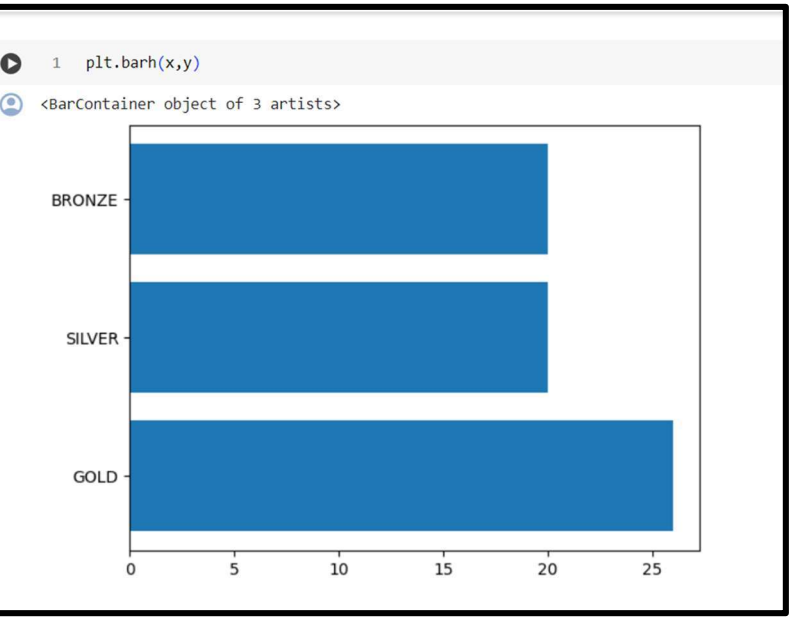
```
<BarContainer object of 3 artists>
```



b. Draw a bar graph horizontally using barh().

```
plt.barh(x,y)
```

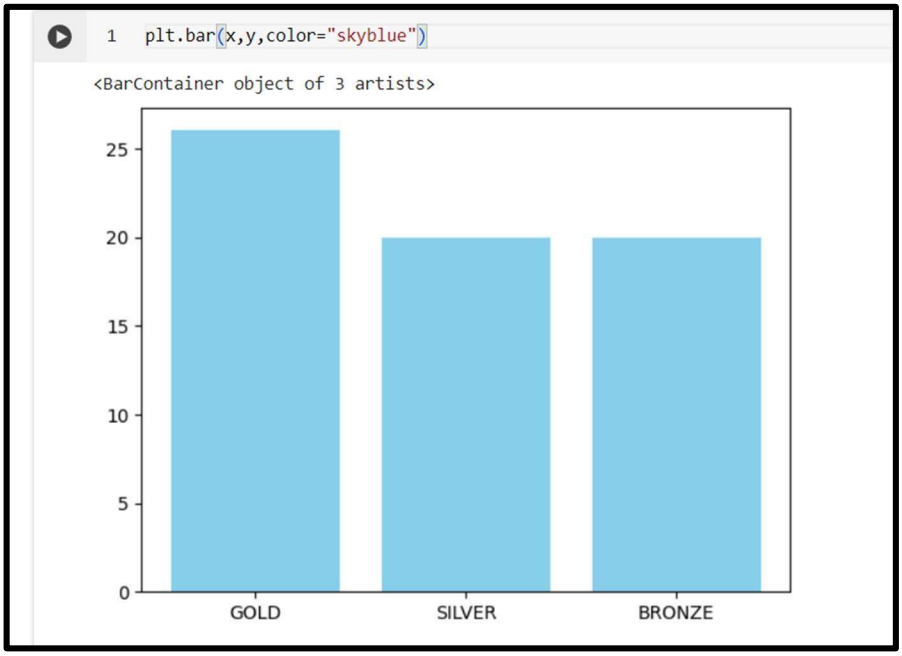
```
<BarContainer object of 3 artists>
```



c. change color of bar.

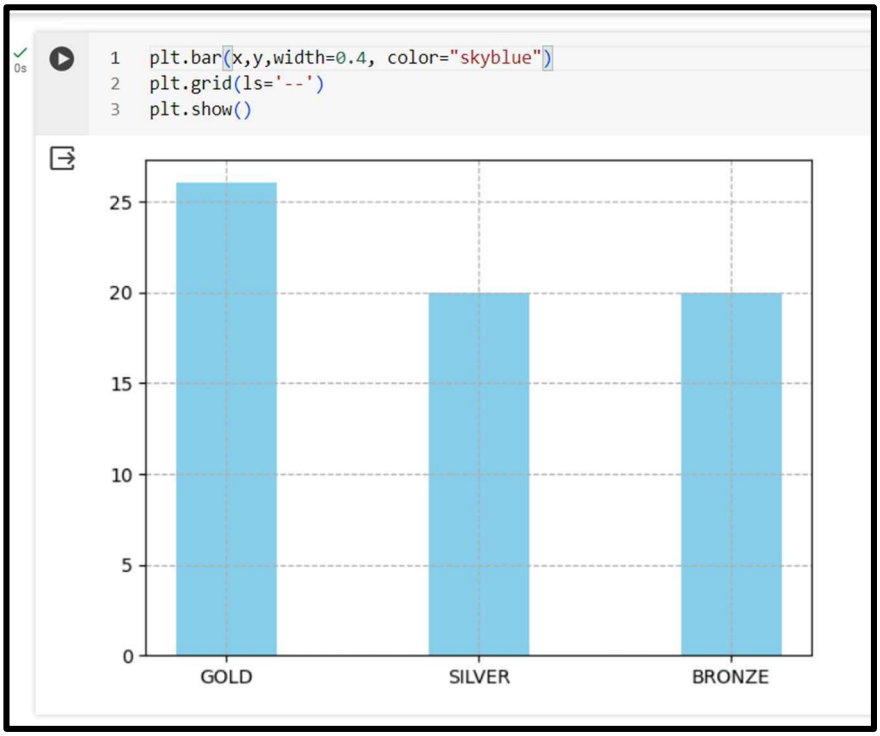
```
1  plt.bar(x,y,color="skyblue")
```

```
<BarContainer object of 3 artists>
```



d. change width of bar.

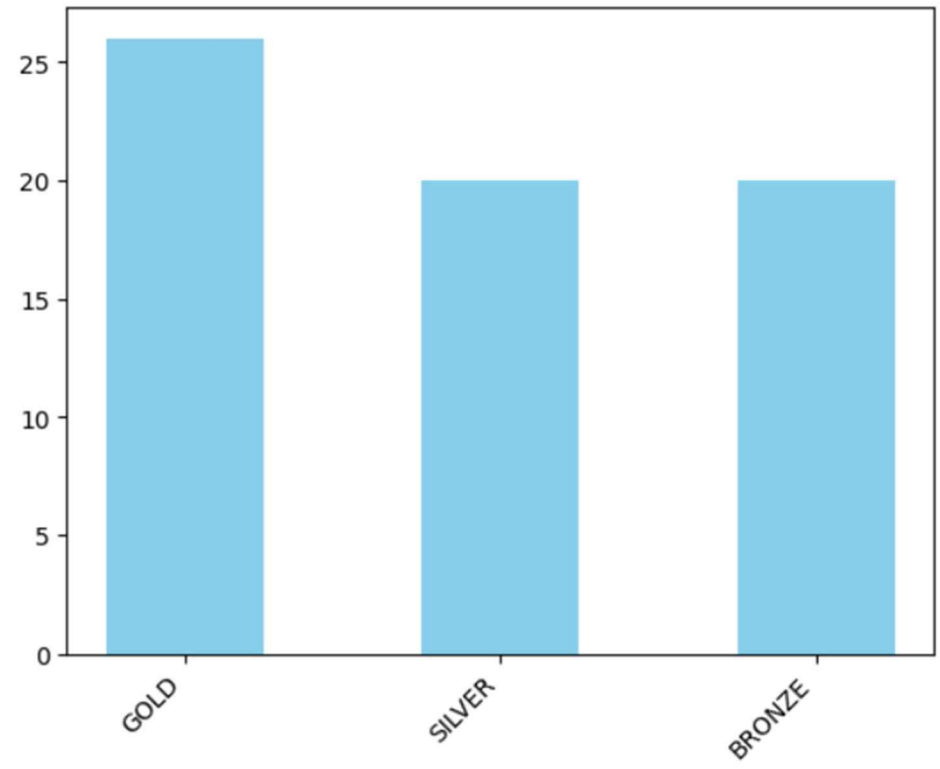
```
1  plt.bar(x,y,width=0.4, color="skyblue")
2  plt.grid(ls='--')
3  plt.show()
```



e. change the alignment of the labels.

```
[6]  1  plt.bar(x,y,width=0.5, color="skyblue")
     2  plt.xticks(rotation=45, ha='right')
```

```
([0, 1, 2], [Text(0, 0, 'GOLD'), Text(1, 0, 'SILVER'), Text(2, 0, 'BRONZE')])
```

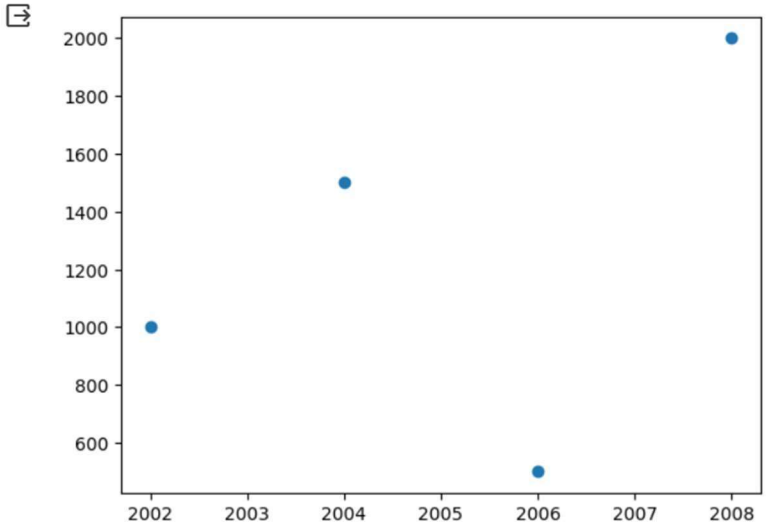


QUESTION 6) Draw a scatter plot that shows the stock trend for 5 years for TATA and Reliance company. Use the following properties:-

SOLUTION :-

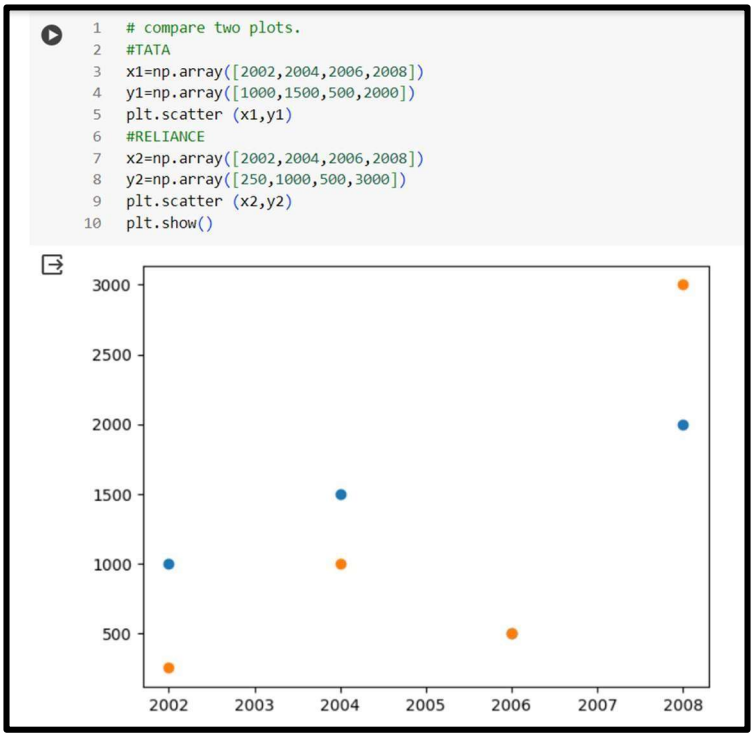
a. Draw the basic scatter graph using scatter()

```
# basic scatter graph
x=np.array([2002,2004,2006,2008])
y=np.array([1000,1500,500,2000])
plt.scatter (x,y)
plt.show()
```



