Name:B.Sumanth
Reg No:192124114
Course Code:DSA0503
Query Processing Partical Programs

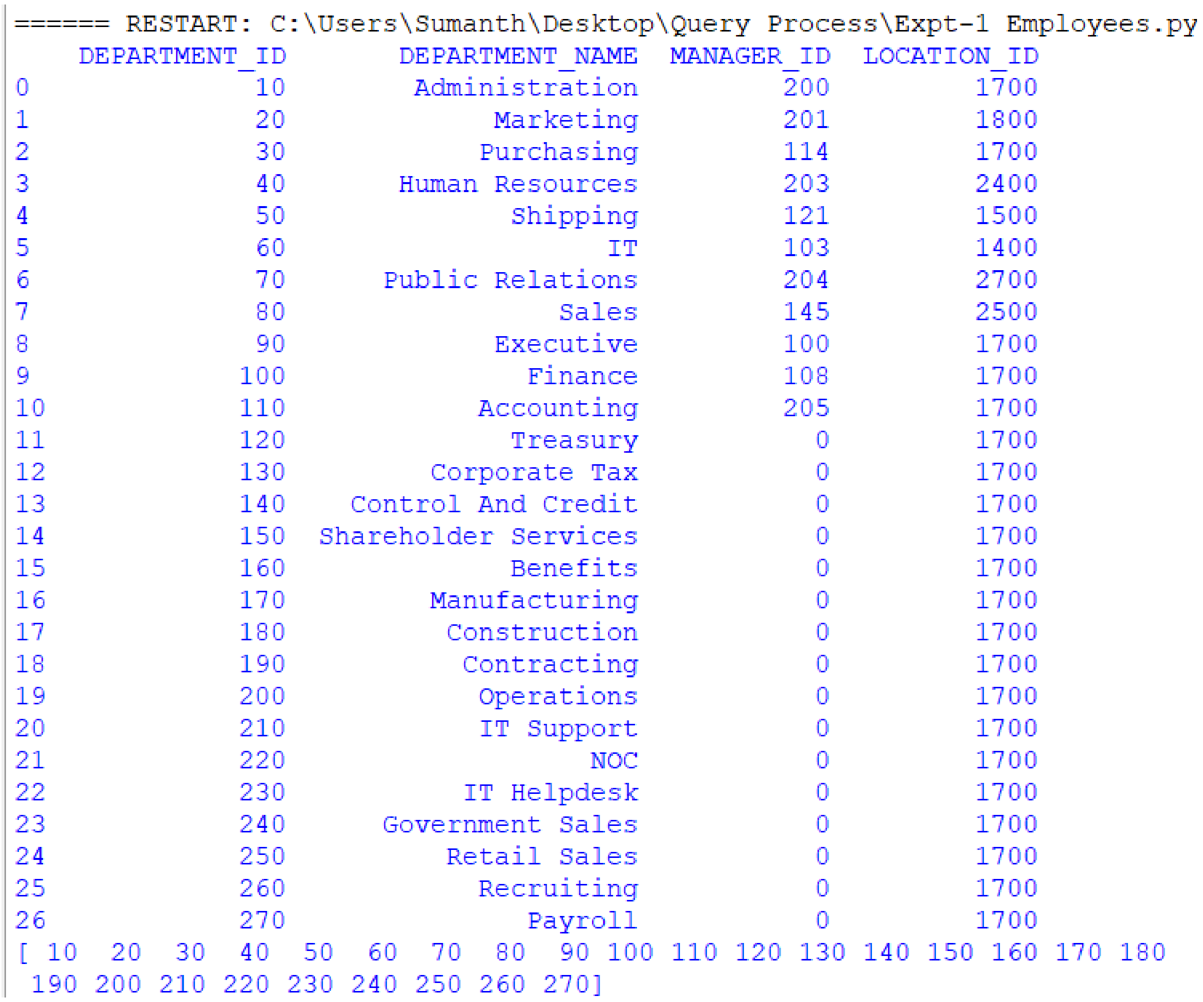
1. Write a Pandas program to select distinct department id from employees file.

```
+---------------+----------------------+------------+-------------+
| DEPARTMENT_ID | DEPARTMENT_NAME      | MANAGER_ID | LOCATION_ID |
+---------------+----------------------+------------+-------------+
|            10 | Administration       |        200 |        1700 |
|            20 | Marketing            |        201 |        1800 |
|            30 | Purchasing           |        114 |        1700 |
|            40 | Human Resources      |        203 |        2400 |
|            50 | Shipping             |        121 |        1500 |
|            60 | IT                   |        103 |        1400 |
|            70 | Public Relations     |        204 |        2700 |
|            80 | Sales                |        145 |        2500 |
|            90 | Executive            |        100 |        1700 |
|           100 | Finance              |        108 |        1700 |
|           110 | Accounting           |        205 |        1700 |
|           120 | Treasury             |          0 |        1700 |
|           130 | Corporate Tax        |          0 |        1700 |
|           140 | Control And Credit   |          0 |        1700 |
|           150 | Shareholder Services |          0 |        1700 |
|           160 | Benefits             |          0 |        1700 |
|           170 | Manufacturing        |          0 |        1700 |
|           180 | Construction         |          0 |        1700 |
|           190 | Contracting          |          0 |        1700 |
|           200 | Operations           |          0 |        1700 |
|           210 | IT Support           |          0 |        1700 |
|           220 | NOC                  |          0 |        1700 |
|           230 | IT Helpdesk          |          0 |        1700 |
|           240 | Government Sales     |          0 |        1700 |
|           250 | Retail Sales         |          0 |        1700 |
|           260 | Recruiting           |          0 |        1700 |
|           270 | Payroll              |          0 |        1700 |
+---------------+----------------------+------------+-------------
```

Program:

```
import pandas as pd
data = {
    'DEPARTMENT_ID': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100,
110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220,
230, 240, 250, 260, 270],
    'DEPARTMENT_NAME': ['Administration', 'Marketing',
'Purchasing', 'Human Resources', 'Shipping', 'IT', 'Public
Relations', 'Sales', 'Executive', 'Finance', 'Accounting',
'Treasury', 'Corporate Tax', 'Control And Credit',
'Shareholder Services', 'Benefits', 'Manufacturing',
'Construction', 'Contracting', 'Operations', 'IT Support',
'NOC', 'IT Helpdesk', 'Government Sales', 'Retail Sales',
'Recruiting', 'Payroll'],
    'MANAGER_ID': [200, 201, 114, 203, 121, 103, 204, 145,
100, 108, 205, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0],
    'LOCATION_ID': [1700, 1800, 1700, 2400, 1500, 1400, 2700,
2500, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700,
1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700, 1700]
}
employees_df = pd.DataFrame(data)
print(employees_df)
distinct_department_ids =
employees_df['DEPARTMENT_ID'].unique()
print(distinct_department_ids)
```

Output:



```
====== RESTART: C:\Users\Sumanth\Desktop\Query Process\Expt-1 Employees.py
    DEPARTMENT_ID       DEPARTMENT_NAME  MANAGER_ID  LOCATION_ID
0              10        Administration         200         1700
1              20             Marketing         201         1800
2              30            Purchasing         114         1700
3              40       Human Resources         203         2400
4              50              Shipping         121         1500
5              60                    IT         103         1400
6              70      Public Relations         204         2700
7              80                 Sales         145         2500
8              90             Executive         100         1700
9             100               Finance         108         1700
10            110            Accounting         205         1700
11            120              Treasury           0         1700
12            130         Corporate Tax           0         1700
13            140    Control And Credit           0         1700
14            150  Shareholder Services           0         1700
15            160              Benefits           0         1700
16            170         Manufacturing           0         1700
17            180          Construction           0         1700
18            190           Contracting           0         1700
19            200            Operations           0         1700
20            210            IT Support           0         1700
21            220                   NOC           0         1700
22            230           IT Helpdesk           0         1700
23            240      Government Sales           0         1700
24            250          Retail Sales           0         1700
25            260            Recruiting           0         1700
26            270               Payroll           0         1700
[ 10  20  30  40  50  60  70  80  90 100 110 120 130 140 150 160 170 180
 190 200 210 220 230 240 250 260 270]
```

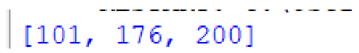
2. Write a Pandas program to display the ID for those employees who did two or more jobs in the past.

EMPLOYEE_ID	START_DATE	END_DATE	JOB_ID	DEPARTMENT_ID
102	2001-01-13	2006-07-24	IT_PROG	60
101	1997-09-21	2001-10-27	AC_ACCOUNT	110
101	2001-10-28	2005-03-15	AC_MGR	110
201	2004-02-17	2007-12-19	MK_REP	20
114	2006-03-24	2007-12-31	ST_CLERK	50
122	2007-01-01	2007-12-31	ST_CLERK	50
200	1995-09-17	2001-06-17	AD_ASST	90
176	2006-03-24	2006-12-31	SA_REP	80
176	2007-01-01	2007-12-31	SA_MAN	80
200	2002-07-01	2006-12-31	AC_ACCOUNT	90

Program:

```
import pandas as pd

data = {

    'EMPLOYEE_ID': [102, 101, 101, 201, 114, 122, 200, 176, 176, 200],

    'START_DATE': ['2001-01-13', '1997-09-21', '2001-10-28', '2004-02-17', '2006-03-24', '2007-01-01', '1995-09-17', '2006-03-24', '2007-01-01', '2002-07-01'],

    'END_DATE': ['2006-07-24', '2001-10-27', '2005-03-15', '2007-12-19', '2007-12-31', '2007-12-31', '2001-06-17', '2006-12-31', '2007-12-31', '2006-12-31'],

    'JOB_ID': ['IT_PROG', 'AC_ACCOUNT', 'AC_MGR', 'MK_REP', 'ST_CLERK', 'ST_CLERK', 'AD_ASST', 'SA_REP', 'SA_MAN', 'AC_ACCOUNT'],

    'DEPARTMENT_ID': [60, 110, 110, 20, 50, 50, 90, 80, 80, 90]

}

df = pd.DataFrame(data)

df['START_DATE'] = pd.to_datetime(df['START_DATE'])

df['END_DATE'] = pd.to_datetime(df['END_DATE'])

employee_jobs_count = df.groupby('EMPLOYEE_ID')['JOB_ID'].nunique()

employees_with_multiple_jobs = employee_jobs_count[employee_jobs_count >= 2]

print(employees_with_multiple_jobs.index.tolist())
```

Output:



```
[101, 176, 200]
```

3. Write a Pandas program to display the details of jobs in descending sequence on job title.

```
+------------+---------------------------------+------------+------------+
| JOB_ID     | JOB_TITLE                       | MIN_SALARY | MAX_SALARY |
+------------+---------------------------------+------------+------------+
| AD_PRES    | President                       |      20080 |      40000 |
| AD_VP      | Administration Vice President   |      15000 |      30000 |
| AD_ASST    | Administration Assistant        |       3000 |       6000 |
| FI_MGR     | Finance Manager                 |       8200 |      16000 |
| FI_ACCOUNT | Accountant                      |       4200 |       9000 |
| AC_MGR     | Accounting Manager              |       8200 |      16000 |
| AC_ACCOUNT | Public Accountant               |       4200 |       9000 |
| SA_MAN     | Sales Manager                   |      10000 |      20080 |
| SA_REP     | Sales Representative            |       6000 |      12008 |
| PU_MAN     | Purchasing Manager              |       8000 |      15000 |
| PU_CLERK   | Purchasing Clerk                |       2500 |       5500 |
| ST_MAN     | Stock Manager                   |       5500 |       8500 |
| ST_CLERK   | Stock Clerk                     |       2008 |       5000 |
| SH_CLERK   | Shipping Clerk                  |       2500 |       5500 |
| IT_PROG    | Programmer                      |       4000 |      10000 |
| MK_MAN     | Marketing Manager               |       9000 |      15000 |
| MK_REP     | Marketing Representative        |       4000 |       9000 |
| HR_REP     | Human Resources Representative  |       4000 |       9000 |
| PR_REP     | Public Relations Representative |       4500 |      10500 |
+------------+---------------------------------+------------+------------+
```

Program:

```
import pandas as pd

data = {

    'JOB_ID': ['AD_PRES', 'AD_VP', 'AD_ASST', 'FI_MGR', 'FI_ACCOUNT', 'AC_MGR', 'AC_ACCOUNT', 'SA_MAN', 'SA_REP', 'PU_MAN', 'PU_CLERK', 'ST_MAN', 'ST_CLERK', 'SH_CLERK', 'IT_PROG', 'MK_MAN', 'MK_REP', 'HR_REP', 'PR_REP'],

    'JOB_TITLE': ['President', 'Administration Vice President', 'Administration Assistant', 'Finance Manager', 'Accountant', 'Accounting Manager', 'Public Accountant', 'Sales Manager', 'Sales Representative', 'Purchasing Manager', 'Purchasing Clerk', 'Stock Manager', 'Stock Clerk', 'Shipping Clerk', 'Programmer', 'Marketing Manager', 'Marketing Representative', 'Human Resources Representative', 'Public Relations Representative'],

    'MIN_SALARY': [20080, 15000, 3000, 8200, 4200, 8200, 4200, 10000, 6000, 8000, 2500, 5500, 2008, 2500, 4000, 9000, 4000, 4000, 4500],

    'MAX_SALARY': [40000, 30000, 6000, 16000, 9000, 16000, 9000, 20080, 12008, 15000, 5500, 8500, 5000, 5500, 10000, 15000, 9000, 9000, 10500]

}

df = pd.DataFrame(data)
```

print("original_data")

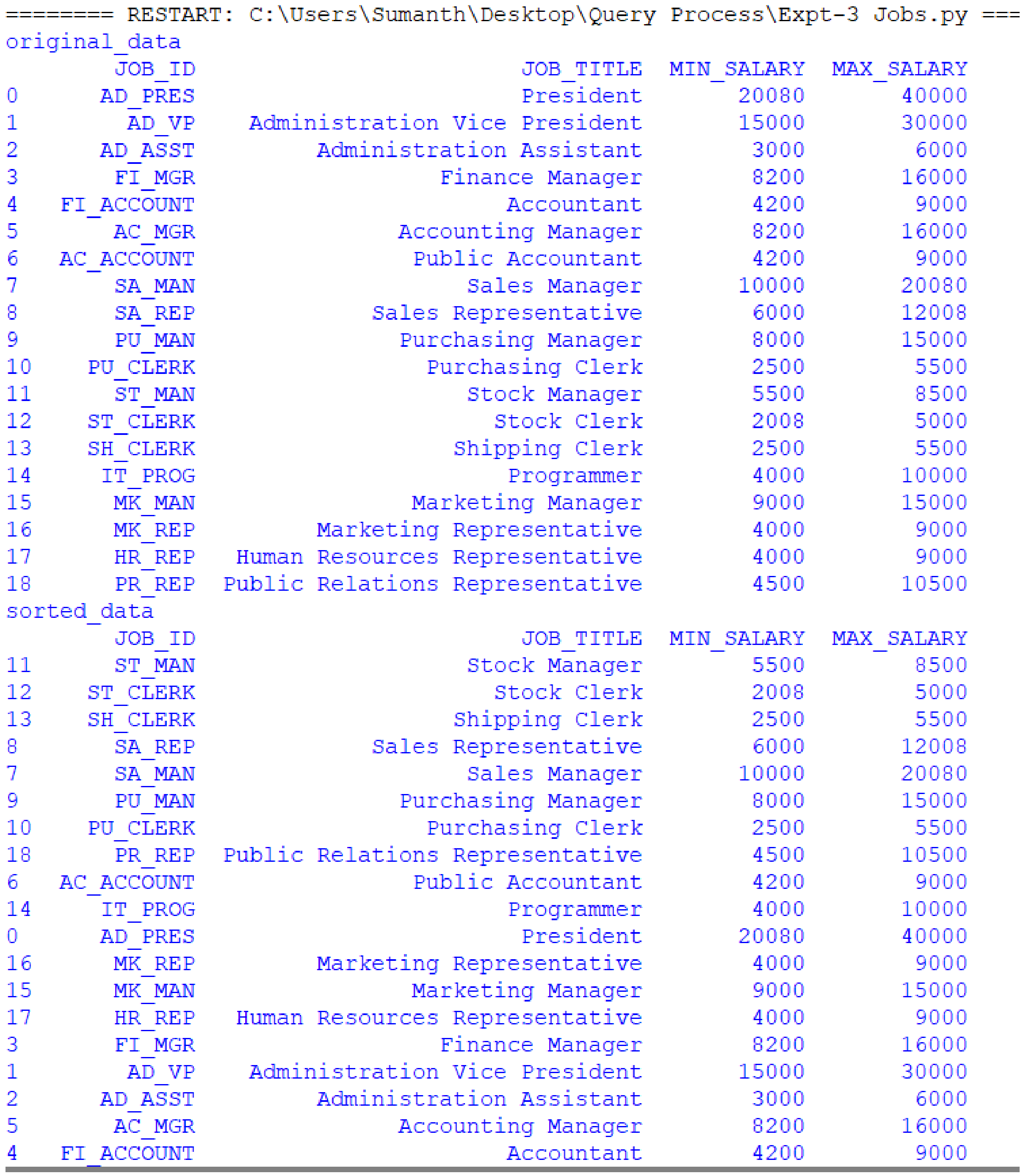
print(df)

df_sorted = df.sort_values(by='JOB_TITLE', ascending=False)

print("sorted_data")

print(df_sorted)

Output:



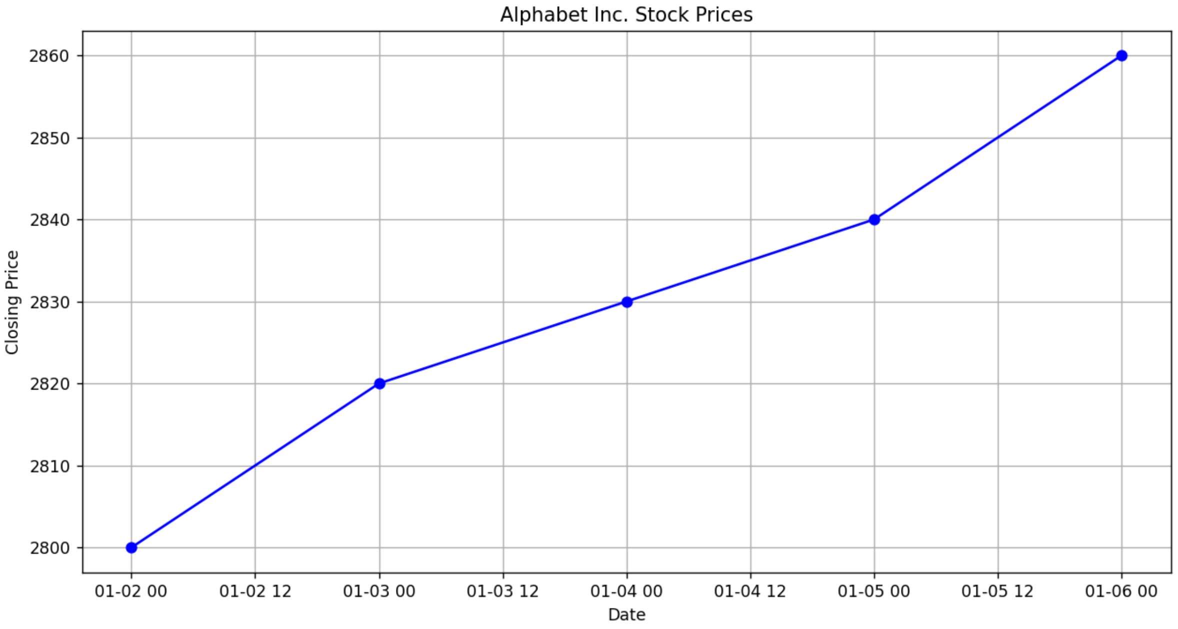
```
======== RESTART: C:\Users\Sumanth\Desktop\Query Process\Expt-3 Jobs.py ===
original_data
        JOB_ID                        JOB_TITLE  MIN_SALARY  MAX_SALARY
0      AD_PRES                        President       20080       40000
1        AD_VP    Administration Vice President       15000       30000
2      AD_ASST         Administration Assistant        3000        6000
3       FI_MGR                  Finance Manager        8200       16000
4   FI_ACCOUNT                       Accountant        4200        9000
5       AC_MGR               Accounting Manager        8200       16000
6   AC_ACCOUNT                Public Accountant        4200        9000
7       SA_MAN                    Sales Manager       10000       20080
8       SA_REP             Sales Representative        6000       12008
9       PU_MAN               Purchasing Manager        8000       15000
10    PU_CLERK                 Purchasing Clerk        2500        5500
11      ST_MAN                    Stock Manager        5500        8500
12    ST_CLERK                      Stock Clerk        2008        5000
13    SH_CLERK                   Shipping Clerk        2500        5500
14     IT_PROG                       Programmer        4000       10000
15      MK_MAN                Marketing Manager        9000       15000
16      MK_REP         Marketing Representative        4000        9000
17      HR_REP   Human Resources Representative        4000        9000
18      PR_REP  Public Relations Representative        4500       10500
sorted_data
        JOB_ID                        JOB_TITLE  MIN_SALARY  MAX_SALARY
11      ST_MAN                    Stock Manager        5500        8500
12    ST_CLERK                      Stock Clerk        2008        5000
13    SH_CLERK                   Shipping Clerk        2500        5500
8       SA_REP             Sales Representative        6000       12008
7       SA_MAN                    Sales Manager       10000       20080
9       PU_MAN               Purchasing Manager        8000       15000
10    PU_CLERK                 Purchasing Clerk        2500        5500
18      PR_REP  Public Relations Representative        4500       10500
6   AC_ACCOUNT                Public Accountant        4200        9000
14     IT_PROG                       Programmer        4000       10000
0      AD_PRES                        President       20080       40000
16      MK_REP         Marketing Representative        4000        9000
15      MK_MAN                Marketing Manager        9000       15000
17      HR_REP   Human Resources Representative        4000        9000
3       FI_MGR                  Finance Manager        8200       16000
1        AD_VP    Administration Vice President       15000       30000
2      AD_ASST         Administration Assistant        3000        6000
5       AC_MGR               Accounting Manager        8200       16000
4   FI_ACCOUNT                       Accountant        4200        9000
```

4. Write a Pandas program to create a line plot of the historical stock prices of Alphabet Inc. between two specific dates.

Program:

```
import pandas as pd

import matplotlib.pyplot as plt


data = {

    'Date': ['2023-01-02', '2023-01-03', '2023-01-04', '2023-01-05', '2023-01-06'],

    'Close_Price': [2800.00, 2820.00, 2830.00, 2840.00, 2860.00]

}

df = pd.DataFrame(data)

df['Date'] = pd.to_datetime(df['Date'])

df.set_index(‘Date', inplace=True)

start_date = '2023-01-02'

end_date = '2023-01-06'

filtered_data = df[start_date:end_date]

plt.figure(figsize=(12, 6))

plt.plot(filtered_data.index, filtered_data['Close_Price'], marker='o', linestyle='-', color='b')

plt.title(‘Alphabet Inc. Stock Prices')

plt.xlabel(‘Date')

plt.ylabel(‘Closing Price')

plt.grid(True)

plt.show()
```

Output:

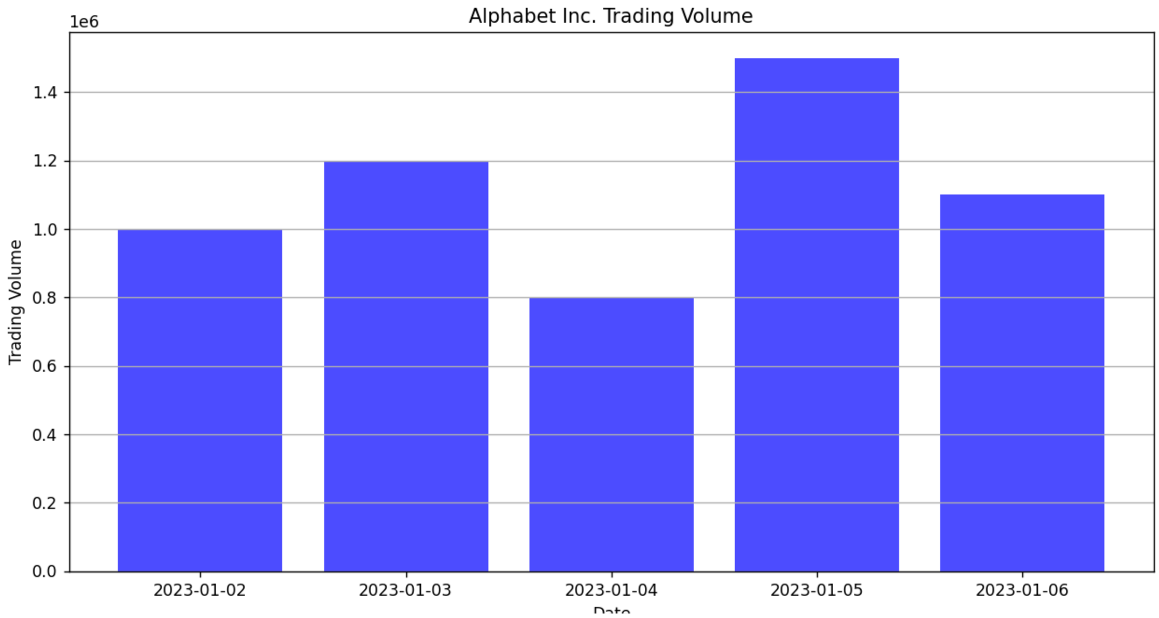


5. Write a Pandas program to create a bar plot of the trading volume of Alphabet Inc. stock between two specific dates.

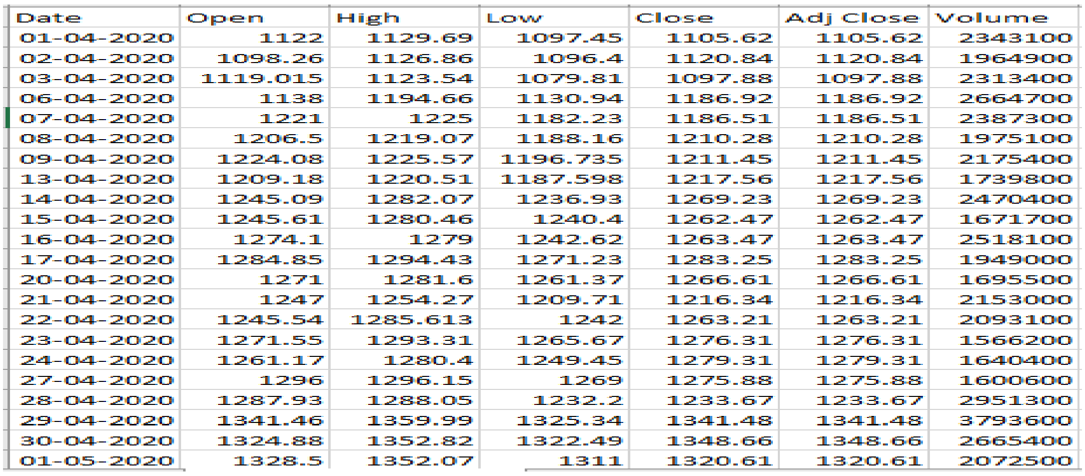
Program:

```
import pandas as pd
import matplotlib.pyplot as plt
data = {
    'Date': ['2023-01-02', '2023-01-03', '2023-01-04', '2023-01-05', '2023-01-06'],
    'Trading_Volume': [1000000, 1200000, 800000, 1500000, 1100000]
}
df = pd.DataFrame(data)
df['Date'] = pd.to_datetime(df['Date'])
df.set_index('Date', inplace=True)
start_date = '2023-01-02'
end_date = '2023-01-06'
filtered_data = df[start_date:end_date]
plt.figure(figsize=(12, 6))
plt.bar(filtered_data.index, filtered_data['Trading_Volume'], color='b', alpha=0.7)
plt.title('Alphabet Inc. Trading Volume')
plt.xlabel('Date')
plt.ylabel('Trading Volume')
plt.grid(axis='y')
plt.show()
```

Output:



6. Write a Pandas program to create a scatter plot of the trading volume/stock prices of Alphabet Inc. stock between two specific dates. **alphabet_stock_data:**