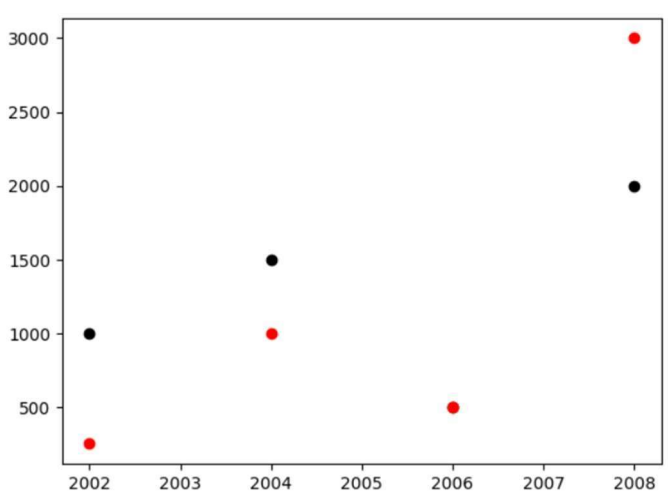
b. Compare two plots in a single plot.

```
# compare two plots.
#TATA
x1=np.array([2002,2004,2006,2008])
y1=np.array([1000,1500,500,2000])
plt.scatter (x1,y1)
#RELIANCE
x2=np.array([2002,2004,2006,2008])
y2=np.array([250,1000,500,3000])
plt.scatter (x2,y2)
plt.show()
```



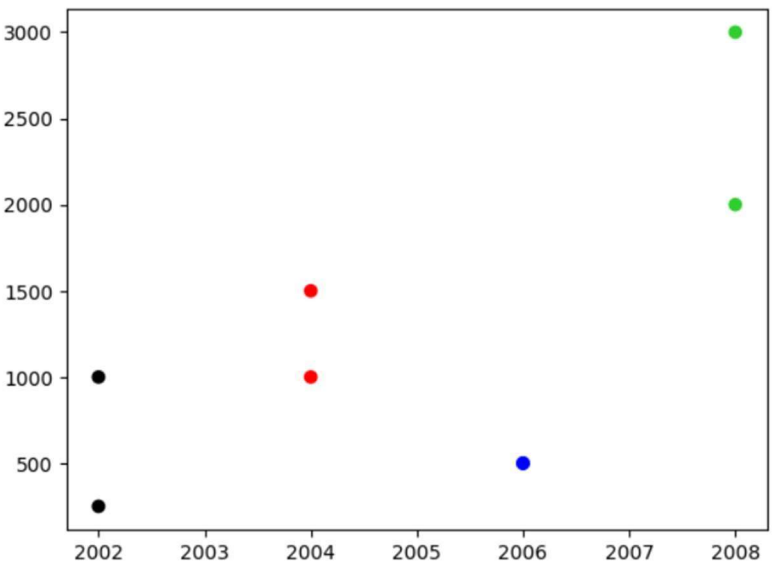
c. Set the color of both plots.

```
# set the color of dots in scatter graph
plt.scatter(x1,y1,color='black')
plt.scatter(x2,y2,color='red')
plt.show()
```



d. Set the different color of every dot.

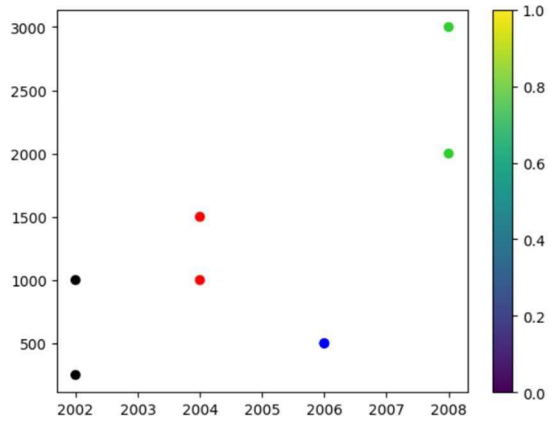
```
# assign different colors to every dot
mycolors=np.array(["black","red","blue","limegreen"])
plt.scatter(x1,y1,color=mycolors)
plt.scatter(x2,y2,color=mycolors)
plt.show()
```



e. Set colormap and display it using colorbar().

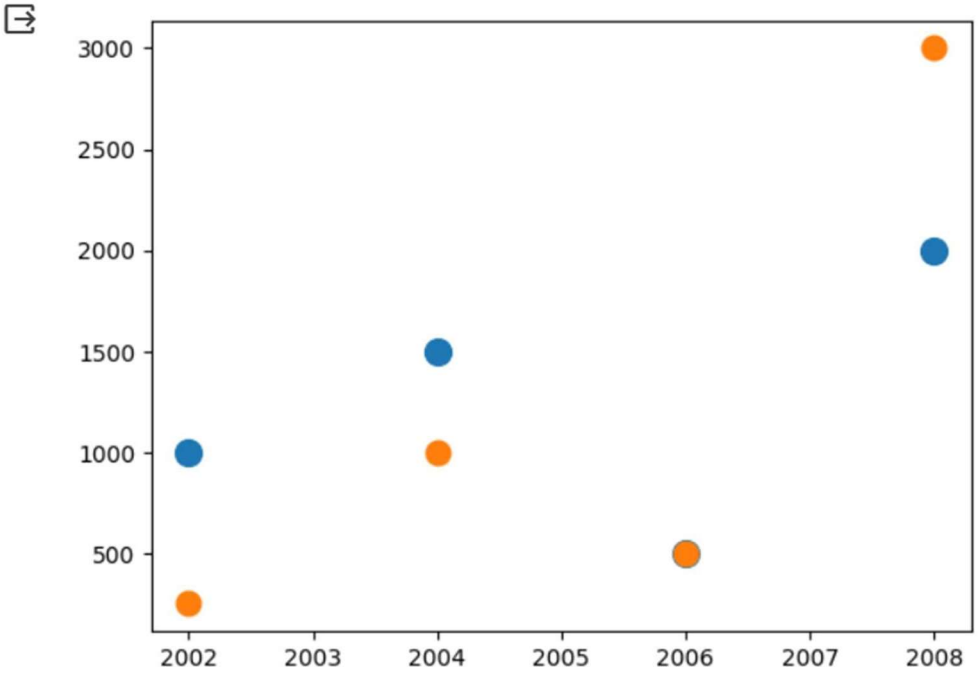
```
# color map
plt.scatter(x1,y1,color=mycolors,cmap='viridis')
plt.scatter(x2,y2,color=mycolors,cmap='viridis')
plt.colorbar()
plt.show()
```

```
<ipython-input-10-43e3a6bf464c>:2: UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored
  plt.scatter(x1,y1,color=mycolors,cmap='viridis')
<ipython-input-10-43e3a6bf464c>:3: UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored
  plt.scatter(x2,y2,color=mycolors,cmap='viridis')
```



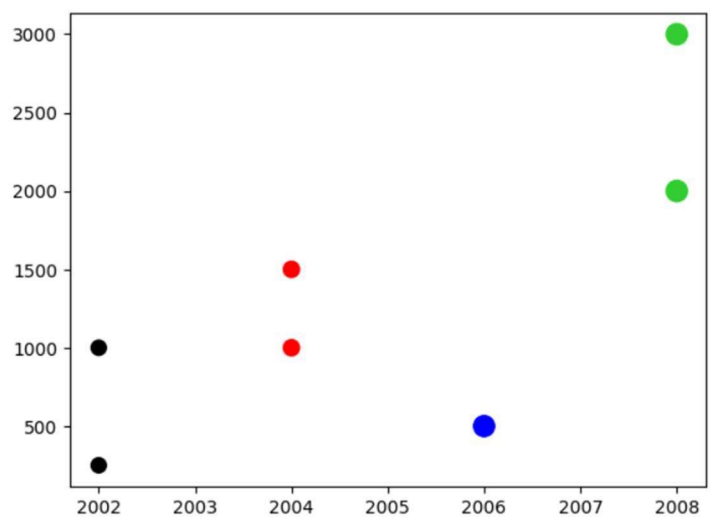
f. Set size of the dot.

```
# set the size of dot
plt.scatter(x1,y1,s=130)
plt.scatter(x2,y2,s=110)
plt.show()
```



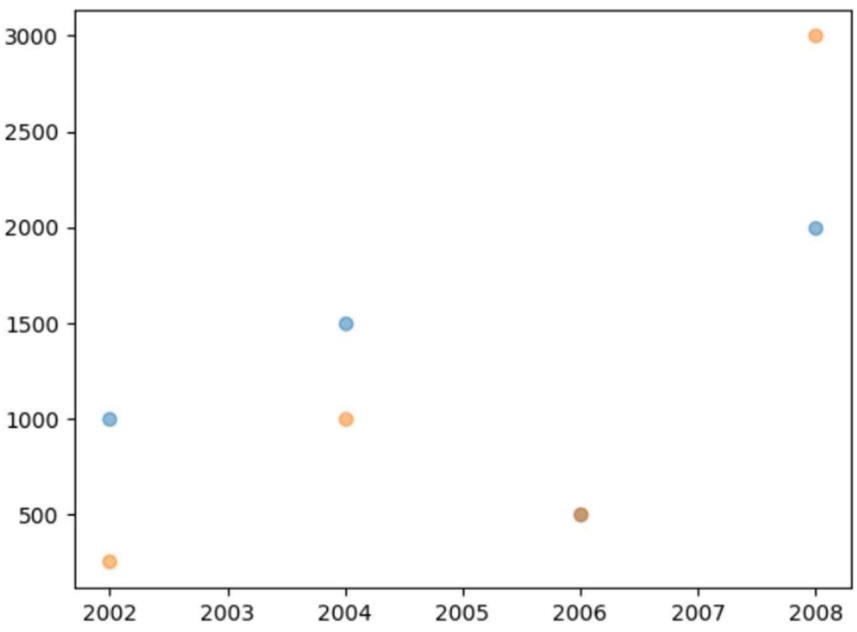
g. Set the different size of every dot.

```
# change size of every dot
mysizes=np.array([70,80,130,140])
mycolors=np.array(["black","red","blue","limegreen"])
plt.scatter (x1,y1,s=mysizes,color=mycolors)
plt.scatter (x2,y2,s=mysizes,color=mycolors)
plt.show()
```



h. Set the transparency of dot.

```
# change the transparency of the dot
plt.scatter(x1,y1,alpha=0.5)
plt.scatter(x2,y2,alpha=0.5)
plt.show()
```



QUESTION 7) Draw a pie chart that shows the sales of a different fruit in a day for a shop: 30% banana, 20% other, 15% orange, 10% guava, 25% mango Use the following properties:-

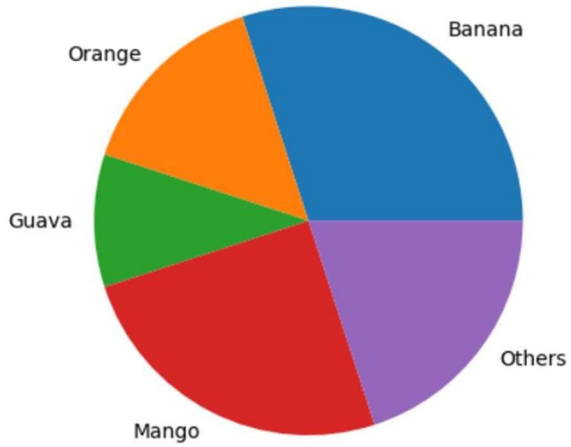
SOLUTION :-

a. Draw basic pie chart using pie().

```
y = np.array([30,15,10,25,20])
plt.pie(y)
plt.show()
```