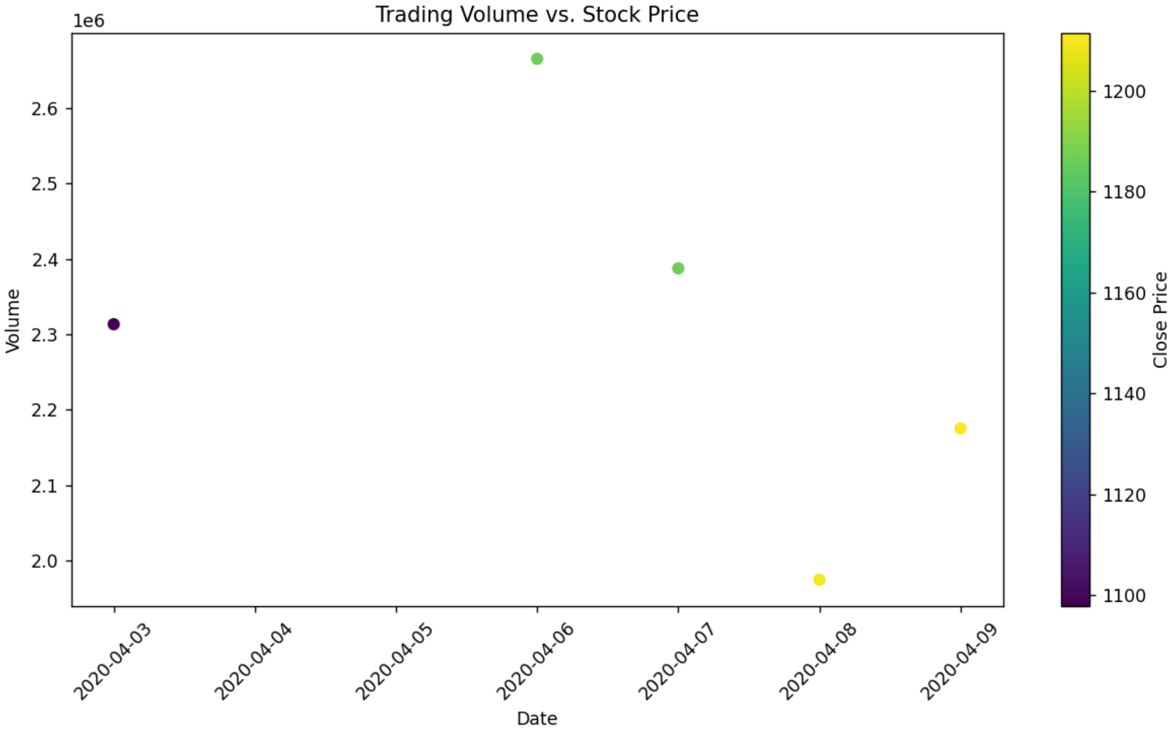


Date	Open	High	Low	Close	Adj Close	Volume
01-04-2020	1122	1129.69	1097.45	1105.62	1105.62	2343100
02-04-2020	1098.26	1126.86	1096.4	1120.84	1120.84	1964900
03-04-2020	1119.015	1123.54	1079.81	1097.88	1097.88	2313400
06-04-2020	1138	1194.66	1130.94	1186.92	1186.92	2664700
07-04-2020	1221	1225	1182.23	1186.51	1186.51	2387300
08-04-2020	1206.5	1219.07	1188.16	1210.28	1210.28	1975100
09-04-2020	1224.08	1225.57	1196.735	1211.45	1211.45	2175400
13-04-2020	1209.18	1220.51	1187.598	1217.56	1217.56	1739800
14-04-2020	1245.09	1282.07	1236.93	1269.23	1269.23	2470400
15-04-2020	1245.61	1280.46	1240.4	1262.47	1262.47	1671700
16-04-2020	1274.1	1279	1242.62	1263.47	1263.47	2518100
17-04-2020	1284.85	1294.43	1271.23	1283.25	1283.25	1949000
20-04-2020	1271	1281.6	1261.37	1266.61	1266.61	1695500
21-04-2020	1247	1254.27	1209.71	1216.34	1216.34	2153000
22-04-2020	1245.54	1285.613	1242	1263.21	1263.21	2093100
23-04-2020	1271.55	1293.31	1265.67	1276.31	1276.31	1566200
24-04-2020	1261.17	1280.4	1249.45	1279.31	1279.31	1640400
27-04-2020	1296	1296.15	1269	1275.88	1275.88	1600600
28-04-2020	1287.93	1288.05	1232.2	1233.67	1233.67	2951300
29-04-2020	1341.46	1359.99	1325.34	1341.48	1341.48	3793600
30-04-2020	1324.88	1352.82	1322.49	1348.66	1348.66	2665400
01-05-2020	1328.5	1352.07	1311	1320.61	1320.61	2072500

Program:

```
import pandas as pd

import matplotlib.pyplot as plt

data = {

    'Date': ['01-04-2020', '02-04-2020', '03-04-2020', '06-04-2020', '07-04-2020',
'08-04-2020', '09-04-2020',

             '13-04-2020', '14-04-2020', '15-04-2020', '16-04-2020', '17-04-2020', '20-
04-2020', '21-04-2020',
        '22-04-2020', '23-04-2020', '24-04-2020', '27-04-2020', '28-04-2020', '29-04-2020', '30-04-2020',

        '01-05-2020'],

   'Open': [1122, 1098.26, 1119.015, 1138, 1221, 1206.5, 1224.08, 1209.18, 1245.09, 1245.61, 1274.1, 1284.85, 1271, 1247, 1245.54, 1271.55, 1261.17, 1296, 1287.93, 1341.46, 1324.88, 1328.5],

   'High': [1129.69, 1126.86, 1123.54, 1194.66, 1225, 1219.07, 1225.57, 1220.51, 1282.07, 1280.46, 1279, 1294.43, 1281.6, 1254.27, 1285.613, 1293.31, 1280.4, 1296.15, 1288.05, 1359.99, 1352.82, 1352.07],

   'Low': [1097.45, 1096.4, 1079.81, 1130.94, 1182.23, 1188.16, 1196.735, 1187.598, 1236.93, 1240.4, 1242.62, 1271.23, 1261.37, 1209.71, 1242, 1265.67, 1249.45, 1269, 1232.2, 1325.34, 1322.49, 1311],

   'Close': [1105.62, 1120.84, 1097.88, 1186.92, 1186.51, 1210.28, 1211.45, 1217.56, 1269.23, 1262.47, 1263.47, 1283.25, 1266.61, 1216.34, 1263.21, 1276.31, 1279.31, 1275.88, 1233.67, 1341.48, 1348.66, 1320.61],

   'Adj Close': [1105.62, 1120.84, 1097.88, 1186.92, 1186.51, 1210.28, 1211.45, 1217.56, 1269.23, 1262.47, 1263.47, 1283.25, 1266.61, 1216.34, 1263.21, 1276.31, 1279.31, 1275.88, 1233.67, 1341.48, 1348.66, 1320.61],

   'Volume': [2343100, 1964900, 2313400, 2664700, 2387300, 1975100, 2175400, 1739800, 2470400, 1671700, 2518100, 1949000, 1695500, 2153000, 2093100, 1566200, 1640400, 1600600, 2951300, 3793600, 2665400, 2072500]

}

data['Date'] = pd.to_datetime(data['Date'], format='%d-%m-%Y')

df = pd.DataFrame(data)

start_date = '2020-04-03'

end_date = '2020-04-10'

filtered_data = df[(df['Date'] >= start_date) & (df['Date'] <= end_date)]

plt.figure(figsize=(10, 6))

plt.scatter(filtered_data['Date'], filtered_data['Volume'], c=filtered_data['Close'], cmap='viridis', marker='o')

plt.title('Trading Volume vs. Stock Price')

plt.xlabel('Date')

plt.ylabel('Volume')

plt.colorbar(label='Close Price')

plt.xticks(rotation=45)
plt.tight_layout()

plt.show()
```

Output:

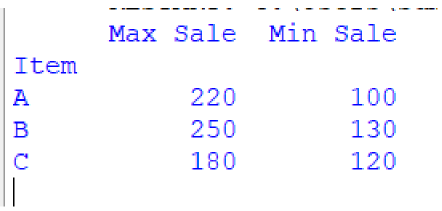


7. Write a Pandas program to create a Pivot table and find the maximum and minimum sale value of the items.(refer sales_data table)

   Program:

```
import pandas as pd
data = {
   'Item': ['A', 'B', 'A', 'C', 'B', 'C', 'A', 'B', 'C'],
   'Sale': [100, 150, 200, 120, 250, 180, 220, 130, 160]
}
sales_data = pd.DataFrame(data)
pivot_table = sales_data.pivot_table(index='Item', values='Sale',
aggfunc={'Sale': ['max', 'min']})
pivot_table.columns = ['Max Sale', 'Min Sale']
print(pivot_table)
```

   Output:



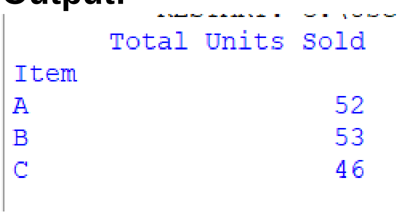
```
      Max Sale  Min Sale
Item
A          220       100
B          250       130
C          180       120
```

8. Write a Pandas program to create a Pivot table and find the item wise unit sold. .(refer sales_data table)

   Program:

```
import pandas as pd
data = {
    'Item': ['A', 'B', 'A', 'C', 'B', 'C', 'A', 'B', 'C'],
    'Units Sold': [10, 15, 20, 12, 25, 18, 22, 13, 16]
}
sales_data = pd.DataFrame(data)
pivot_table = sales_data.pivot_table(index='Item', values='Units Sold',
aggfunc='sum')
pivot_table.columns = ['Total Units Sold']
print(pivot_table)
```

   Output:



```
      Total Units Sold
Item
A                   52
B                   53
C                   46
```

9. Write a Pandas program to create a Pivot table and find the total sale amount region wise, manager wise, sales man wise. .(refer sales_data table)

Sales_data:

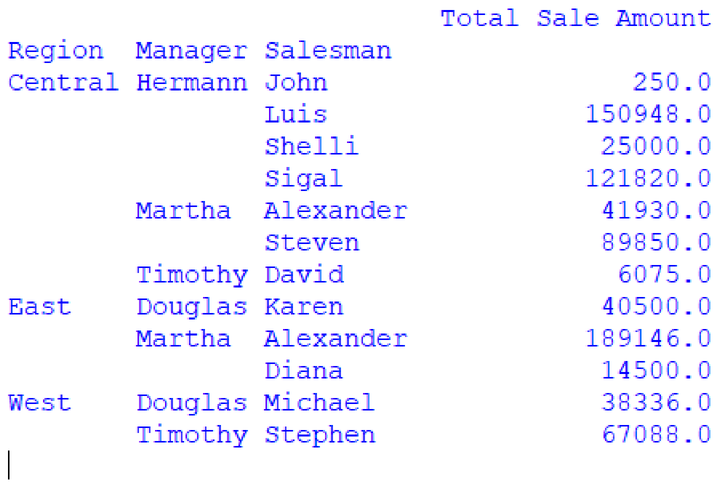
OrderDate	Region	Manager	SalesMan	Item	Units	Unit_price	Sale_amt
1-6-18	East	Martha	Alexander	Television	95	1,198.00	1,13,810.00
1-23-18	Central	Hermann	Shelli	Home Theater	50	500.00	25,000.00
2-9-18	Central	Hermann	Luis	Television	36	1,198.00	43,128.00
2-26-18	Central	Timothy	David	Cell Phone	27	225.00	6,075.00
3-15-18	West	Timothy	Stephen	Television	56	1,198.00	67,088.00

4-1-18	East	Martha	Alexander	Home Theater	60	500.00	30,000.00
4-18-18	Central	Martha	Steven	Television	75	1,198.00	89,850.00
5-5-18	Central	Hermann	Luis	Television	90	1,198.00	1,07,820.00
5-22-18	West	Douglas	Michael	Television	32	1,198.00	38,336.00
6-8-18	East	Martha	Alexander	Home Theater	60	500.00	30,000.00
6-25-18	Central	Hermann	Sigal	Television	90	1,198.00	1,07,820.00
7-12-18	East	Martha	Diana	Home Theater	29	500.00	14,500.00
7-29-18	East	Douglas	Karen	Home Theater	81	500.00	40,500.00
8-15-18	East	Martha	Alexander	Television	35	1,198.00	41,930.00
9-1-18	Central	Douglas	John	Desk	2	125.00	250.00
9-18-18	East	Martha	Alexander	Video Games	16	58.50	936.00
10-5-18	Central	Hermann	Sigal	Home Theater	28	500.00	14,000.00
10-22-18	East	Martha	Alexander	Cell Phone	64	225.00	14,400.00

Program:

```
import pandas as pd

data = {

    'OrderDate': ['1-6-18', '1-23-18', '2-9-18', '2-26-18', '3-15-18', '4-1-18', '4-18-18', '5-5-18', '5-22-18', '6-8-18', '6-25-18', '7-12-18', '7-29-18', '8-15-18', '9-1-18', '9-18-18', '10-5-18', '10-22-18'],

    'Region': ['East', 'Central', 'Central', 'Central', 'West', 'East', 'Central', 'Central', 'West', 'East', 'Central', 'East', 'East', 'Central', 'Central', 'East', 'Central', 'East'],

    'Manager': ['Martha', 'Hermann', 'Hermann', 'Timothy', 'Timothy', 'Martha', 'Martha', 'Hermann', 'Douglas', 'Martha', 'Hermann', 'Martha', 'Douglas', 'Martha', 'Hermann', 'Martha', 'Hermann', 'Martha'],

    'Salesman': ['Alexander', 'Shelli', 'Luis', 'David', 'Stephen', 'Alexander', 'Steven', 'Luis', 'Michael', 'Alexander', 'Sigal', 'Diana', 'Karen', 'Alexander', 'John', 'Alexander', 'Sigal', 'Alexander'],

    'Item': ['Television', 'Home Theater', 'Television', 'Cell Phone', 'Television', 'Home Theater', 'Television', 'Television', 'Television', 'Home Theater', 'Television', 'Home Theater', 'Home Theater', 'Television', 'Desk', 'Video Games', 'Home Theater', 'Cell Phone'],
    'Units': [95, 50, 36, 27, 56, 60, 75, 90, 32, 60, 90, 29, 81, 35, 2, 16, 28, 64],
    'Unit_price': [1198.00, 500.00, 1198.00, 225.00, 1198.00, 500.00, 1198.00, 1198.00, 1198.00, 500.00, 1198.00, 500.00, 500.00, 1198.00, 125.00, 58.50, 500.00, 225.00],
    'Sale_amt': [113810.00, 25000.00, 43128.00, 6075.00, 67088.00, 30000.00, 89850.00, 107820.00, 38336.00, 30000.00, 107820.00, 14500.00, 40500.00, 41930.00, 250.00, 936.00, 14000.00, 14400.00]
}
sales_data = pd.DataFrame(data)
pivot_table = pd.pivot_table(sales_data,
                        values='Sale_amt',
                        index=['Region', 'Manager', 'Salesman'],
                        aggfunc='sum')
pivot_table.columns = ['Total Sale Amount']
print(pivot_table)
```

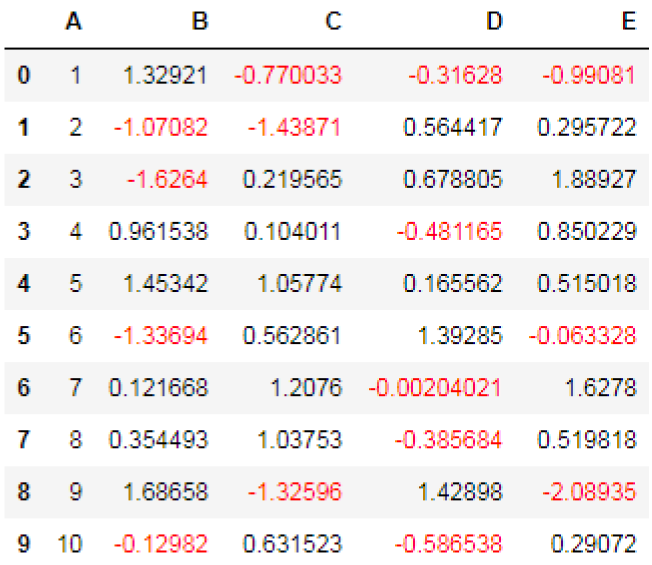
Output:



```
                                Total Sale Amount
Region  Manager Salesman
Central Hermann John                        250.0
                Luis                     150948.0
                Shelli                    25000.0
                Sigal                    121820.0
        Martha  Alexander                 41930.0
                Steven                    89850.0
        Timothy David                      6075.0
East    Douglas Karen                     40500.0
        Martha  Alexander                189146.0
                Diana                     14500.0
West    Douglas Michael                   38336.0
        Timothy Stephen                   67088.0
```

10.Create a dataframe of ten rows, four columns with random values. Write a Pandas program to highlight the negative numbers red and positive numbers black.

Expected Output:

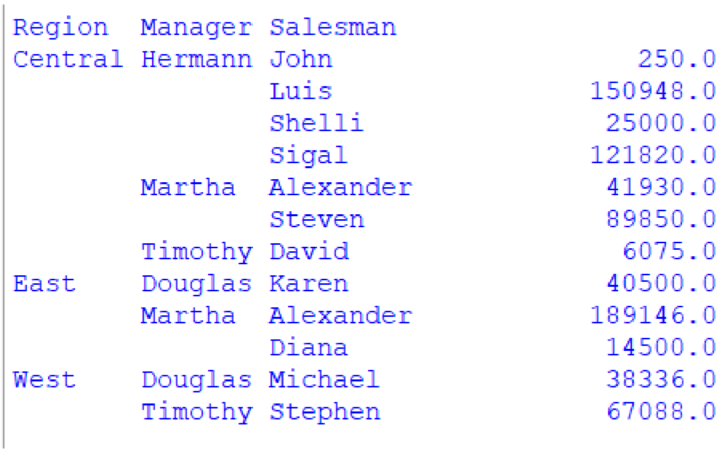


	A	B	C	D	E
0	1	1.32921	-0.770033	-0.31628	-0.99081
1	2	-1.07082	-1.43871	0.564417	0.295722
2	3	-1.6264	0.219565	0.678805	1.88927
3	4	0.961538	0.104011	-0.481165	0.850229
4	5	1.45342	1.05774	0.165562	0.515018
5	6	-1.33694	0.562861	1.39285	-0.063328
6	7	0.121668	1.2076	-0.00204021	1.6278
7	8	0.354493	1.03753	-0.385684	0.519818
8	9	1.68658	-1.32596	1.42898	-2.08935
9	10	-0.12982	0.631523	-0.586538	0.29072

Program:

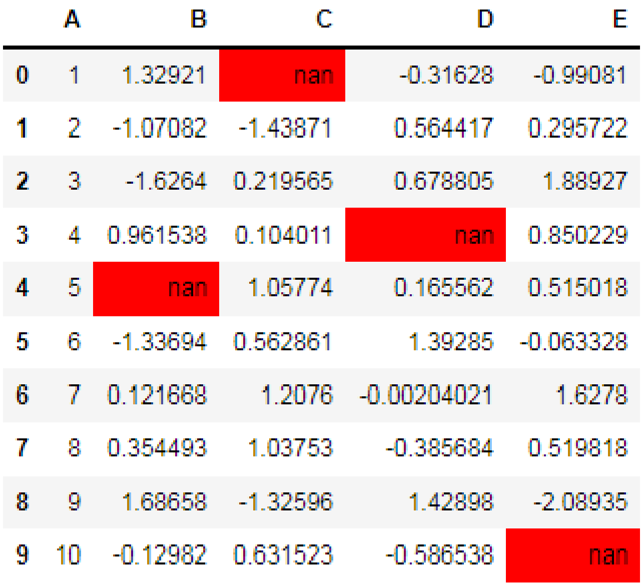
```
import pandas as pd
import numpy as np
data = {
    'Column1': np.random.uniform(-1, 1, 10),
    'Column2': np.random.uniform(-1, 1, 10),
    'Column3': np.random.uniform(-1, 1, 10),
    'Column4': np.random.uniform(-1, 1, 10)
}

df = pd.DataFrame(data)
def color_negative_red(val):
    color = 'red' if val < 0 else 'black'
    return f'color: {color}'
styled_df = df.style.applymap(color_negative_red)
styled_df
```

Output:



```
Region  Manager Salesman
Central Hermann John                 250.0
                Luis              150948.0
                Shelli             25000.0
                Sigal             121820.0
        Martha  Alexander          41930.0
                Steven             89850.0
        Timothy David               6075.0
East    Douglas Karen              40500.0
        Martha  Alexander         189146.0
                Diana              14500.0
West    Douglas Michael            38336.0
        Timothy Stephen            67088.0
```

11.Create a dataframe of ten rows, four columns with random values. Convert some values to nan values. Write a Pandas program which will highlight the nan values.



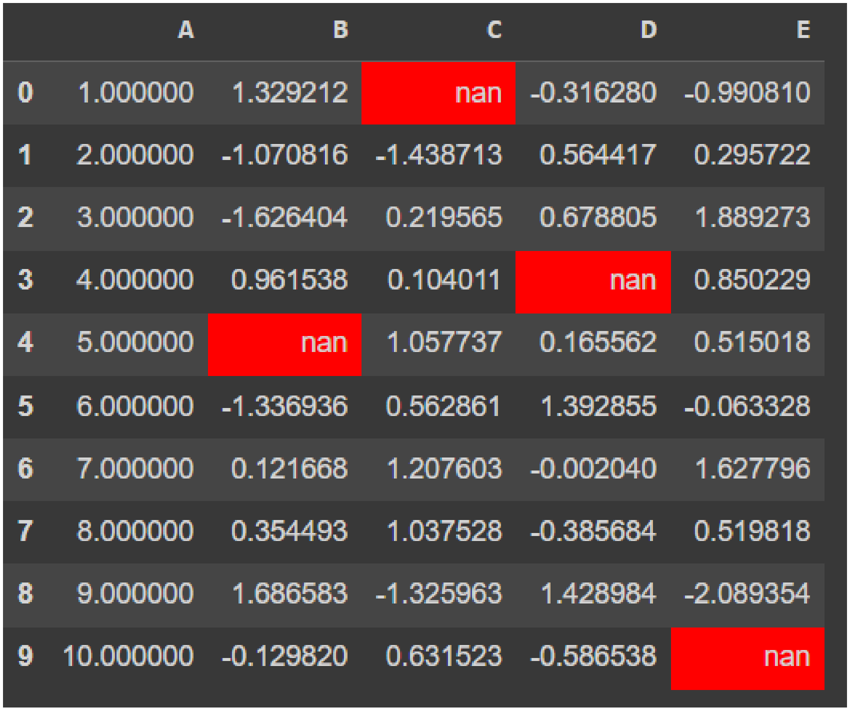
	A	B	C	D	E
0	1	1.32921	nan	-0.31628	-0.99081
1	2	-1.07082	-1.43871	0.564417	0.295722
2	3	-1.6264	0.219565	0.678805	1.88927
3	4	0.961538	0.104011	nan	0.850229
4	5	nan	1.05774	0.165562	0.515018
5	6	-1.33694	0.562861	1.39285	-0.063328
6	7	0.121668	1.2076	-0.00204021	1.6278
7	8	0.354493	1.03753	-0.385684	0.519818
8	9	1.68658	-1.32596	1.42898	-2.08935
9	10	-0.12982	0.631523	-0.586538	nan

Program:

```
import pandas as pd
import numpy as np
np.random.seed(24)
df = pd.DataFrame({'A': np.linspace(1, 10, 10)})
df = pd.concat([df, pd.DataFrame(np.random.randn(10, 4),
columns=list('BCDE'))],
           axis=1)
df.iloc[0, 2] = np.nan
df.iloc[3, 3] = np.nan
df.iloc[4, 1] = np.nan
df.iloc[9, 4] = np.nan
```

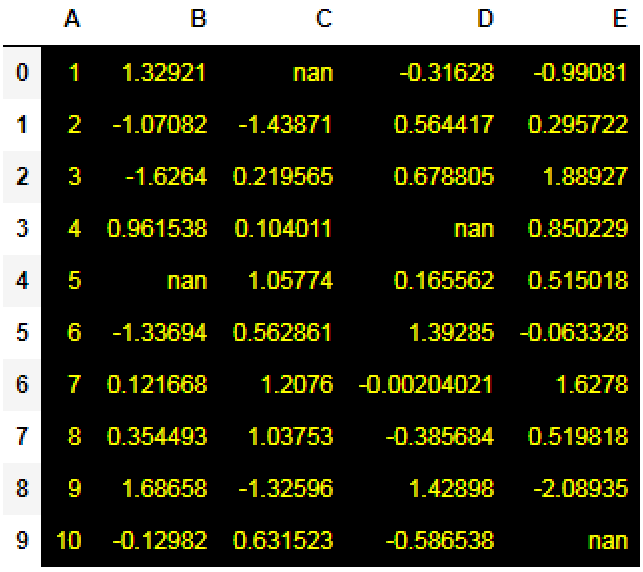
```
print("Original array:")
print(df)
def color_negative_red(val):
    color = 'red' if val < 0 else 'black'
    return 'color: %s' % color
print("\nNegative numbers red and positive numbers black:")
df.style.highlight_null(null_color='red')
```

Output:



	A	B	C	D	E
0	1.000000	1.329212	nan	-0.316280	-0.990810
1	2.000000	-1.070816	-1.438713	0.564417	0.295722
2	3.000000	-1.626404	0.219565	0.678805	1.889273
3	4.000000	0.961538	0.104011	nan	0.850229
4	5.000000	nan	1.057737	0.165562	0.515018
5	6.000000	-1.336936	0.562861	1.392855	-0.063328
6	7.000000	0.121668	1.207603	-0.002040	1.627796
7	8.000000	0.354493	1.037528	-0.385684	0.519818
8	9.000000	1.686583	-1.325963	1.428984	-2.089354
9	10.000000	-0.129820	0.631523	-0.586538	nan

12.Create a dataframe of ten rows, four columns with random values. Write a Pandas program to set dataframe background Color black and font color yellow.

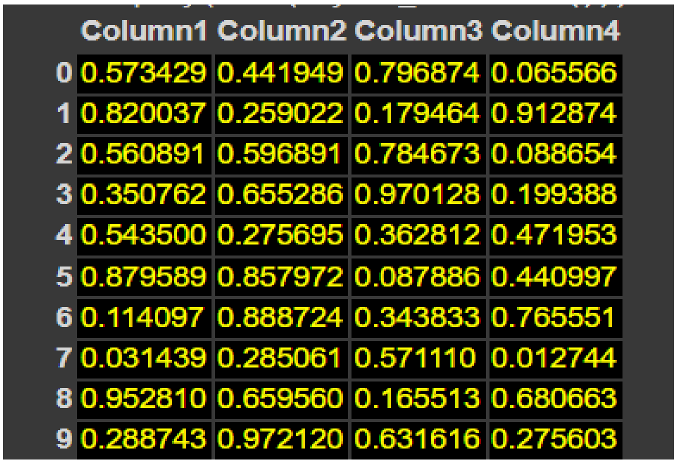


	A	B	C	D	E
0	1	1.32921	nan	-0.31628	-0.99081
1	2	-1.07082	-1.43871	0.564417	0.295722
2	3	-1.6264	0.219565	0.678805	1.88927
3	4	0.961538	0.104011	nan	0.850229
4	5	nan	1.05774	0.165562	0.515018
5	6	-1.33694	0.562861	1.39285	-0.063328
6	7	0.121668	1.2076	-0.00204021	1.6278
7	8	0.354493	1.03753	-0.385684	0.519818
8	9	1.68658	-1.32596	1.42898	-2.08935
9	10	-0.12982	0.631523	-0.586538	nan

Program:

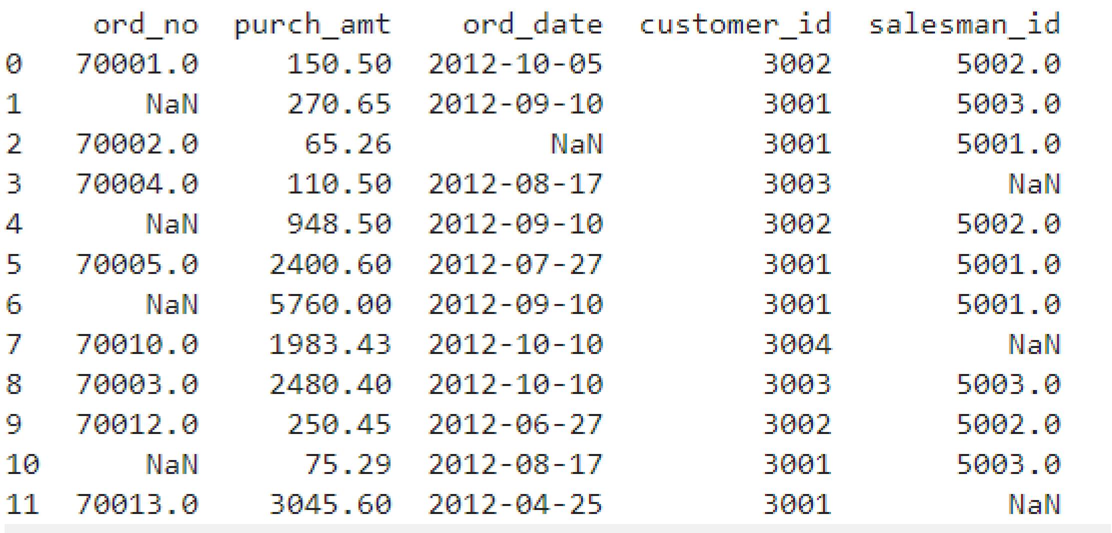
```
import pandas as pd

import numpy as np

from IPython.display import display, HTML

data = {

    'Column1': np.random.rand(10),

    'Column2': np.random.rand(10),

    'Column3': np.random.rand(10),

    'Column4': np.random.rand(10)

}

df = pd.DataFrame(data)

def highlight(val):

    return 'background-color: black; color: yellow;'

styled_df = df.style.applymap(highlight)

display(HTML(styled_df.render()))
```

output:



	Column1	Column2	Column3	Column4
0	0.573429	0.441949	0.796874	0.065566
1	0.820037	0.259022	0.179464	0.912874
2	0.560891	0.596891	0.784673	0.088654
3	0.350762	0.655286	0.970128	0.199388
4	0.543500	0.275695	0.362812	0.471953
5	0.879589	0.857972	0.087886	0.440997
6	0.114097	0.888724	0.343833	0.765551
7	0.031439	0.285061	0.571110	0.012744
8	0.952810	0.659560	0.165513	0.680663
9	0.288743	0.972120	0.631616	0.275603

13.Write a Pandas program to detect missing values of a given DataFrame. Display True or False.