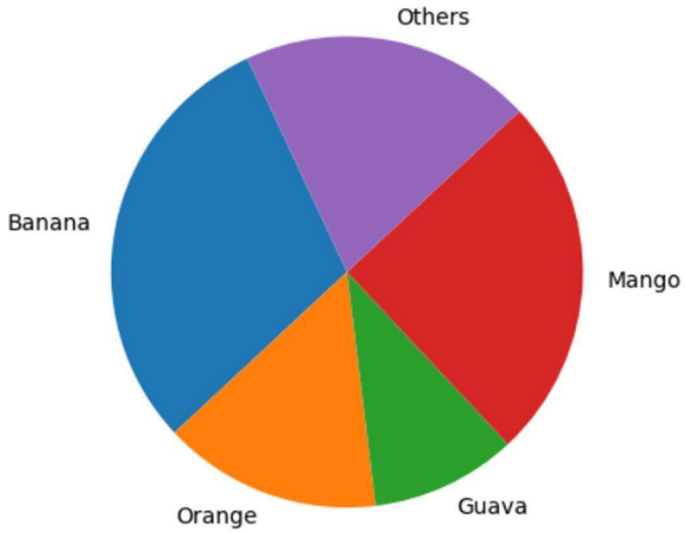
b. Add labels to wedges.

```
my_label = ["Banana", "Orange", "Guava", "Mango", "Others"]
plt.pie(y, labels=my_label)
plt.show()
```



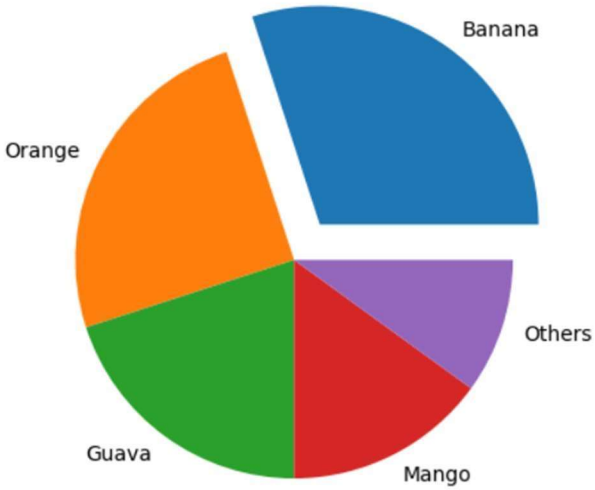
c. Change the start angle of any wedge.

```
my_label = ["Banana", "Orange", "Guava", "Mango", "Others"]
plt.pie(y, labels=my_label, startangle=115)
plt.show()
```



d. Displace the centre of wedge.

```
my_label = ["Banana", "Orange", "Guava", "Mango", "Others"]
y = [30, 25, 20, 15, 10]
my_explode = ([0.2, 0, 0, 0, 0])
plt.pie(y, labels=my_label, explode =my_explode)
plt.show()
```



6.	Draw a scatter plot that shows the stock trend for 5 years for TATA and Reliance company. Use the following properties:- a. Draw the basic scatter graph using scatter() b. Compare two plots in a single plot. c. Set the color of both plots. d. Set the different color of every dot. e. Set colormap and display it using colorbar(). f. Set size of the dot. g. Set the different size of every dot. h. Set the transparency of dot.	
7.	Draw a pie chart that shows the sales of a different fruit in a day for a shop: 30% banana, 20% other, 15% orange, 10% guava, 25% mango Use the following properties:- a. Draw basic pie chart using pie(). b. Add labels to wedges. c. Change the start angle of any wedge. d. Displace the centre of wedge. e. Add shadow to slice wedges. f. Change the colour of wedges. g. Add legend. h. Add legend with header.	
8.	Create a csv file displaying countries and no. Of companies in each country. Display a bar graph from this csv file and style the bar graph too.	
9.	Write a program to combine two dataframes (data contained in two csv files – f1.csv country, no. Of companies and f2.csv – country, no. Of employees) Using merge() , join(), concat().	
10.	Implement these graphs using seaborn library:- Lineplot, Distplot, Lmplot, Countplot, Relplot, Displot, Catplot	

Q QUESTION 1) Write a program to implement data preprocessing on student's dataset.

SOLUTION :-

```
[2]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
```

```
[4]  data=pd.read_csv('missing.csv')
     data
```

	Roll no	Name	Marks
0	11.0	abhi	34.0
1	12.0	amrit	54.0
2	13.0	NaN	54.0
3	14.0	ansh	NaN
4	15.0	nim	67.0
5	NaN	NaN	NaN
6	17.0	shu	54.0
7	18.0	NaN	65.0
8	19.0	charu	NaN
9	20.0	sham	67.0

a) Delete the row containing null values.

```
df_clear=data.dropna()
df_clear
```

	Roll no	Name	Marks
0	11.0	abhi	34.0
1	12.0	amrit	54.0
4	15.0	nim	67.0
6	17.0	shu	54.0
9	20.0	sham	67.0

b) Delete the row containing all null values.

```
df_clear=data.dropna(how='all')
df_clear
```

	Roll no	Name	Marks
0	11.0	abhi	34.0
1	12.0	amrit	54.0
2	13.0	NaN	54.0
3	14.0	ansh	NaN
4	15.0	nim	67.0
6	17.0	shu	54.0
7	18.0	NaN	65.0
8	19.0	charu	NaN
9	20.0	sham	67.0

c) Replace the null values using forward method.

```
[13] data.fillna(method='ffill')
```

	Roll no	Name	Marks
0	11.0	abhi	34.0
1	12.0	amrit	54.0
2	13.0	amrit	54.0
3	14.0	ansh	54.0
4	15.0	nim	67.0
5	15.0	nim	67.0
6	17.0	shu	54.0
7	18.0	shu	65.0
8	19.0	charu	65.0
9	20.0	sham	67.0

d) Replace the null values using backward method.

```
[13] data.fillna(method='ffill')
```

	Roll no	Name	Marks
0	11.0	abhi	34.0
1	12.0	amrit	54.0
2	13.0	amrit	54.0
3	14.0	ansh	54.0
4	15.0	nim	67.0
5	15.0	nim	67.0
6	17.0	shu	54.0
7	18.0	shu	65.0
8	19.0	charu	65.0
9	20.0	sham	67.0

e) Replace with constant value.

```
data.fillna(1)
```

	Roll no	Name	Marks
0	11.0	abhi	34.0
1	12.0	amrit	54.0
2	13.0	1	54.0
3	14.0	ansh	1.0
4	15.0	nim	67.0
5	1.0	1	1.0
6	17.0	shu	54.0
7	18.0	1	65.0
8	19.0	charu	1.0
9	20.0	sham	67.0

f) Replace with central tendency measures – mean, mode, medium.

```
mean_value = data['Marks'].mean()
data['Marks'] = data['Marks'].fillna(mean_value)
data['Marks']
```

```
0    34.000000
1    54.000000
2    54.000000
3    56.428571
4    67.000000
5    56.428571
6    54.000000
7    65.000000
8    56.428571
9    67.000000
Name: Marks, dtype: float64
```

```
median_value = data['Marks'].median()
data['Marks'] = data['Marks'].fillna(median_value)
data['Marks']
```

```
0    34.000000
1    54.000000
2    54.000000
3    56.428571
4    67.000000
5    56.428571
6    54.000000
7    65.000000
8    56.428571
9    67.000000
Name: Marks, dtype: float64
```

```
data['Marks'].fillna(data['Marks'].mode()[0])
```

```
0    34.000000
1    54.000000
2    54.000000
3    56.428571
4    67.000000
5    56.428571
6    54.000000
7    65.000000
8    56.428571
9    67.000000
Name: Marks, dtype: float64
```

QUESTION2) Implement data preprocessing on the given e commerce dataset:

SOLUTION :-

+ Code + Text

```
1  import pandas as pd
2  import numpy as np
3
4  # Read the CSV file into a DataFrame
5  dataset = pd.read_csv('data.csv')
6
7  # Display the dataset
8  dataset
9
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

a. Describe the data.

a. Head

```
1 dataset.head()
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes

b. Tail

```
1 dataset.tail()
```

	Country	Age	Salary	Purchased
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

c. Shape

```
1 dataset.shape
```

```
(10, 4)
```

d. Info

```
[4]  1  dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Country    10 non-null     object
 1   Age        9 non-null      float64
 2   Salary     9 non-null      float64
 3   Purchased  10 non-null     object
dtypes: float64(2), object(2)
memory usage: 448.0+ bytes
```

b. Check whether null values are present or not and display column wise also.

```
1  dataset.isnull()
```

	Country	Age	Salary	Purchased
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	True	False
5	False	False	False	False
6	False	True	False	False
7	False	False	False	False
8	False	False	False	False
9	False	False	False	False

c. Extract the independent and dependent variables.

```
[ ]  1  x =dataset.iloc[ :,:-1].values
```

```
[ ]  1  x
```

```
array([['France', 44.0, 72000.0],
       ['Spain', 27.0, 48000.0],
       ['Germany', 30.0, 54000.0],
       ['Spain', 38.0, 61000.0],
       ['Germany', 40.0, nan],
       ['France', 35.0, 58000.0],
       ['Spain', nan, 52000.0],
       ['France', 48.0, 79000.0],
       ['Germany', 50.0, 83000.0],
       ['France', 37.0, 67000.0]], dtype=object)
```

```
[ ]  1  y =dataset.iloc[ :,-1: ].values
```

```
[ ]  1  y
```

```
array([['No'],
       ['Yes'],
       ['No'],
       ['No'],
       ['Yes'],
       ['Yes'],
       ['No'],
       ['Yes'],
       ['No'],
       ['Yes']], dtype=object)
```

d. Handle the missing data.

```
1  dataset.dropna(how='any')
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
5	France	35.0	58000.0	Yes
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
1  dataset.dropna(how='all')
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes
5	France	35.0	58000.0	Yes
6	Spain	NaN	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
[ ]  1  dataset.fillna(method='bfill')
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	58000.0	Yes
5	France	35.0	58000.0	Yes
6	Spain	48.0	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
[ ]  1  dataset.fillna(method="ffill")
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	61000.0	Yes
5	France	35.0	58000.0	Yes
6	Spain	35.0	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
[ ]  1  dataset.fillna(0)
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	0.0	Yes
5	France	35.0	58000.0	Yes
6	Spain	0.0	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
1  dataset.fillna(dataset.mean(numeric_only=True))
```

	Country	Age	Salary	Purchased
0	France	44.000000	72000.000000	No
1	Spain	27.000000	48000.000000	Yes
2	Germany	30.000000	54000.000000	No
3	Spain	38.000000	61000.000000	No
4	Germany	40.000000	63777.777778	Yes
5	France	35.000000	58000.000000	Yes
6	Spain	38.777778	52000.000000	No
7	France	48.000000	79000.000000	Yes
8	Germany	50.000000	83000.000000	No
9	France	37.000000	67000.000000	Yes

```
1  dataset.fillna(dataset.mode())
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	61000.0	Yes
5	France	35.0	58000.0	Yes
6	Spain	44.0	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