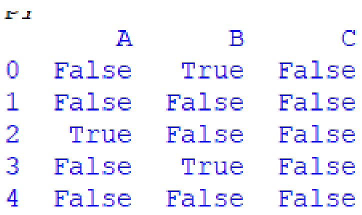


```
     ord_no  purch_amt    ord_date  customer_id  salesman_id
0   70001.0     150.50  2012-10-05         3002       5002.0
1       NaN     270.65  2012-09-10         3001       5003.0
2   70002.0      65.26         NaN         3001       5001.0
3   70004.0     110.50  2012-08-17         3003          NaN
4       NaN     948.50  2012-09-10         3002       5002.0
5   70005.0    2400.60  2012-07-27         3001       5001.0
6       NaN    5760.00  2012-09-10         3001       5001.0
7   70010.0    1983.43  2012-10-10         3004          NaN
8   70003.0    2480.40  2012-10-10         3003       5003.0
9   70012.0     250.45  2012-06-27         3002       5002.0
10      NaN      75.29  2012-08-17         3001       5003.0
11  70013.0    3045.60  2012-04-25         3001          NaN
```

Program:

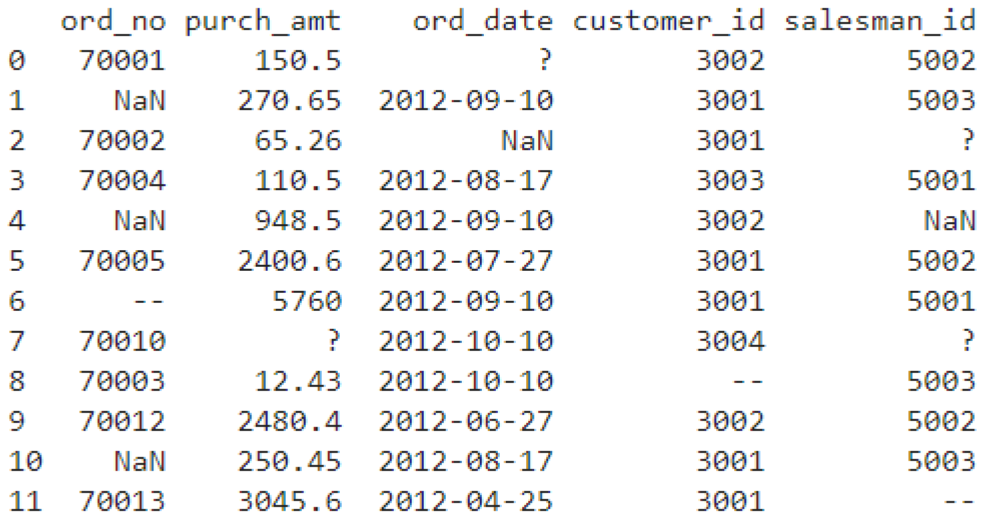
```
import pandas as pd

data = {'A': [1, 2, None, 4, 5],

        'B': [None, 2, 3, None, 5],

        'C': [1, 2, 3, 4, 5]}

df = pd.DataFrame(data)

missing_values = df.isna()

print(missing_values)
```

Output:



```
       A      B      C
0  False   True  False
1  False  False  False
2   True  False  False
3  False   True  False
4  False  False  False
```

14. Write a Pandas program to find and replace the missing values in a given DataFrame which do not have any valuable information.

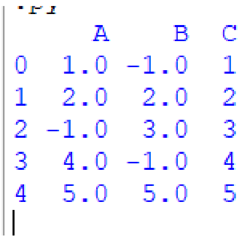


```
   ord_no purch_amt    ord_date customer_id salesman_id
0   70001     150.5           ?        3002        5002
1     NaN    270.65  2012-09-10        3001        5003
2   70002     65.26         NaN        3001           ?
3   70004     110.5  2012-08-17        3003        5001
4     NaN     948.5  2012-09-10        3002         NaN
5   70005    2400.6  2012-07-27        3001        5002
6      --      5760  2012-09-10        3001        5001
7   70010         ?  2012-10-10        3004           ?
8   70003     12.43  2012-10-10          --        5003
9   70012    2480.4  2012-06-27        3002        5002
10    NaN    250.45  2012-08-17        3001        5003
11  70013    3045.6  2012-04-25        3001          --
```

Program:

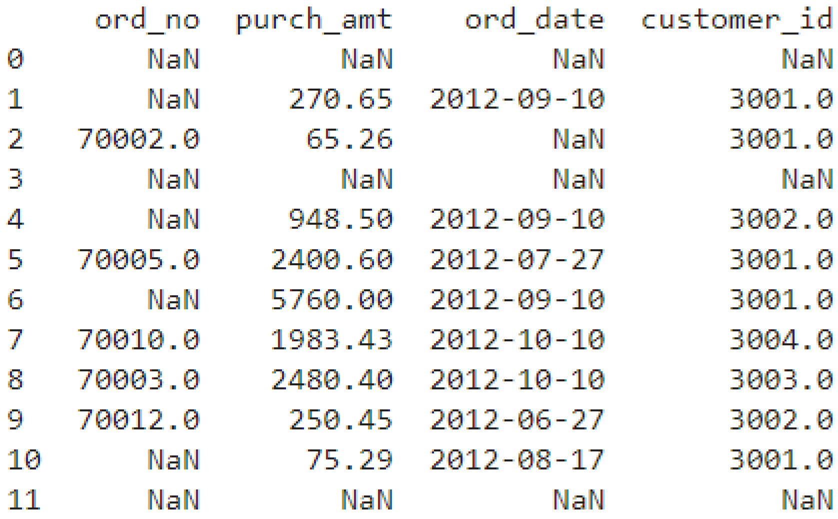
```
import pandas as pd

import numpy as np

data = {'A': [1, 2, None, 4, 5],

        'B': [None, 2, 3, None, 5],

        'C': [1, 2, 3, 4, 5]}

df = pd.DataFrame(data)

value_to_replace = -1

df_filled = df.fillna(value_to_replace)

print(df_filled)
```

Output:



```
     A    B  C
0  1.0 -1.0  1
1  2.0  2.0  2
2 -1.0  3.0  3
3  4.0 -1.0  4
4  5.0  5.0  5
```

15.Write a Pandas program to keep the rows with at least 2 NaN values in a given DataFrame.

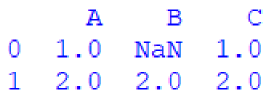


```
      ord_no  purch_amt    ord_date  customer_id
0        NaN        NaN         NaN          NaN
1        NaN     270.65  2012-09-10       3001.0
2    70002.0      65.26         NaN       3001.0
3        NaN        NaN         NaN          NaN
4        NaN     948.50  2012-09-10       3002.0
5    70005.0    2400.60  2012-07-27       3001.0
6        NaN    5760.00  2012-09-10       3001.0
7    70010.0    1983.43  2012-10-10       3004.0
8    70003.0    2480.40  2012-10-10       3003.0
9    70012.0     250.45  2012-06-27       3002.0
10       NaN      75.29  2012-08-17       3001.0
11       NaN        NaN         NaN          NaN
```

Program:

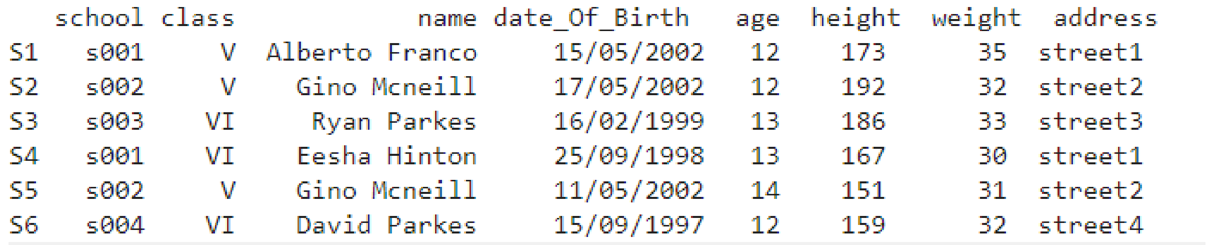
```
import pandas as pd

import numpy as np

data = {'A': [1, 2, None, 4, None],

        'B': [None, 2, 3, None, None],

        'C': [1, 2, None, None, 5]}


df = pd.DataFrame(data)

threshold = 2

filtered_df = df.dropna(thresh=threshold)

print(filtered_df)
```

Output:



```
     A    B    C
0  1.0  NaN  1.0
1  2.0  2.0  2.0
```

16.Write a Pandas program to split the following dataframe into groups based on school code. Also check the type of GroupBy object.

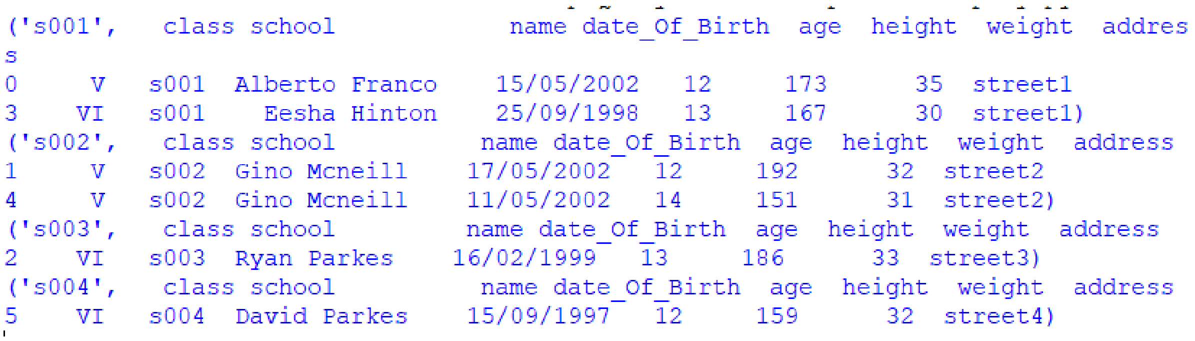


```
   school class            name date_Of_Birth   age  height   weight  address
S1    s001     V  Alberto Franco     15/05/2002   12     173       35  street1
S2    s002     V    Gino Mcneill     17/05/2002   12     192       32  street2
S3    s003    VI     Ryan Parkes     16/02/1999   13     186       33  street3
S4    s001    VI    Eesha Hinton     25/09/1998   13     167       30  street1
S5    s002     V    Gino Mcneill     11/05/2002   14     151       31  street2
S6    s004    VI    David Parkes     15/09/1997   12     159       32  street4
```

Program:

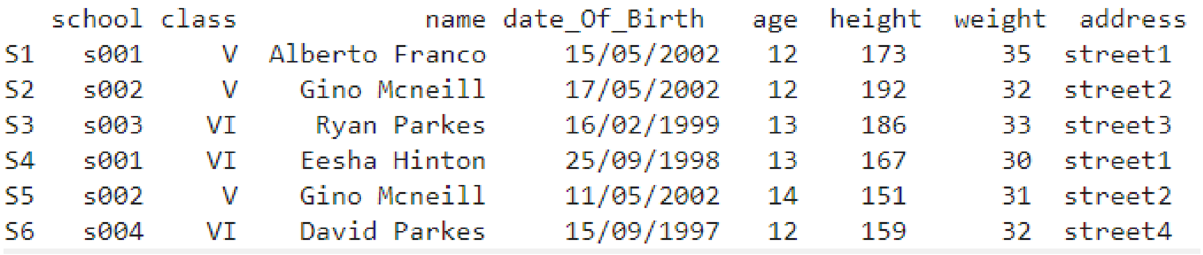
```
import pandas as pd

df = pd.DataFrame({'class': ['V', 'V', 'VI', 'VI', 'V', 'VI'],

                   'school': ['s001', 's002', 's003', 's001', 's002', 's004'],

                   'name': ['Alberto Franco', 'Gino Mcneill', 'Ryan Parkes', 'Eesha Hinton', 'Gino Mcneill', 'David Parkes'],

                   'date_Of_Birth': ['15/05/2002', '17/05/2002', '16/02/1999', '25/09/1998', '11/05/2002', '15/09/1997'],

                   'age': [12, 12, 13, 13, 14, 12],

                   'height': [173, 192, 186, 167, 151, 159],

                   'weight': [35, 32, 33, 30, 31, 32],

                   'address': ['street1', 'street2', 'street3', 'street1', 'street2', 'street4']})

groups = list(df.groupby('school'))

for group in groups:

    print(group)
```

Output:



```
('s001',    class school            name date_Of_Birth  age  height  weight  addres
s
0      V    s001  Alberto Franco    15/05/2002   12     173      35  street1
3     VI    s001    Eesha Hinton    25/09/1998   13     167      30  street1)
('s002',    class school          name date_Of_Birth  age  height  weight  address
1      V    s002  Gino Mcneill    17/05/2002   12     192      32  street2
4      V    s002  Gino Mcneill    11/05/2002   14     151      31  street2)
('s003',    class school         name date_Of_Birth  age  height  weight  address
2     VI    s003  Ryan Parkes    16/02/1999   13     186      33  street3)
('s004',    class school          name date_Of_Birth  age  height  weight  address
5     VI    s004  David Parkes    15/09/1997   12     159      32  street4)
```

17.Write a Pandas program to split the following dataframe by school code and get mean, min, and max value of age for each school.



```
   school class            name date_Of_Birth   age  height   weight  address
S1   s001     V  Alberto Franco     15/05/2002   12     173       35  street1
S2   s002     V    Gino Mcneill     17/05/2002   12     192       32  street2
S3   s003    VI     Ryan Parkes     16/02/1999   13     186       33  street3
S4   s001    VI    Eesha Hinton     25/09/1998   13     167       30  street1
S5   s002     V    Gino Mcneill     11/05/2002   14     151       31  street2
S6   s004    VI    David Parkes     15/09/1997   12     159       32  street4
```

Program:

import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],

'Age': [25, 22, 24, 23, 25],

'School Code': ['S001', 'S002', 'S001', 'S003', 'S002']}

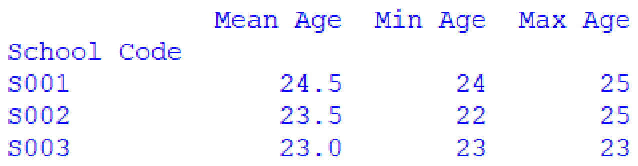
df = pd.DataFrame(data)

result = df.groupby('School Code')['Age'].agg(['mean', 'min', 'max'])

result = result.rename(columns={'mean': 'Mean Age', 'min': 'Min Age', 'max': 'Max Age'})

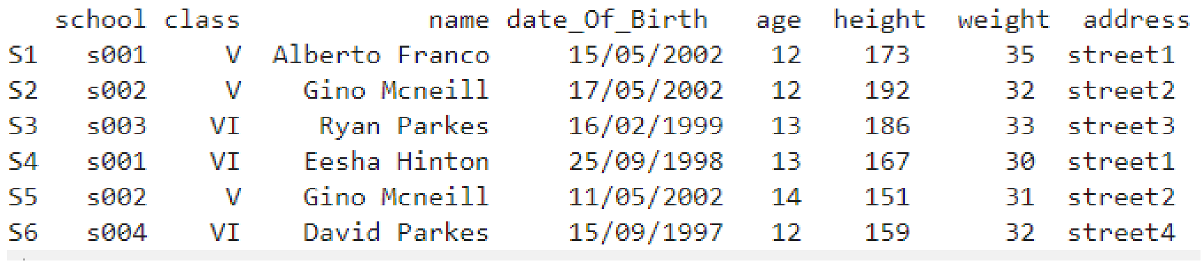
print(result)

Output:



```
             Mean Age  Min Age  Max Age
School Code                            
S001             24.5       24       25
S002             23.5       22       25
S003             23.0       23       23
```

18.Write a Pandas program to split the following given dataframe into groups based on school code and class.

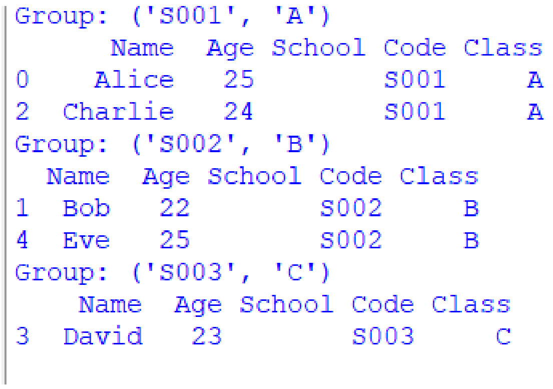


```
   school class            name date_Of_Birth   age  height  weight  address
S1   s001     V  Alberto Franco     15/05/2002   12     173      35  street1
S2   s002     V    Gino Mcneill     17/05/2002   12     192      32  street2
S3   s003    VI     Ryan Parkes     16/02/1999   13     186      33  street3
S4   s001    VI    Eesha Hinton     25/09/1998   13     167      30  street1
S5   s002     V    Gino Mcneill     11/05/2002   14     151      31  street2
S6   s004    VI    David Parkes     15/09/1997   12     159      32  street4
```

Program:

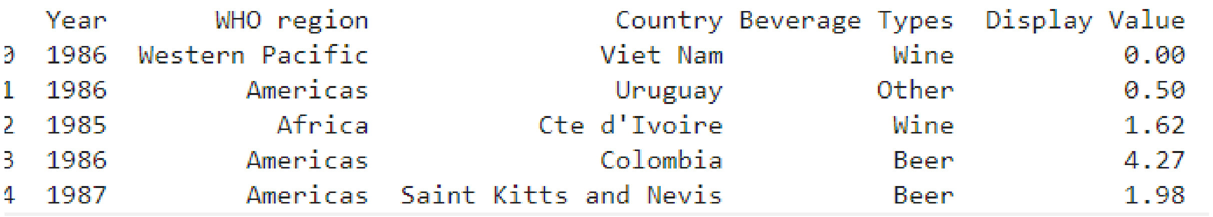
```
import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],

        'Age': [25, 22, 24, 23, 25],

        'School Code': ['S001', 'S002', 'S001', 'S003', 'S002'],

        'Class': ['A', 'B', 'A', 'C', 'B']}

df = pd.DataFrame(data)

grouped = df.groupby(['School Code', 'Class'])

for group, group_df in grouped:

    print("Group:", group)

    print(group_df)
```

Output:



```
Group: ('S001', 'A')
      Name  Age School Code Class
0    Alice   25        S001     A
2  Charlie   24        S001     A
Group: ('S002', 'B')
  Name  Age School Code Class
1  Bob   22        S002     B
4  Eve   25        S002     B
Group: ('S003', 'C')
    Name  Age School Code Class
3  David   23        S003     C
```

19.Write a Pandas program to display the dimensions or shape of the World alcohol consumption dataset. Also extract the column names from the dataset.



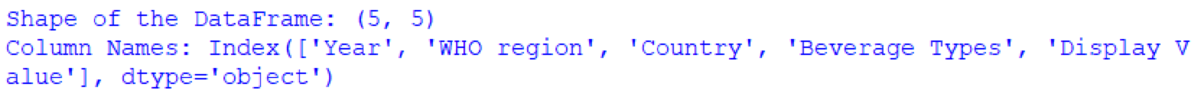
```
   Year       WHO region                Country Beverage Types  Display Value
0  1986  Western Pacific               Viet Nam           Wine           0.00
1  1986         Americas                Uruguay          Other           0.50
2  1985           Africa           Cte d'Ivoire           Wine           1.62
3  1986         Americas               Colombia           Beer           4.27
4  1987         Americas  Saint Kitts and Nevis           Beer           1.98
```

Program:

```
import pandas as pd
data = {
    'Year': [1986, 1986, 1986, 1986, 1985],
    'WHO region': ['Western Pacific', 'Americas', 'Africa', 'Americas', 'Americas'],
    'Country': ["Viet Nam", "Uruguay", "Cote d'Ivoire", "Colombia", "Saint Kitts"],
    'Beverage Types': ['Wine', 'Other', 'Wine', 'Beer', 'Beer'],
    'Display Value': [0.00, 0.50, 3.62, 4.27, 1.98]
}
df = pd.DataFrame(data)
shape = df.shape
print("Shape of the DataFrame:", shape)
```

```
column_names = df.columns

print("Column Names:", column_names)
```

Output:



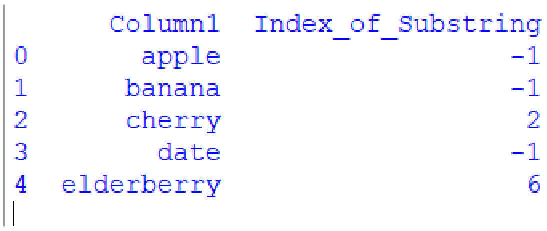
```
Shape of the DataFrame: (5, 5)
Column Names: Index(['Year', 'WHO region', 'Country', 'Beverage Types', 'Display V
alue'], dtype='object')
```

20.Write a Pandas program to find the index of a given substring of a DataFrame column.

Program:

```
import pandas as pd

data = {'Column1': ['apple', 'banana', 'cherry', 'date', 'elderberry']}

df = pd.DataFrame(data)

substring = 'erry'

index_of_substring = df['Column1'].str.find(substring)

df['Index_of_Substring'] = index_of_substring

print(df)
```

Output:



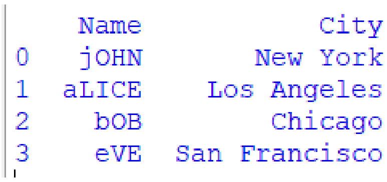
```
      Column1  Index_of_Substring
0       apple                  -1
1      banana                  -1
2      cherry                   2
3        date                  -1
4  elderberry                   6
```

21.Write a Pandas program to swap the cases of a specified character column in a given DataFrame.

Program:

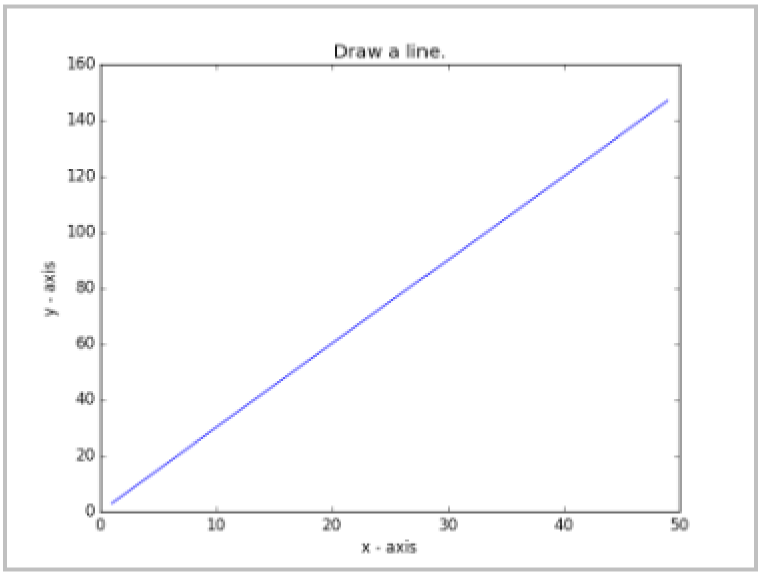
```
import pandas as pd
data = {'Name': ['John', 'Alice', 'Bob', 'Eve'],
        'City': ['New York', 'Los Angeles', 'Chicago', 'San Francisco']}
df = pd.DataFrame(data)
column_to_swap = 'Name'
df[column_to_swap] = df[column_to_swap].str.swapcase()
print(df)
```

Output:



```
    Name           City
0   jOHN       New York
1  aLICE    Los Angeles
2    bOB        Chicago
3    eVE  San Francisco
```

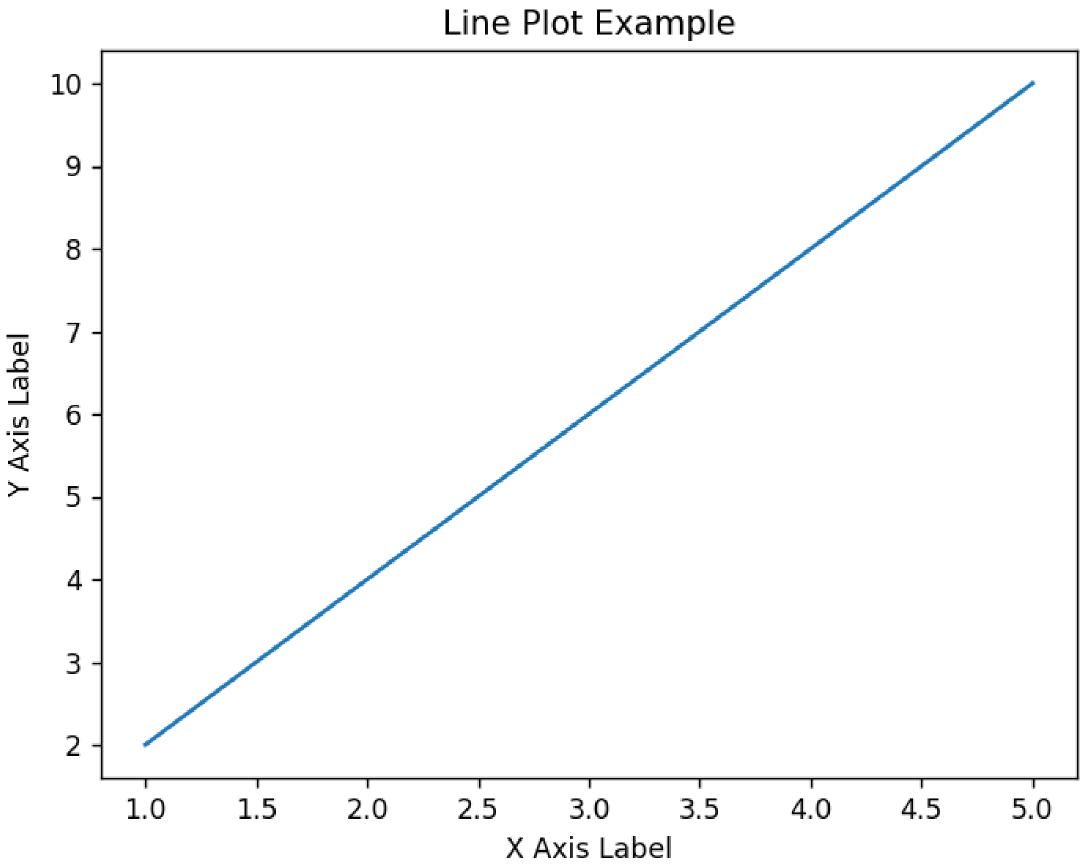
22.Write a Python program to draw a line with suitable label in the x axis, y axis and a title.