```
[ ]  1  dataset.fillna
     2  (dataset.median(numeric_only=True))
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	61000.0	Yes
5	France	35.0	58000.0	Yes
6	Spain	38.0	52000.0	No
7	France	48.0	79000.0	Yes
8	Germany	50.0	83000.0	No
9	France	37.0	67000.0	Yes

e. Split into training and test data.

```
[ ]  1  x_train, x_test, y_train, y_test = train_test_split (x,y, test_size=0.2, random_state=0)
```

```
1  x_train
```

```
array([['Germany', 40.0, nan],
       ['France', 37.0, 67000.0],
       ['Spain', 27.0, 48000.0],
       ['Spain', nan, 52000.0],
       ['France', 48.0, 79000.0],
       ['Spain', 38.0, 61000.0],
       ['France', 44.0, 72000.0],
       ['France', 35.0, 58000.0]], dtype=object)
```

```
[ ]  1  x_test
```

```
array([['Germany', 30.0, 54000.0],
       ['Germany', 50.0, 83000.0]], dtype=object)
```

```
[ ]  1  y_train
```

```
array([['Yes'],
       ['Yes'],
       ['Yes'],
       ['No'],
       ['Yes'],
       ['No'],
       ['No'],
       ['Yes']], dtype=object)
```

```
[ ]  1  y_test
```

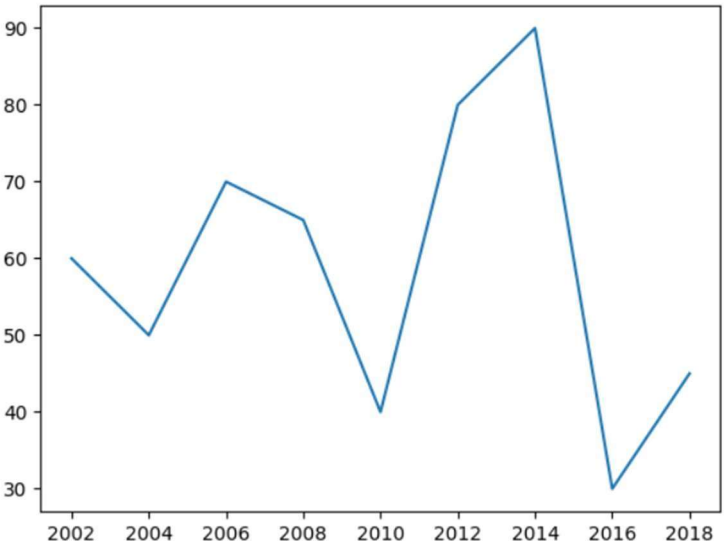
```
array([['No'],
       ['No']], dtype=object)
```

QUESTION3) Write a program to implement line plots by comparing performances of computer science subject of different years.

a. Draw a basic line plot using plot()

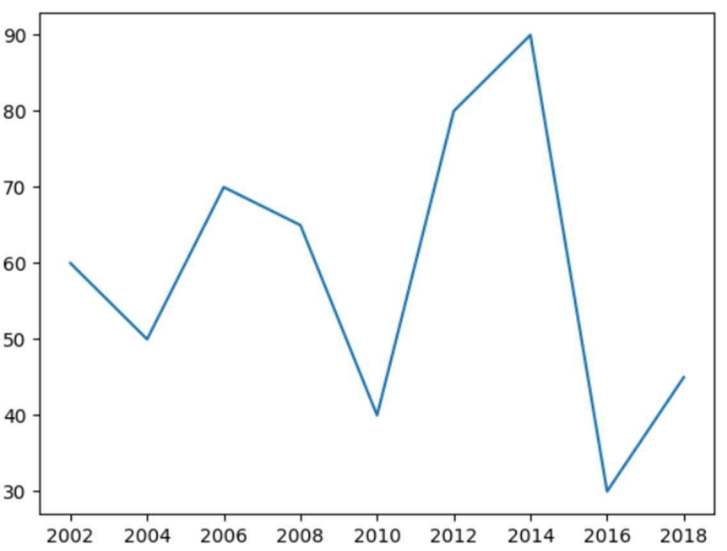
```
x=np.array([2002,2004,2006,2008,2010,2012,2014,2016,2018])
y=np.array([60,50,70,65,40,80,90,30,45])
plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x7a74eeeb4850>]
```



b. Display a line plot using show()

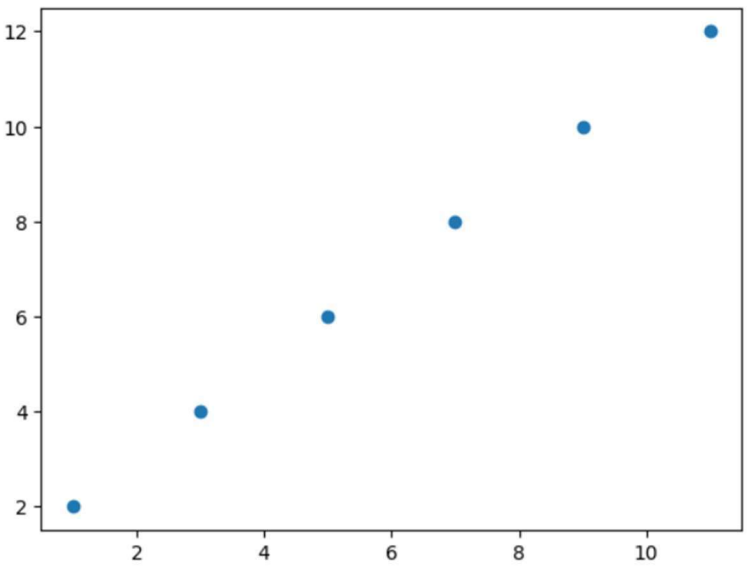
```
x = np.array([2002, 2004, 2006, 2008, 2010, 2012, 2014, 2016, 2018])
y = np.array([60, 50, 70, 65, 40, 80, 90, 30, 45])

plt.plot(x, y)
plt.show()
```



c. Display markers (initial and final endpoints)

```
x = np.array([1,3,5,7,9,11])
y = np.array([2,4,6,8,10,12])
plt.plot(x,y,'o')
```

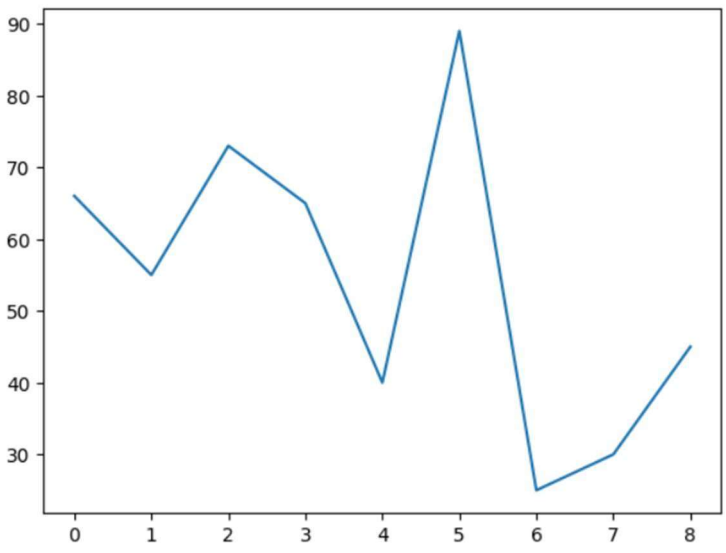
```
[<matplotlib.lines.Line2D at 0x7f53c890a800>]
```



d. Take only y data.

```
x=np.array([2002,2004,2006,2008,2010,2012,2014,2016,2018])
y=np.array([66,55,73,65,40,89,25,30,45])
plt.plot(y)
```

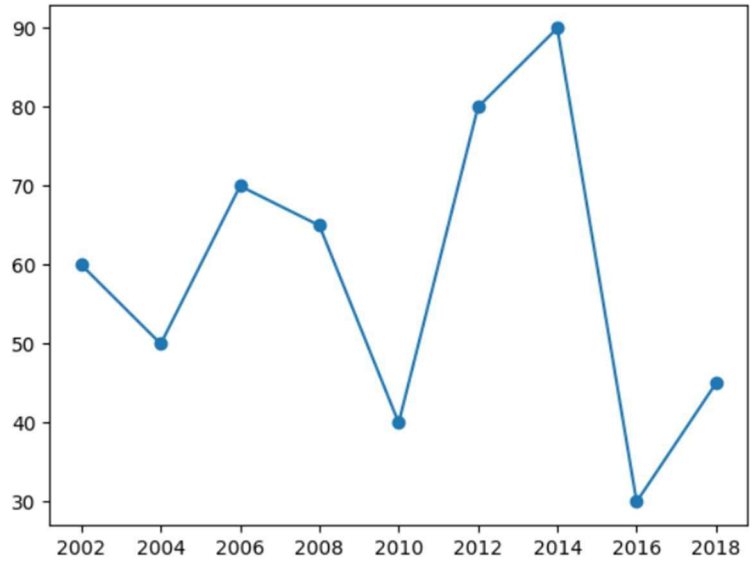
```
[<matplotlib.lines.Line2D at 0x7f53c7c16110>]
```



e. Display markers on every point.

```
x=np.array([2002,2004,2006,2008,2010,2012,2014,2016,2018])
y=np.array([60,50,70,65,40,80,90,30,45])
plt.plot(x,y,marker='o')
```

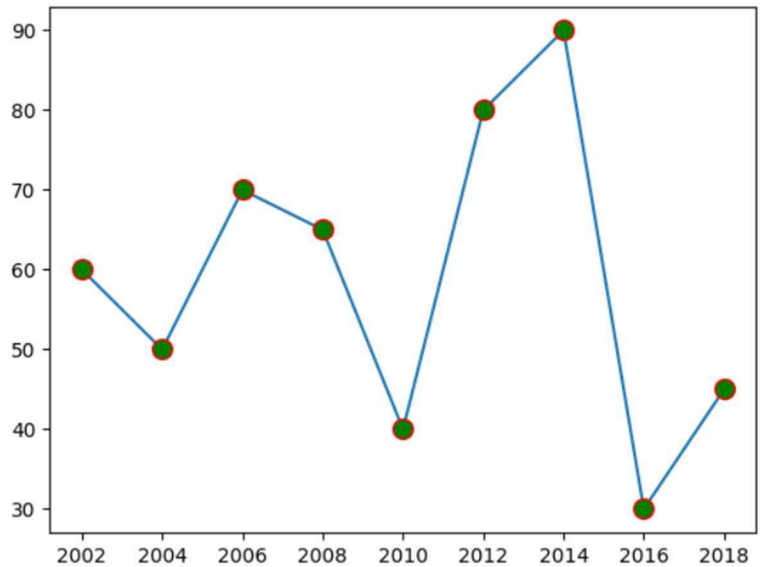
```
[<matplotlib.lines.Line2D at 0x7f53c7a9a620>]
```



f. Set the marker properties : set the marker color, size, edge color, face color.

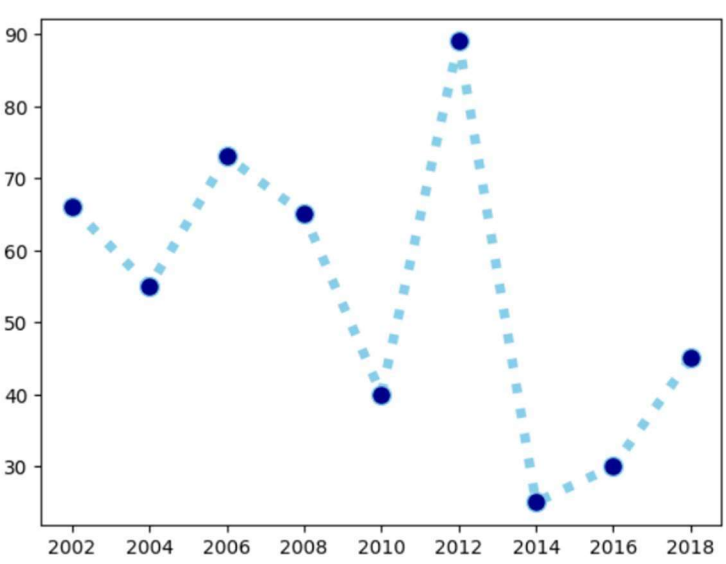
```
x=np.array([2002,2004,2006,2008,2010,2012,2014,2016,2018])
y=np.array([60,50,70,65,40,80,90,30,45])
plt.plot(x,y,marker='o',ms='10',mec='r',mfc='g')
```

```
[<matplotlib.lines.Line2D at 0x7f53c795f430>]
```



g. Set the line properties: line style, line color, line width.