Program:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]

y = [2, 4, 6, 8, 10]

plt.plot(x, y)

plt.xlabel('X Axis Label')

plt.ylabel('Y Axis Label')

plt.title('Line Plot Example')

plt.show()
```

Output:

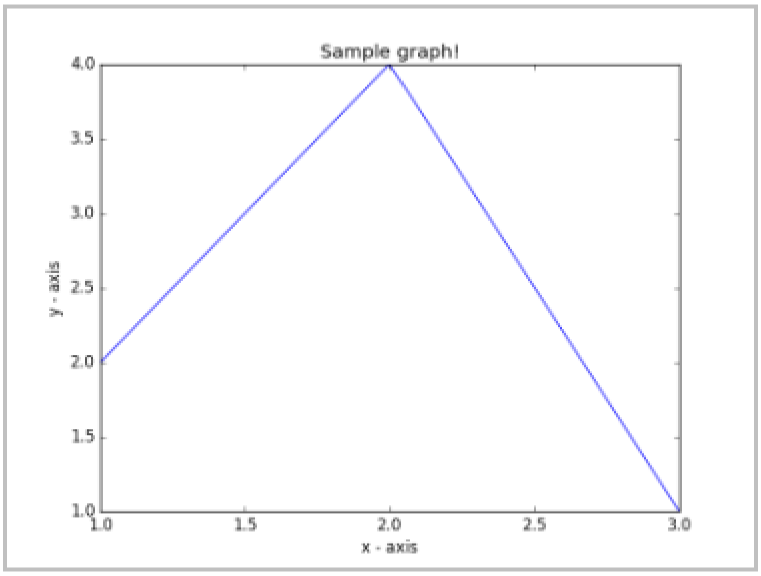


23.Write a Python program to draw a line using given axis values taken from a text file, with suitable label in the x axis, y axis and a title.

Test Data:

test.txt

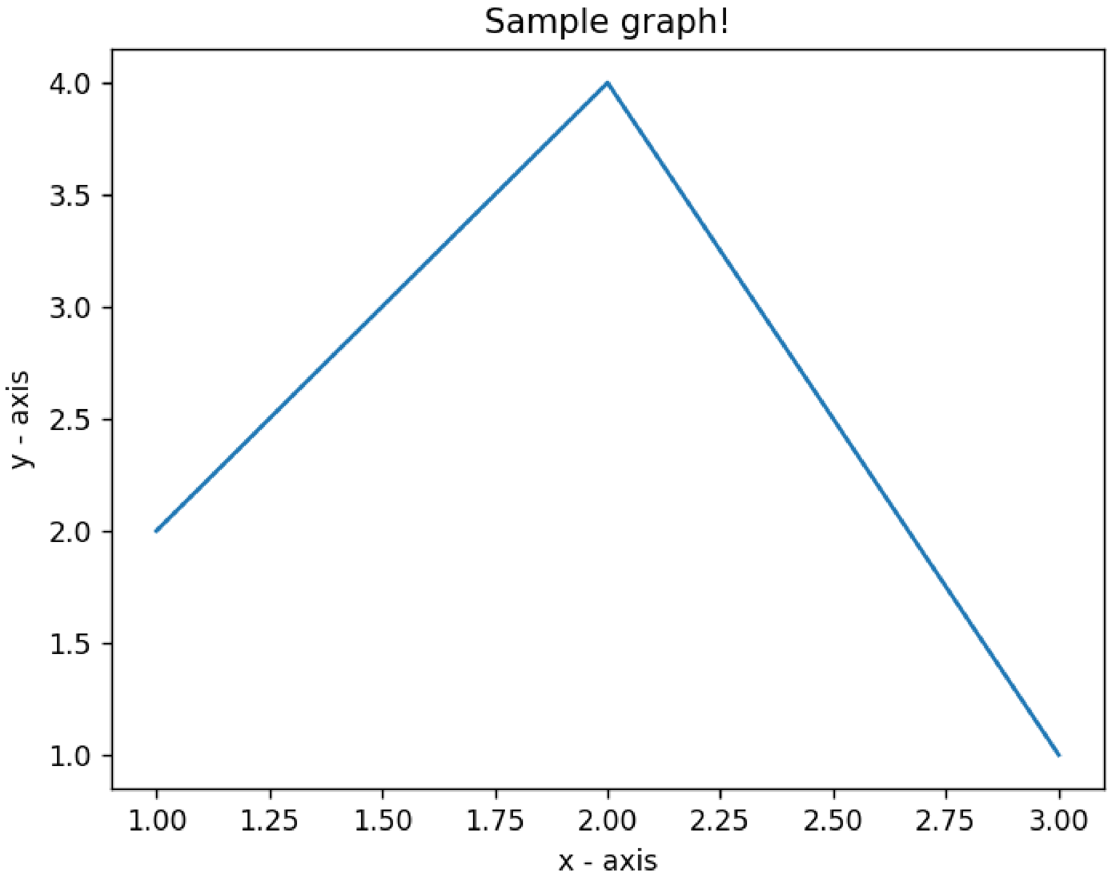
```
1 2
2 4
3 1
```



Program:

```
import matplotlib.pyplot as plt

x = [1,2,3]

y = [2,4,1]

plt.plot(x, y)

plt.xlabel('x - axis')

plt.ylabel('y - axis')

plt.title('Sample graph!')

plt.show()
```

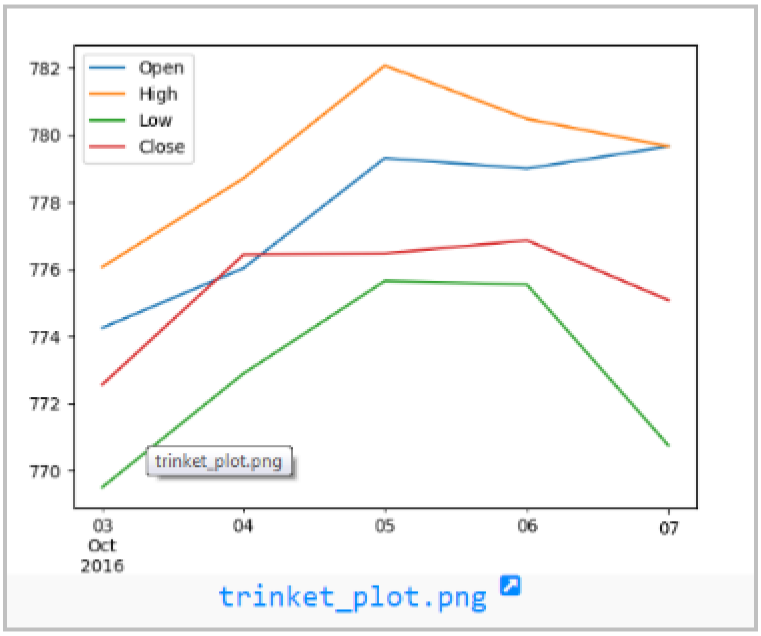
Output:



24.Write a Python program to draw line charts of the financial data of Alphabet Inc. between October 3, 2016 to October 7, 2016.

Sample Financial data (fdata.csv):

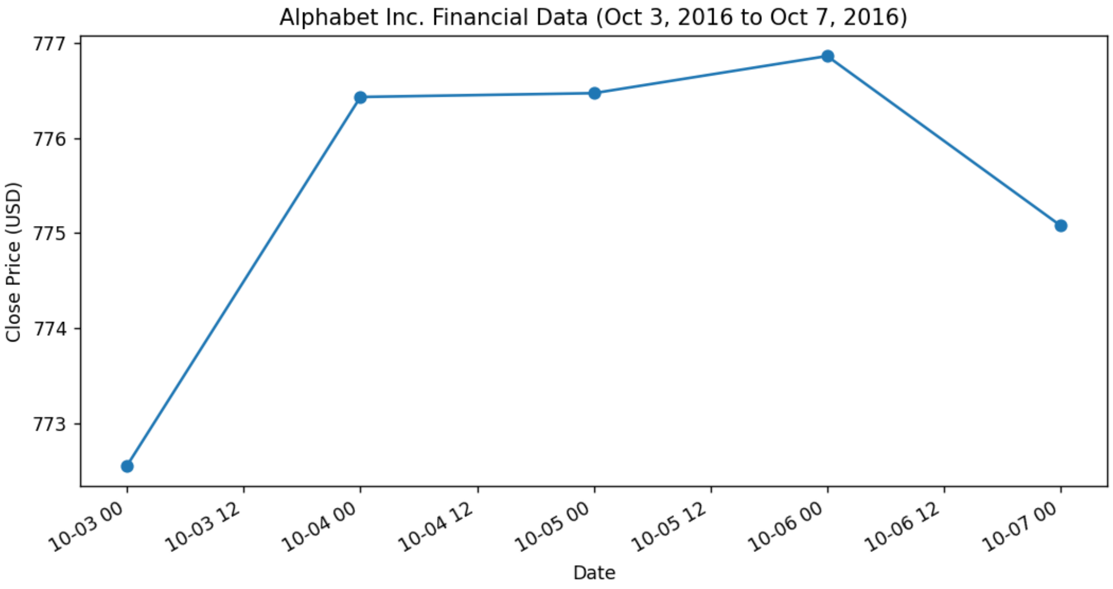
```
Date,Open,High,Low,Close
10-03-16,774.25,776.065002,769.5,772.559998
10-04-16,776.030029,778.710022,772.890015,776.429993
10-05-16,779.309998,782.070007,775.650024,776.469971
10-06-16,779,780.47998,775.539978,776.859985
10-07-16,779.659973,779.659973,770.75,775.080017
```



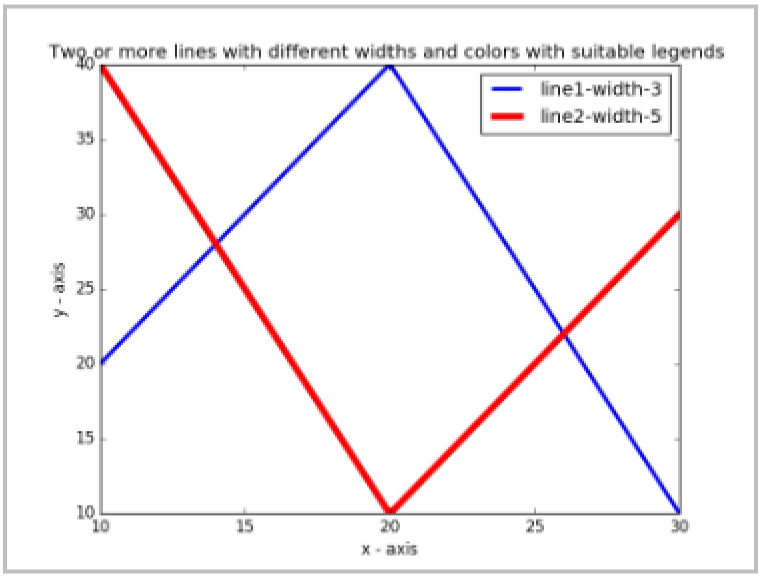
Program:

```
import matplotlib.pyplot as plt

from datetime import datetime

data = [

{"Date": "10-03-16", "Open": 774.25, "High": 776.065002, "Low": 769.5, "Close": 772.559998},

{"Date": "10-04-16", "Open": 776.030029, "High": 778.710022, "Low": 772.890015, "Close": 776.429993},

{"Date": "10-05-16", "Open": 779.309998, "High": 782.070007, "Low": 775.650024, "Close": 776.469971},

{"Date": "10-06-16", "Open": 779, "High": 780.47998, "Low": 775.539978, "Close": 776.859985}, {"Date": "10-07-16", "Open":779.659973, "High": 779.659973, "Low": 770.75, "Close": 775.080017}]

dates = [datetime.strptime(entry["Date"], "%m-%d-%y") for entry in data]

close_prices = [entry["Close"] for entry in data]

plt.figure(figsize=(10,5))
plt.plot(dates, close_prices, marker='o', linestyle='-')

plt.xlabel('Date')

plt.ylabel('Close Price (USD)')

plt.title('Alphabet Inc. Financial Data (Oct 3, 2016 to Oct 7, 2016)')

plt.gcf().autofmt_xdate()