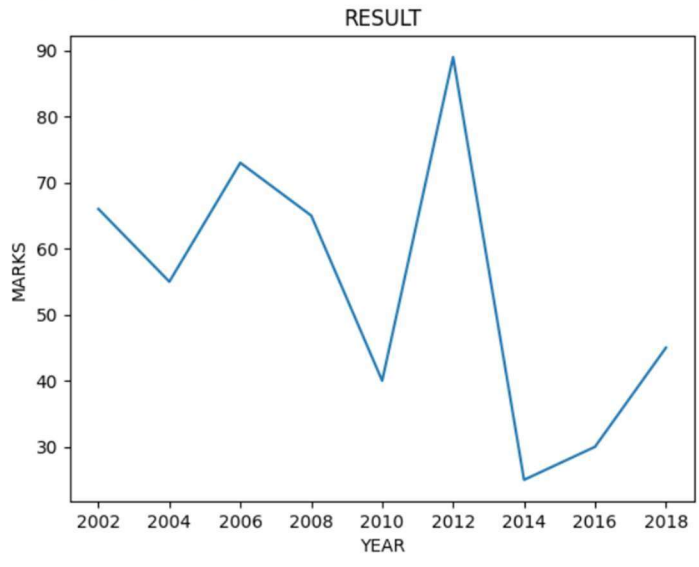
```
x=np.array([2002,2004,2006,2008,2010,2012,2014,2016,2018])
y=np.array([66,55,73,65,40,89,25,30,45])
plt.plot(x, y, ls='dotted', c='skyblue', lw='5', marker='o', ms='10',mfc='darkblue')
plt.show()
```



h. Set the x axis label, y axis label and title of the graph.

```
plt.plot(x,y)
plt.title("RESULT")
plt.xlabel("YEAR")
plt.ylabel("MARKS")
```

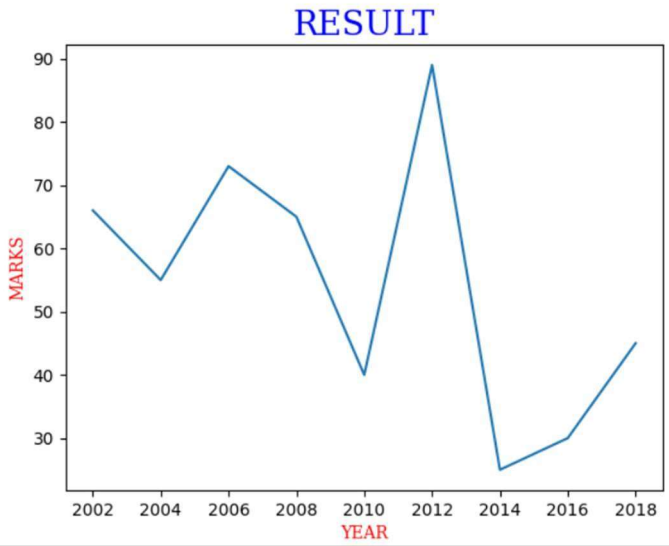
```
Text(0, 0.5, 'MARKS')
```



i. Set the font properties for title and labels : font family, color and size.

```
font1 = {'family': 'serif', 'color': 'blue', 'size': 20}
font2 = {'family': 'serif', 'color': 'red', 'size': 10}
plt.plot(x,y)
plt.title("RESULT",fontdict=font1)
plt.xlabel("YEAR",fontdict=font2)
plt.ylabel("MARKS", fontdict=font2)
```

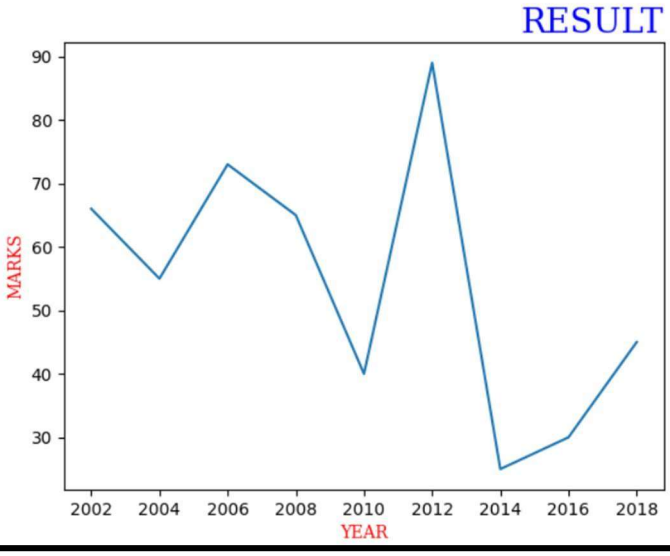
```
Text(0, 0.5, 'MARKS')
```



j. Change the position of title.

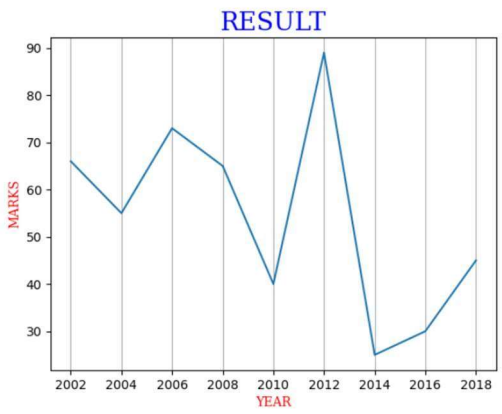
```
font1 = {'family': 'serif', 'color': 'blue', 'size': 20}
font2 = {'family': 'serif', 'color': 'red', 'size': 10}
plt.plot(x,y)
plt.title("RESULT", loc="right",fontdict=font1)
plt.xlabel("YEAR",fontdict=font2)
plt.ylabel("MARKS", fontdict=font2)
```

```
Text(0, 0.5, 'MARKS')
```

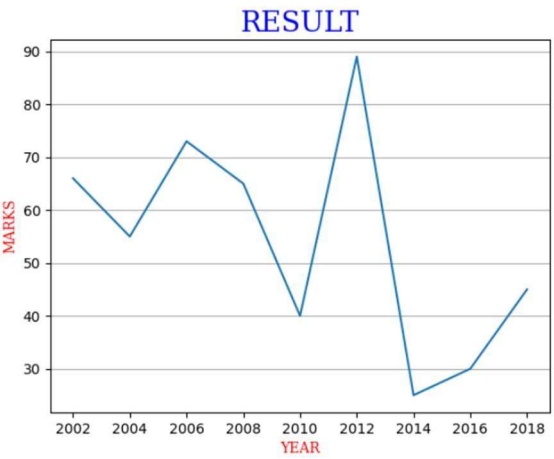


k. Add gridline to the plot(only x axis, only y axis, both).

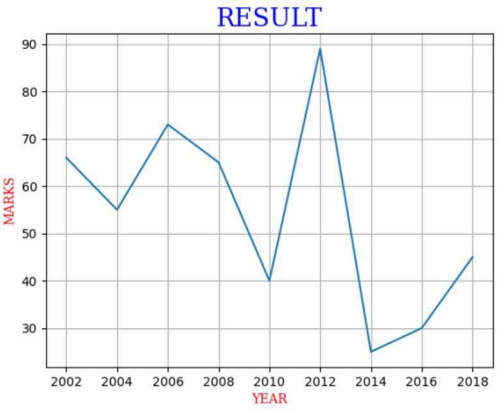
```
plt.title("RESULT",fontdict=font1)
plt.xlabel("YEAR",fontdict=font2)
plt.ylabel("MARKS", fontdict=font2)
plt.plot(x, y)
# Alternatively, you can turn on grid for specific axes:
plt.grid(True, axis='x')  # Turn on grid for x-axis only
plt.show()
```




```
plt.title("RESULT",fontdict=font1)
plt.xlabel("YEAR",fontdict=font2)
plt.ylabel("MARKS", fontdict=font2)
plt.plot(x, y)
plt.grid(True, axis='y')  # Turn on grid for y-axis only
```

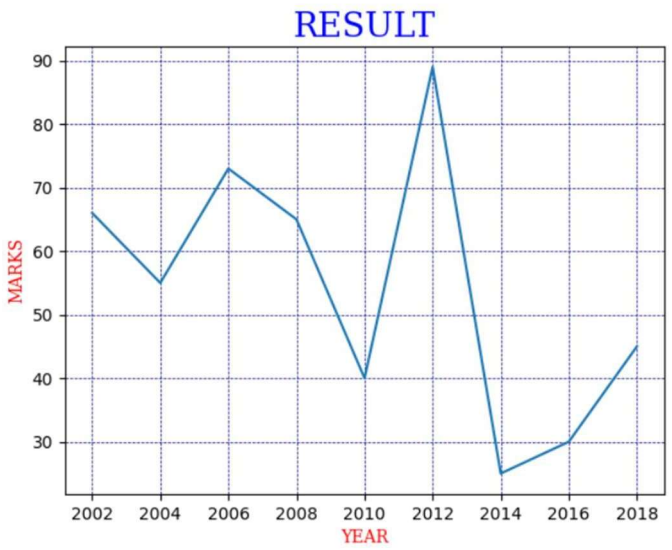



```
plt.title("RESULT",fontdict=font1)
plt.xlabel("YEAR",fontdict=font2)
plt.ylabel("MARKS", fontdict=font2)
plt.plot(x, y)
plt.grid()
```



l. Set grid properties : grid color, grid linestyle, grid line width.

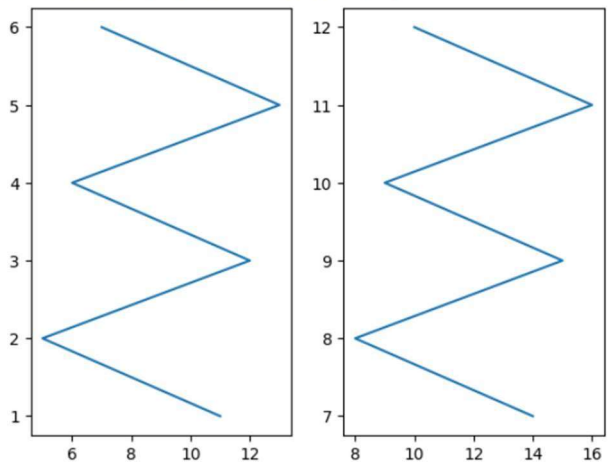
```
plt.title("RESULT",fontdict=font1)
plt.xlabel("YEAR",fontdict=font2)
plt.ylabel("MARKS", fontdict=font2)
plt.plot(x, y)
plt.grid(color = 'darkblue', linestyle ='--', linewidth = 0.5)

```



m. Display multiple plots using subplot() .

```
x = np.array([11,5,12,6,13,7])
y = np.array([1,2,3,4,5,6])
plt.subplot(1,2,1)
plt.plot(x,y)

x = np.array([14,8,15,9,16,10])
y = np.array([7,8,9,10,11,12])
plt.subplot(1,2,2)
plt.plot(x,y)
```

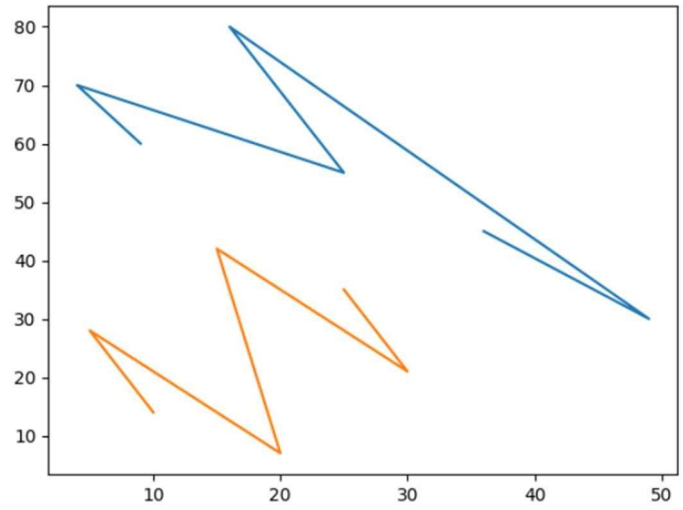
```
[<matplotlib.lines.Line2D at 0x7badca825a80>]
```



n. Display multiple lineplots in a single plot for comparison.

```
x = np.array([9,4,25,16,49,36])
y = np.array([60,70,55,80,30,45])
x1 = np.array([10,5,20,15,30,25])
y1 = np.array([14,28,7,42,21,35])
plt.plot(x,y,x1,y1)
```

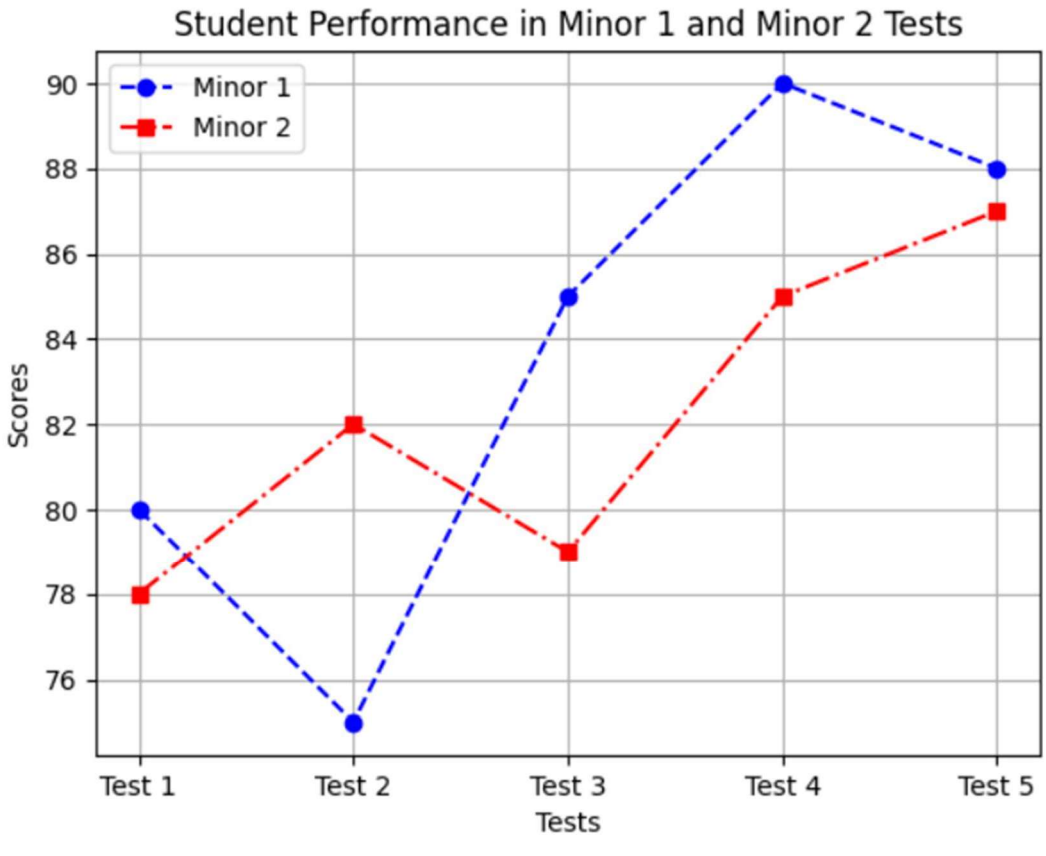
```
[<matplotlib.lines.Line2D at 0x7badc6437640>,
 <matplotlib.lines.Line2D at 0x7badc64341f0>]
```



QUESTION 4) Write a program to plot a line chart of student's performance in 5 tests of minor1 and minor2 by setting the line style and marker style.

SOLUTION :-

```python
import matplotlib.pyplot as plt

# Sample student performance data for minor1 and minor2 tests
tests = ['Test 1', 'Test 2', 'Test 3', 'Test 4', 'Test 5']
minor1_scores = [80, 75, 85, 90, 88]
minor2_scores = [78, 82, 79, 85, 87]

# Plotting the line chart with customized line and marker styles
plt.plot(tests, minor1_scores, linestyle='--', marker='o', color='blue', label='Minor 1')
plt.plot(tests, minor2_scores, linestyle='-.', marker='s', color='red', label='Minor 2')

# Adding labels and title
plt.xlabel('Tests')
plt.ylabel('Scores')
plt.title('Student Performance in Minor 1 and Minor 2 Tests')
plt.legend()

# Display the plot
plt.grid()  # Adding grid
plt.show()

```



QUESTION5) Write a program to implement a bar graph showing the no. of medals in Commonwealth game.

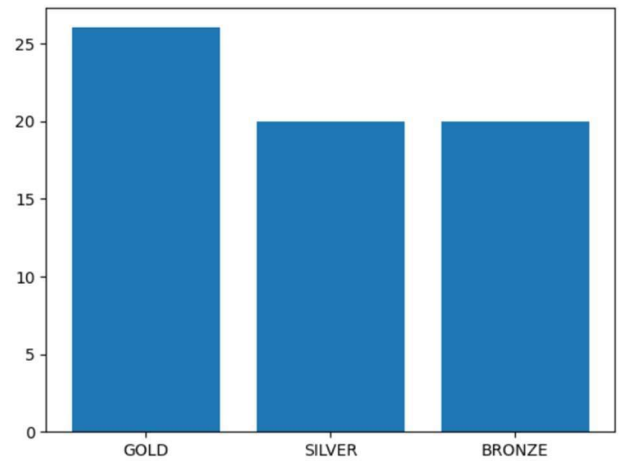
SOLUTION:

```
#WAP to implement a bar graph showing the no. of medels in common wealth game.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

a. Create a basic bar graph using bar().

```
x = np.array(['GOLD','SILVER','BRONZE'])
y = np.array ([26,20,20])
plt.bar(x,y)
```

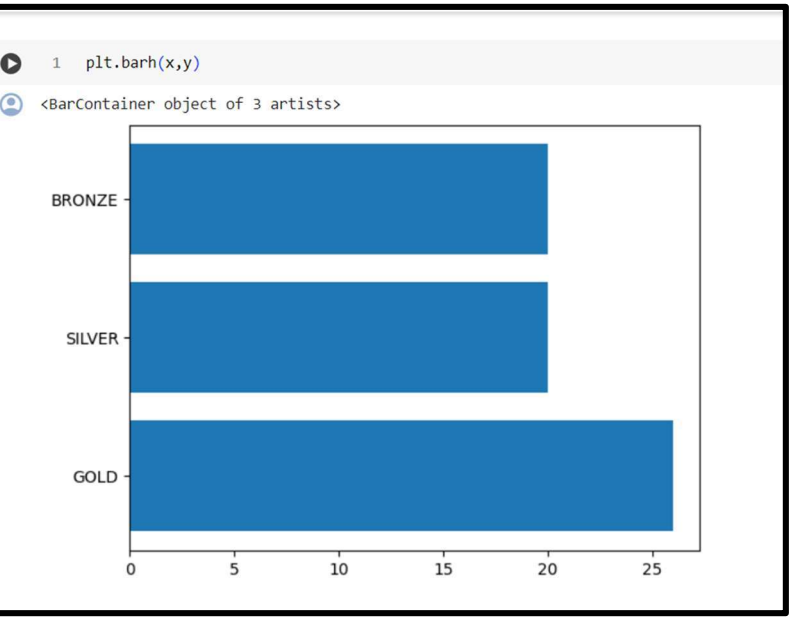
```
<BarContainer object of 3 artists>
```



b. Draw a bar graph horizontally using barh().

```
plt.barh(x,y)
```

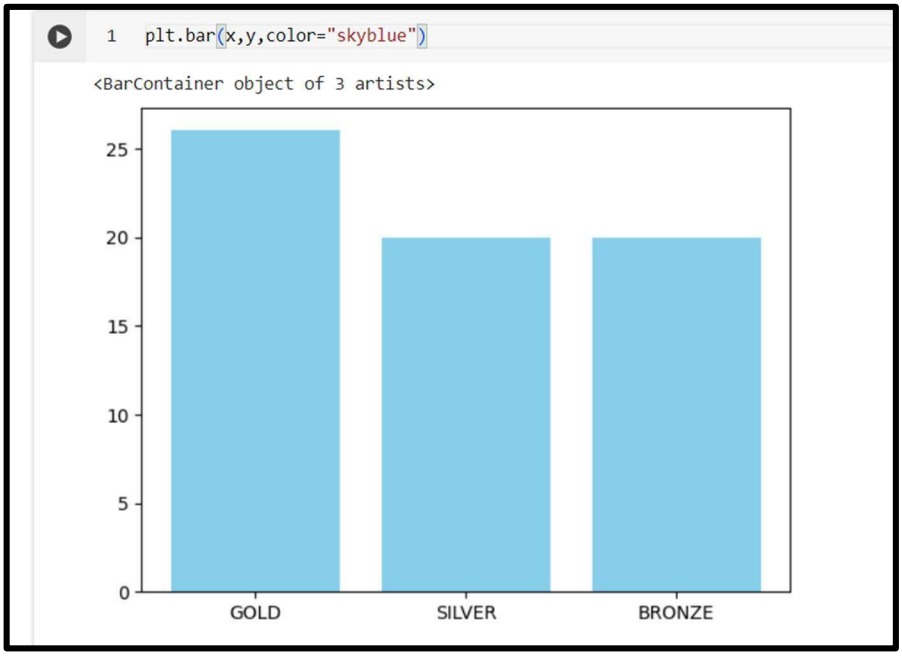
```
<BarContainer object of 3 artists>
```



c. change color of bar.

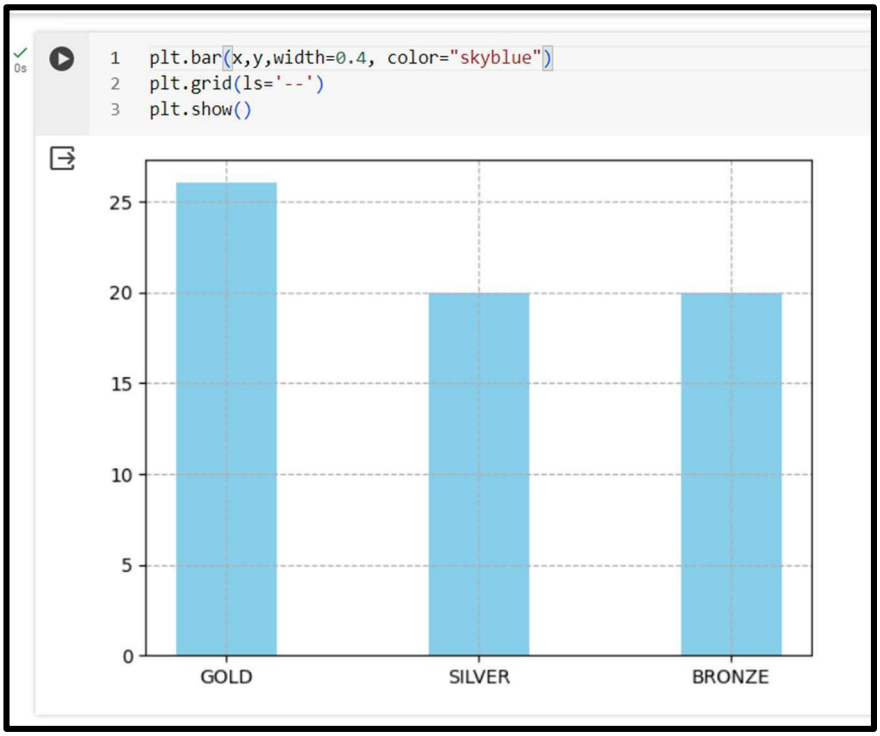
```
1  plt.bar(x,y,color="skyblue")
```

```
<BarContainer object of 3 artists>
```



d. change width of bar.

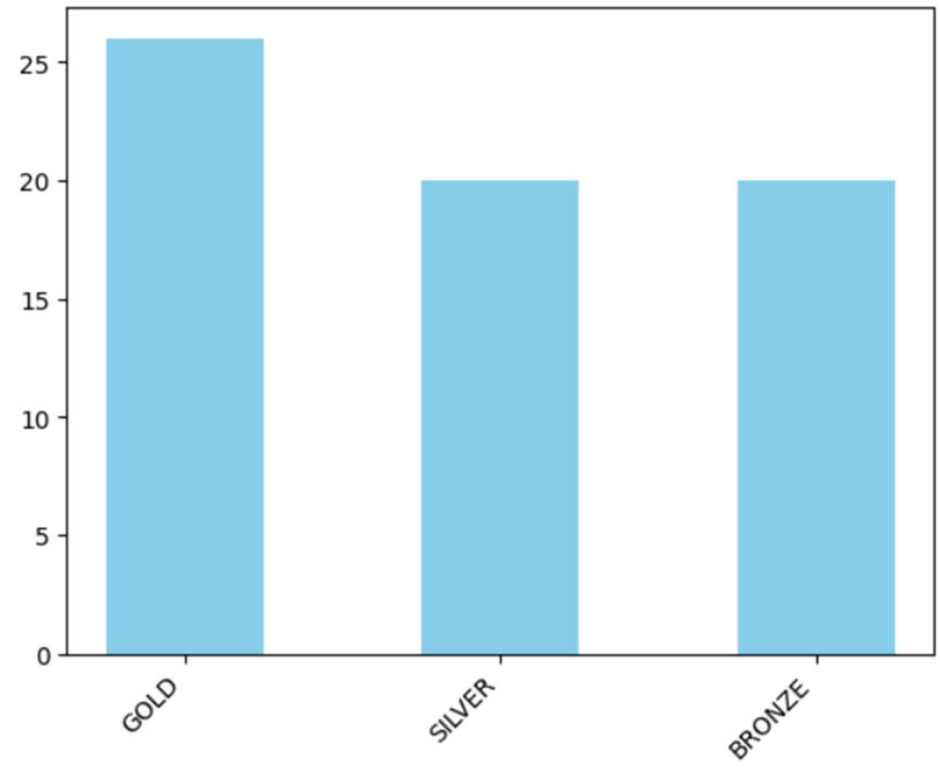
```
1  plt.bar(x,y,width=0.4, color="skyblue")
2  plt.grid(ls='--')
3  plt.show()
```



e. change the alignment of the labels.

```
[6]  1  plt.bar(x,y,width=0.5, color="skyblue")
     2  plt.xticks(rotation=45, ha='right')
```

```
([0, 1, 2], [Text(0, 0, 'GOLD'), Text(1, 0, 'SILVER'), Text(2, 0, 'BRONZE')])
```

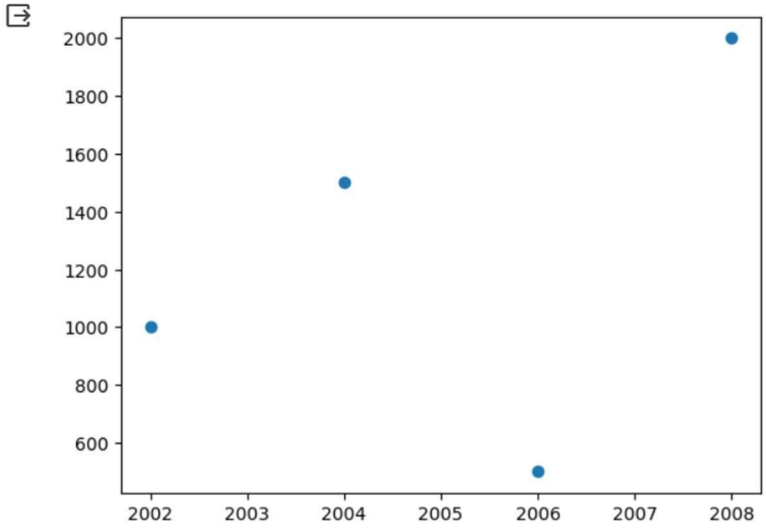


QUESTION 6) Draw a scatter plot that shows the stock trend for 5 years for TATA and Reliance company. Use the following properties:-

SOLUTION :-

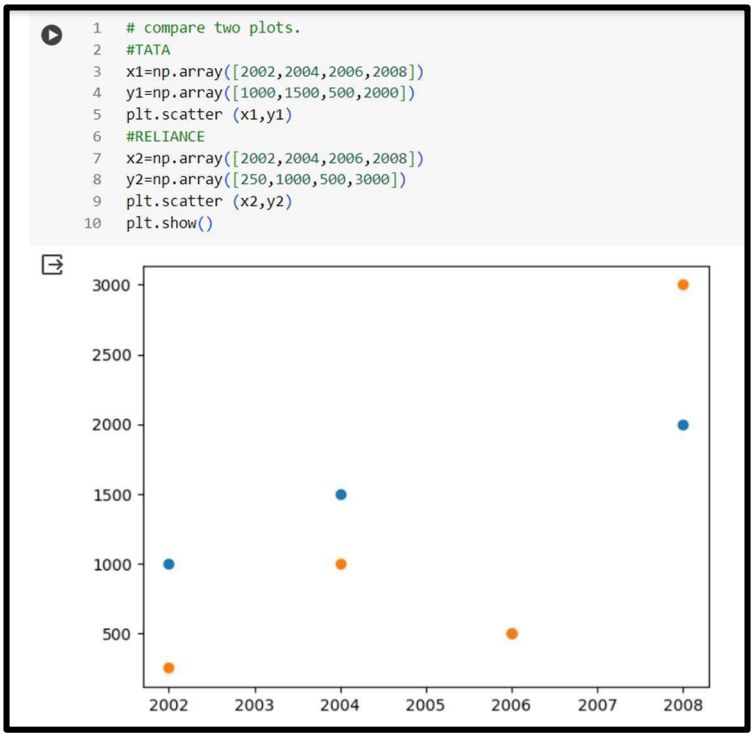
a. Draw the basic scatter graph using scatter()

```
# basic scatter graph
x=np.array([2002,2004,2006,2008])
y=np.array([1000,1500,500,2000])
plt.scatter (x,y)
plt.show()
```



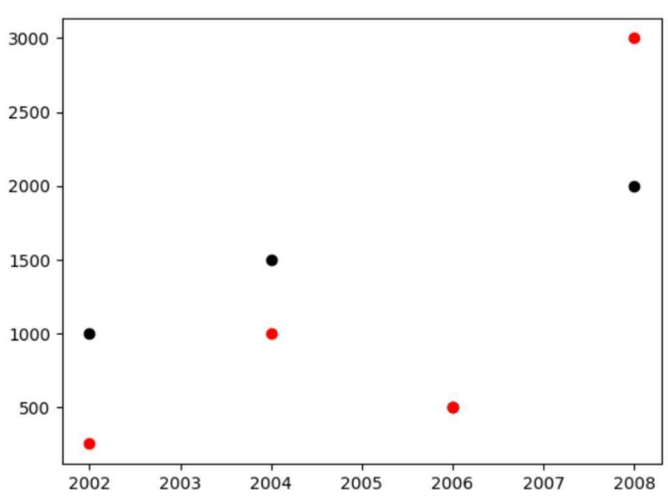
b. Compare two plots in a single plot.

```
# compare two plots.
#TATA
x1=np.array([2002,2004,2006,2008])
y1=np.array([1000,1500,500,2000])
plt.scatter (x1,y1)
#RELIANCE
x2=np.array([2002,2004,2006,2008])
y2=np.array([250,1000,500,3000])
plt.scatter (x2,y2)
plt.show()
```



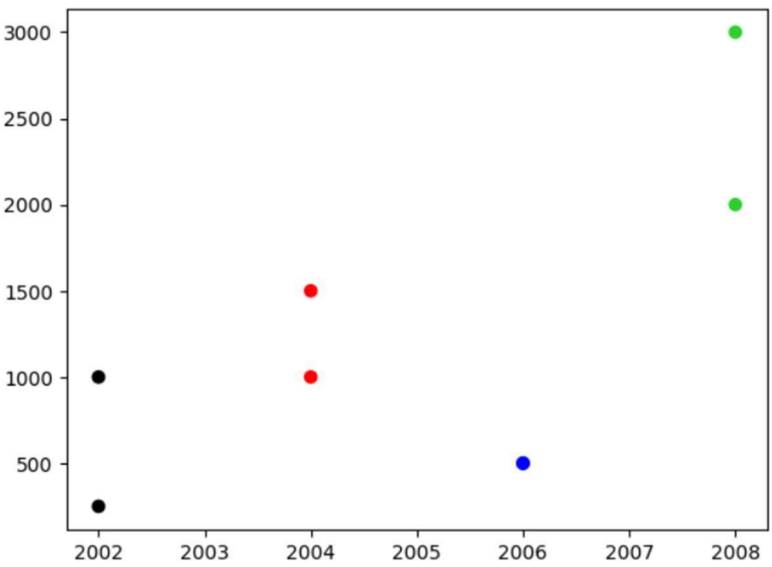
c. Set the color of both plots.

```
# set the color of dots in scatter graph
plt.scatter(x1,y1,color='black')
plt.scatter(x2,y2,color='red')
plt.show()
```



d. Set the different color of every dot.

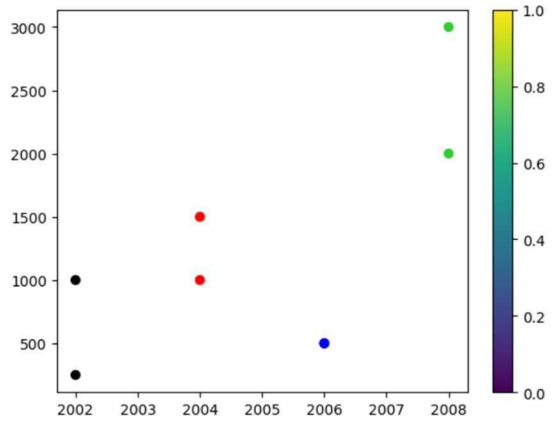
```
# assign different colors to every dot
mycolors=np.array(["black","red","blue","limegreen"])
plt.scatter(x1,y1,color=mycolors)
plt.scatter(x2,y2,color=mycolors)
plt.show()
```



e. Set colormap and display it using colorbar().

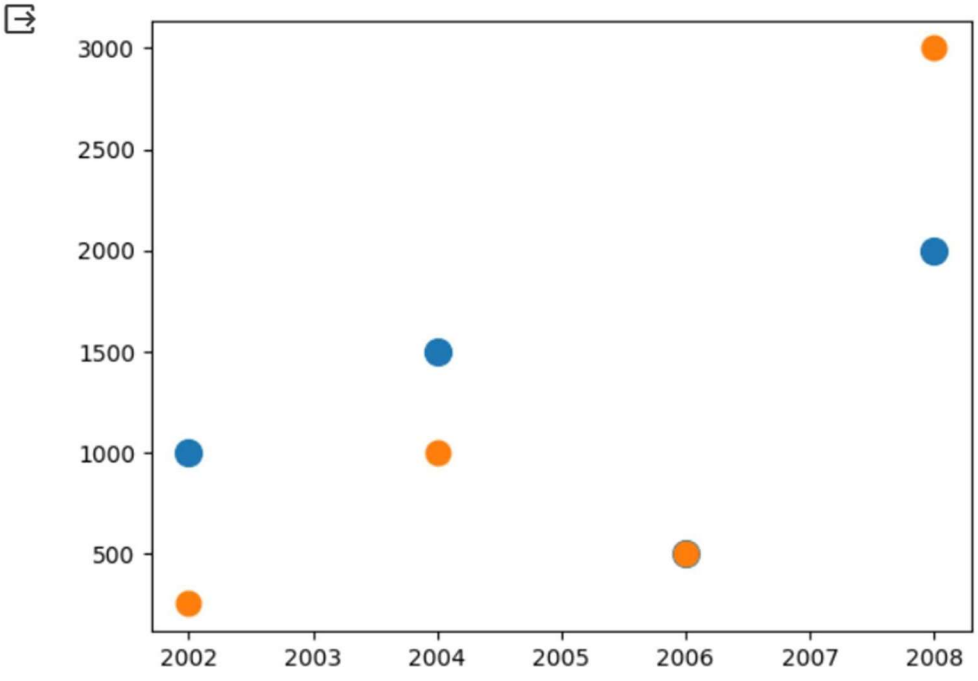
```
# color map
plt.scatter(x1,y1,color=mycolors,cmap='viridis')
plt.scatter(x2,y2,color=mycolors,cmap='viridis')
plt.colorbar()
plt.show()
```

```
<ipython-input-10-43e3a6bf464c>:2: UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored
  plt.scatter(x1,y1,color=mycolors,cmap='viridis')
<ipython-input-10-43e3a6bf464c>:3: UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will be ignored
  plt.scatter(x2,y2,color=mycolors,cmap='viridis')
```



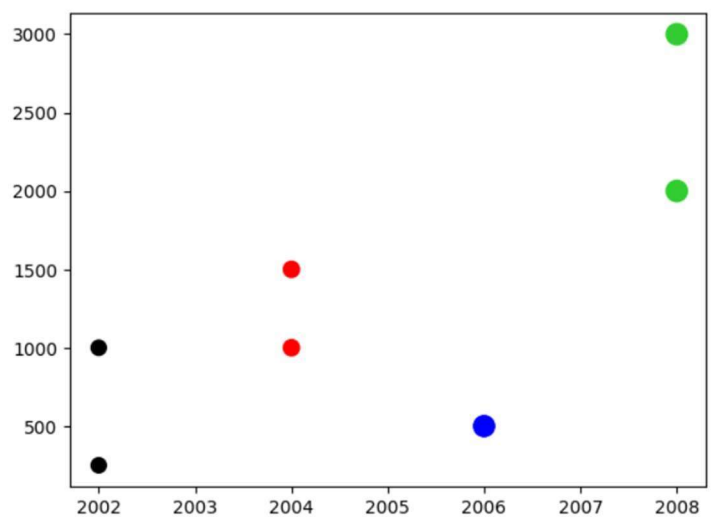
f. Set size of the dot.

```
# set the size of dot
plt.scatter(x1,y1,s=130)
plt.scatter(x2,y2,s=110)
plt.show()
```



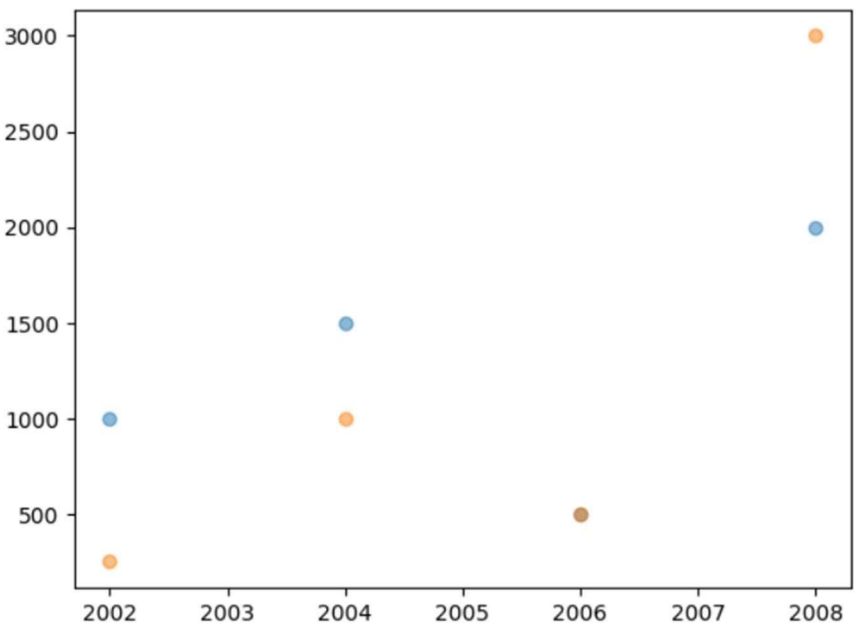
g. Set the different size of every dot.

```
# change size of every dot
mysizes=np.array([70,80,130,140])
mycolors=np.array(["black","red","blue","limegreen"])
plt.scatter (x1,y1,s=mysizes,color=mycolors)
plt.scatter (x2,y2,s=mysizes,color=mycolors)
plt.show()
```



h. Set the transparency of dot.

```
# change the transparency of the dot
plt.scatter(x1,y1,alpha=0.5)
plt.scatter(x2,y2,alpha=0.5)
plt.show()
```



QUESTION 7) Draw a pie chart that shows the sales of a different fruit in a day for a shop: 30% banana, 20% other, 15% orange, 10% guava, 25% mango Use the following properties:-

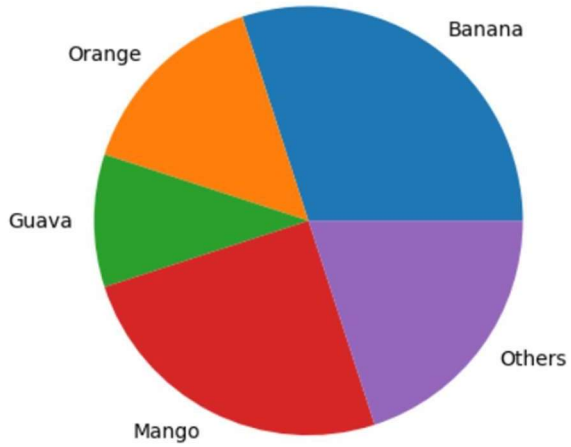
SOLUTION :-

a. Draw basic pie chart using pie().

```
y = np.array([30,15,10,25,20])
plt.pie(y)
plt.show()
```

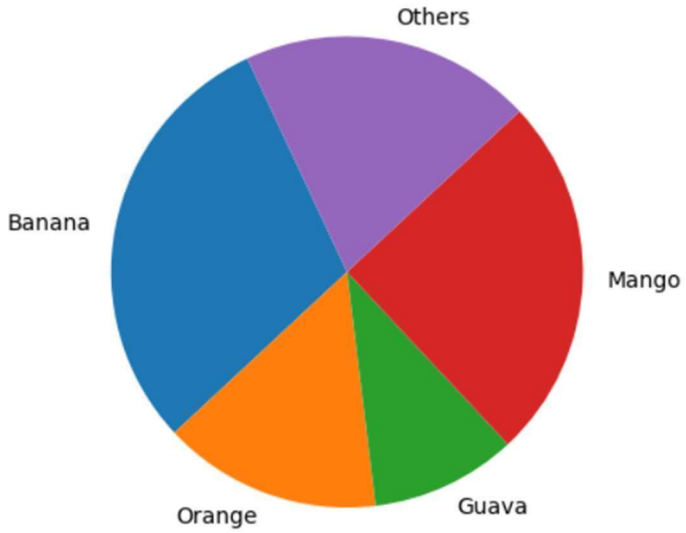
b. Add labels to wedges.

```
my_label = ["Banana", "Orange", "Guava", "Mango", "Others"]
plt.pie(y, labels=my_label)
plt.show()
```



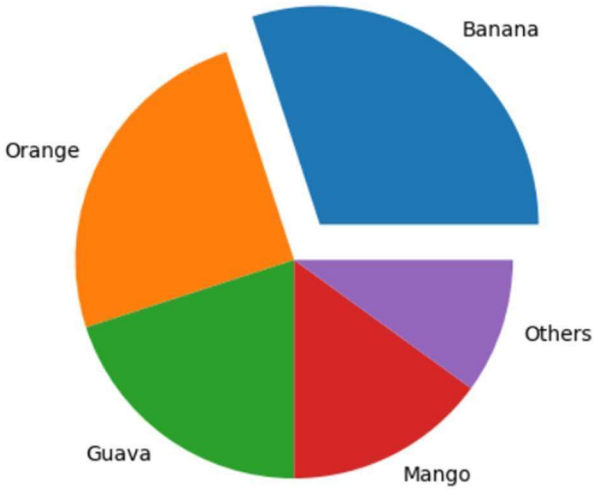
c. Change the start angle of any wedge.

```
my_label = ["Banana", "Orange", "Guava", "Mango", "Others"]
plt.pie(y, labels=my_label, startangle=115)
plt.show()
```



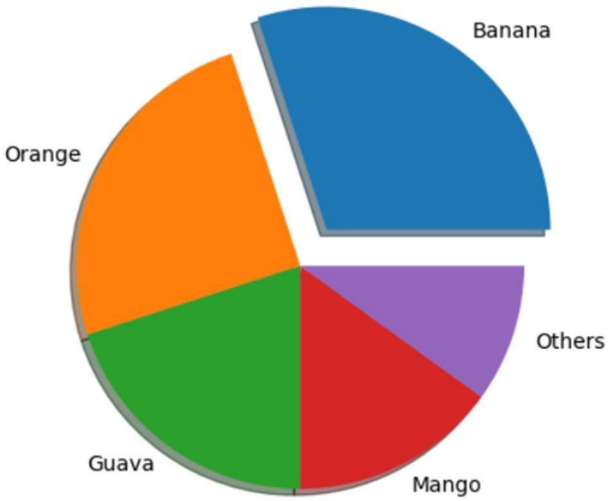
d. Displace the centre of wedge.

```
my_label = ["Banana", "Orange", "Guava", "Mango", "Others"]
y = [30, 25, 20, 15, 10]
my_explode = ([0.2, 0, 0, 0, 0])
plt.pie(y, labels=my_label, explode =my_explode)
plt.show()
```



e. Add shadow to slice wedges.

```
my_label = ["Banana", "Orange", "Guava", "Mango", "Others"]
y = [30, 25, 20, 15, 10]
my_explode =  ([0.2, 0, 0, 0, 0])
plt.pie(y, labels=my_label, explode =my_explode, shadow = True)
plt.show()
```



f. Change the colour of wedges.

```
my_label = ["Banana", "Orange", "Guava", "Mango", "Others"]
y = [30, 25, 20, 15, 10]
my_explode = ([0.2, 0.1 , 0.3 , 0.1, 0.2])
plt.pie(y, labels=my_label, explode =my_explode)
plt.show()
```