plt.show()
```

Output:



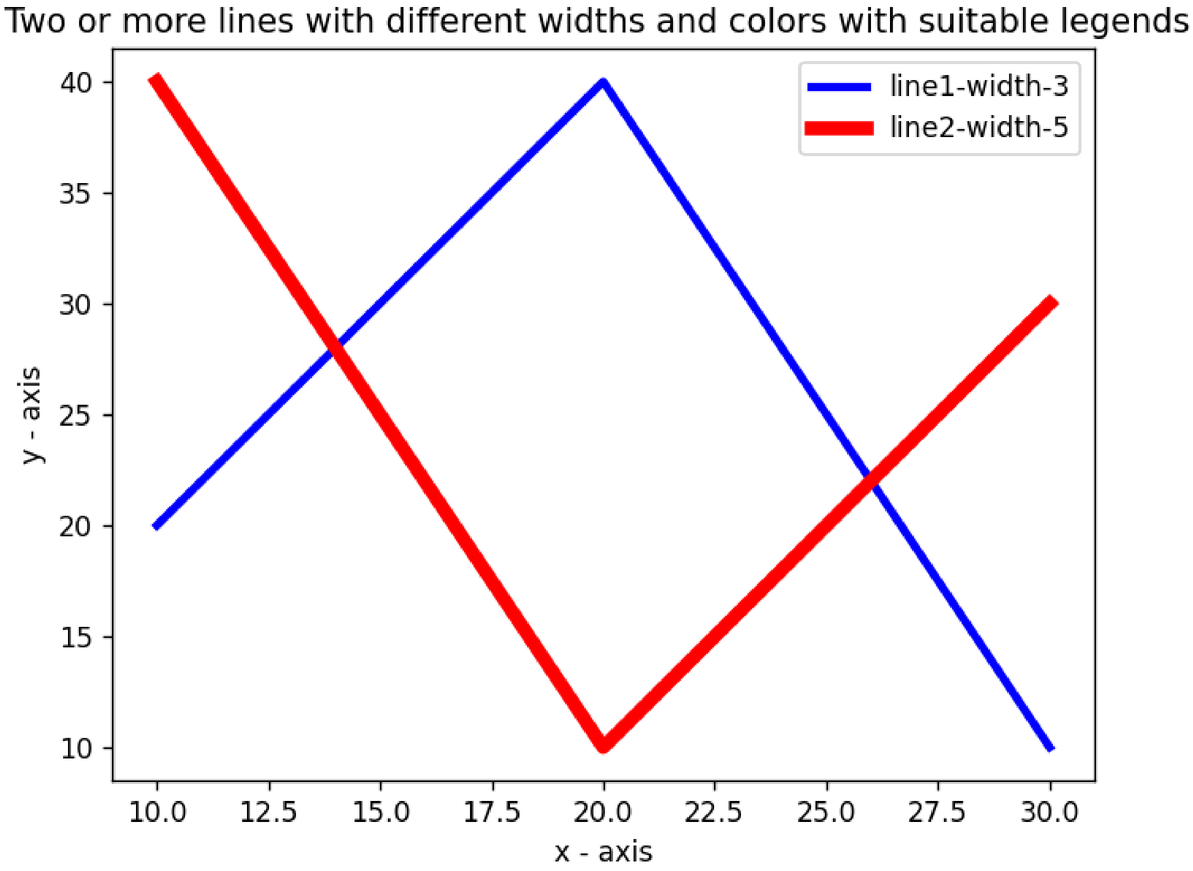
25.Write a Python program to plot two or more lines with legends, different widths and colors.



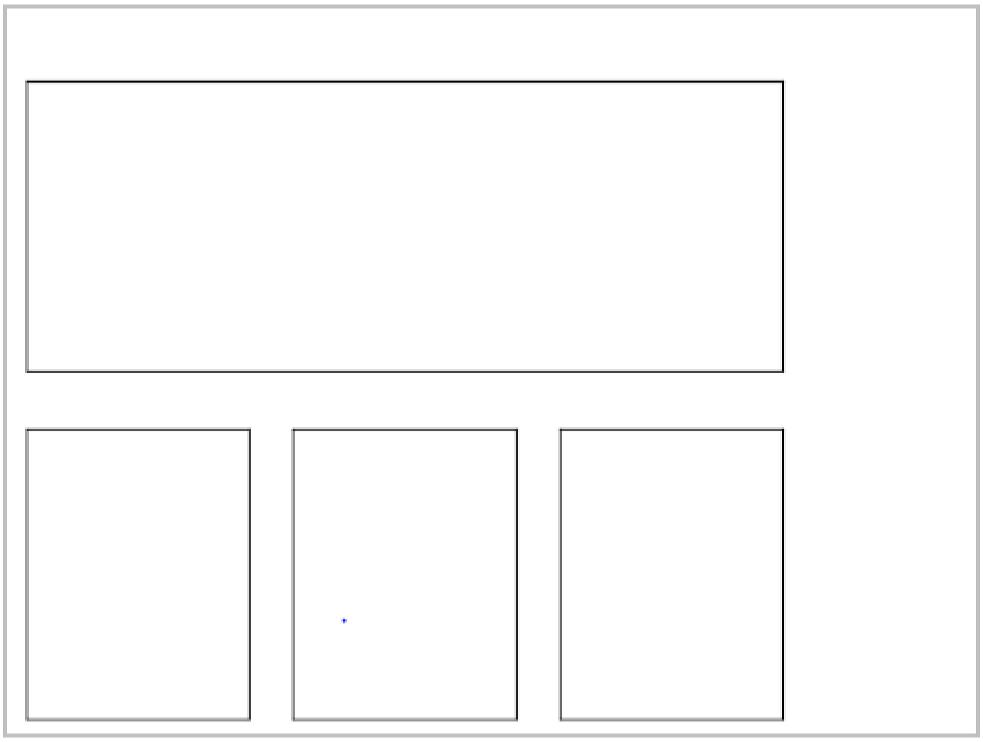
Program:

```
import matplotlib.pyplot as plt

x1 = [10,20,30]

y1 = [20,40,10]

x2 = [10,20,30]

y2 = [40,10,30]

plt.xlabel('x - axis')

plt.ylabel('y - axis')

plt.title('Two or more lines with different widths and colors with suitable legends ')

plt.plot(x1,y1, color='blue', linewidth = 3,  label = 'line1-width-3')

plt.plot(x2,y2, color='red', linewidth = 5,  label = 'line2-width-5')

plt.legend()

plt.show()
```

Output:



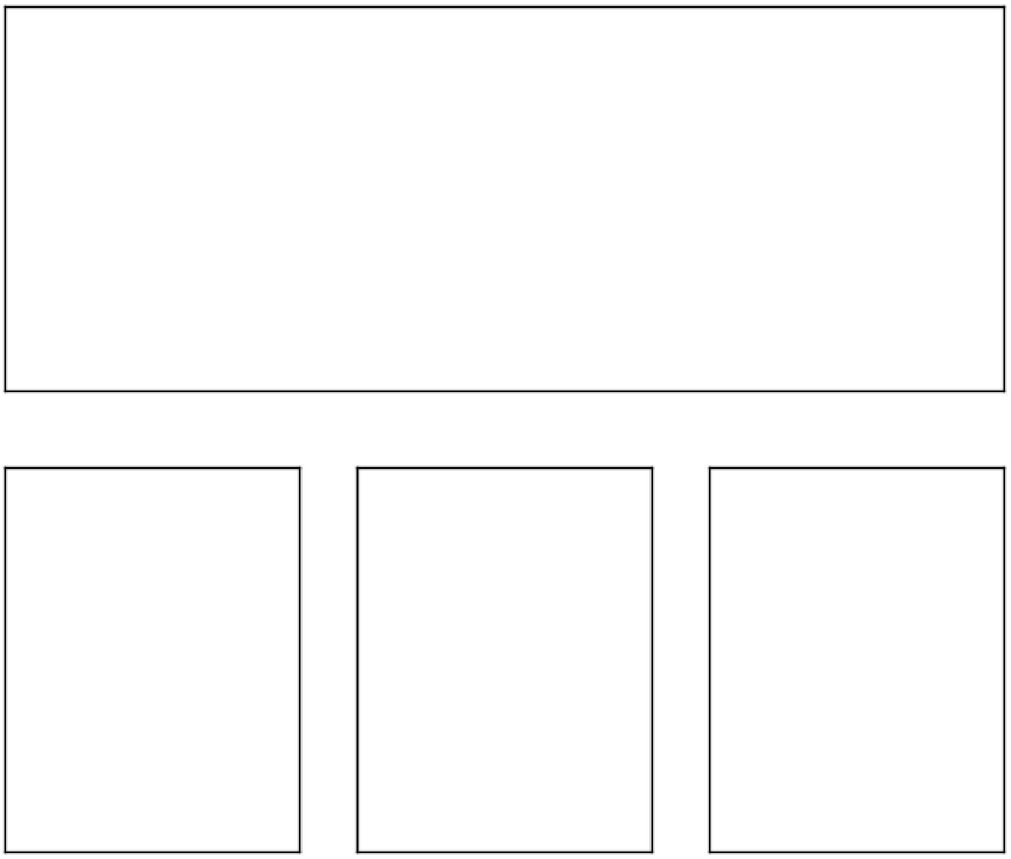
26.Write a Python program to create multiple plots.



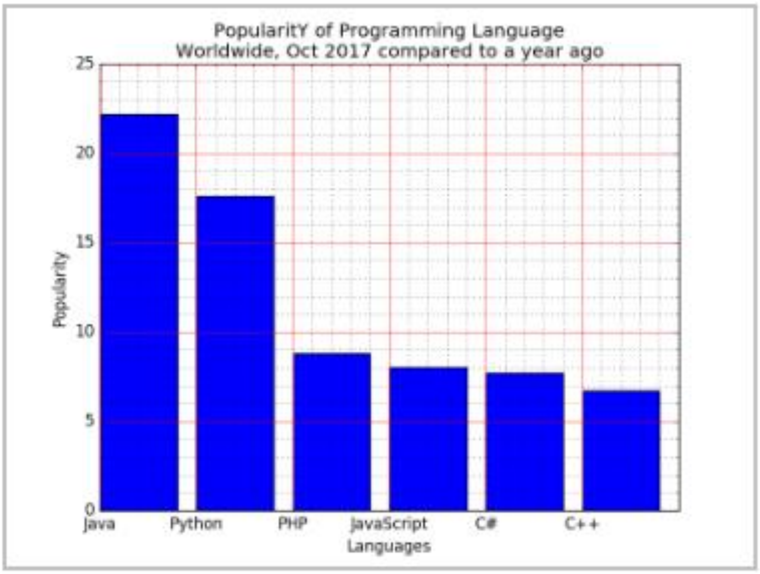
Program:

```
import matplotlib.pyplot as plt
fig = plt.figure()
fig.subplots_adjust(bottom=0.020, left=0.020, top = 0.900, right=0.800)
plt.subplot(2, 1, 1)
plt.xticks(()), plt.yticks(())
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()\
```

Output:



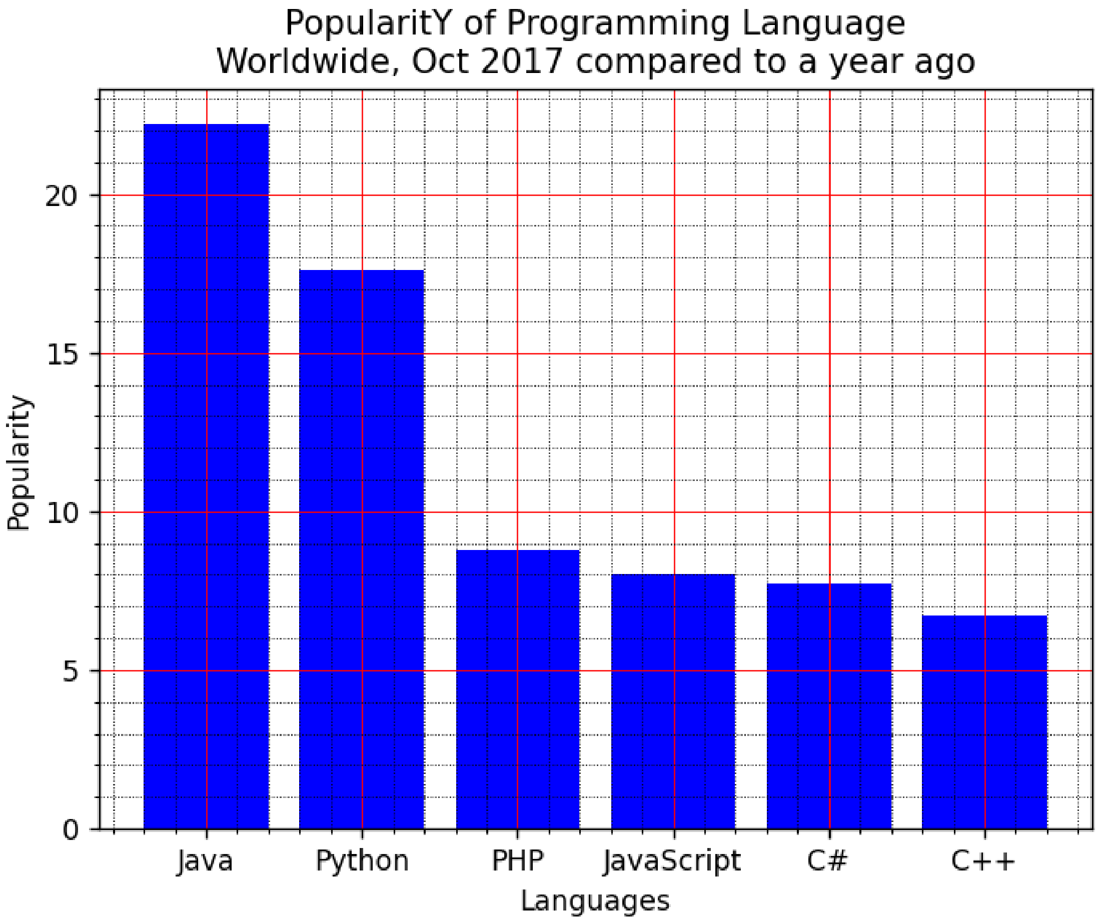
27.Write a Python programming to display a bar chart of the popularity of programming Languages.
Sample data:
Programming languages: Java, Python, PHP, JavaScript, C#, C++
Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7



Program:

```
import matplotlib.pyplot as plt

x = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']

popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]

x_pos = [i for i, _ in enumerate(x)]

plt.bar(x_pos, popularity, color='blue')

plt.xlabel("Languages")

plt.ylabel("Popularity")

plt.title("PopularitY of Programming Language\n" + "Worldwide, Oct 2017 compared to a year ago")

plt.xticks(x_pos, x)

plt.minorticks_on()

plt.grid(which='major', linestyle='-', linewidth='0.5', color='red')

plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')

plt.show()
```

Output:

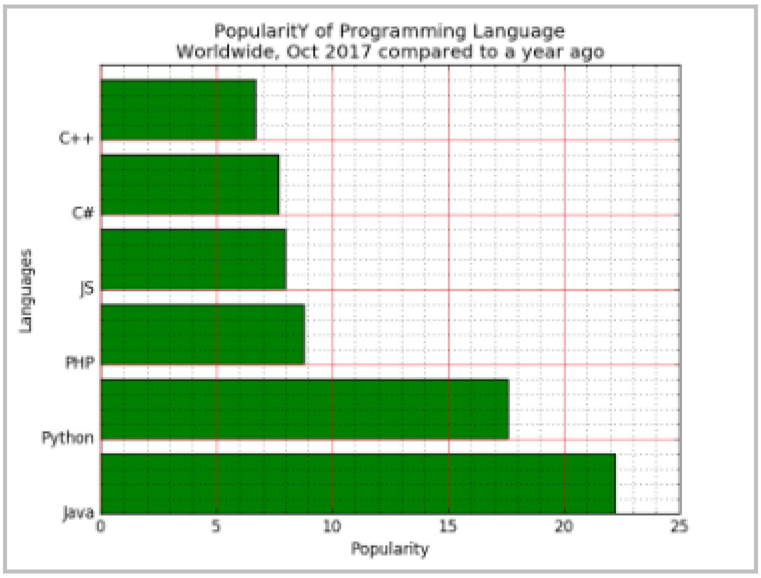


28.Write a Python programming to display a horizontal bar chart of the popularity of programming Languages.

Sample data:

Programming languages: Java, Python, PHP, JavaScript, C#, C++

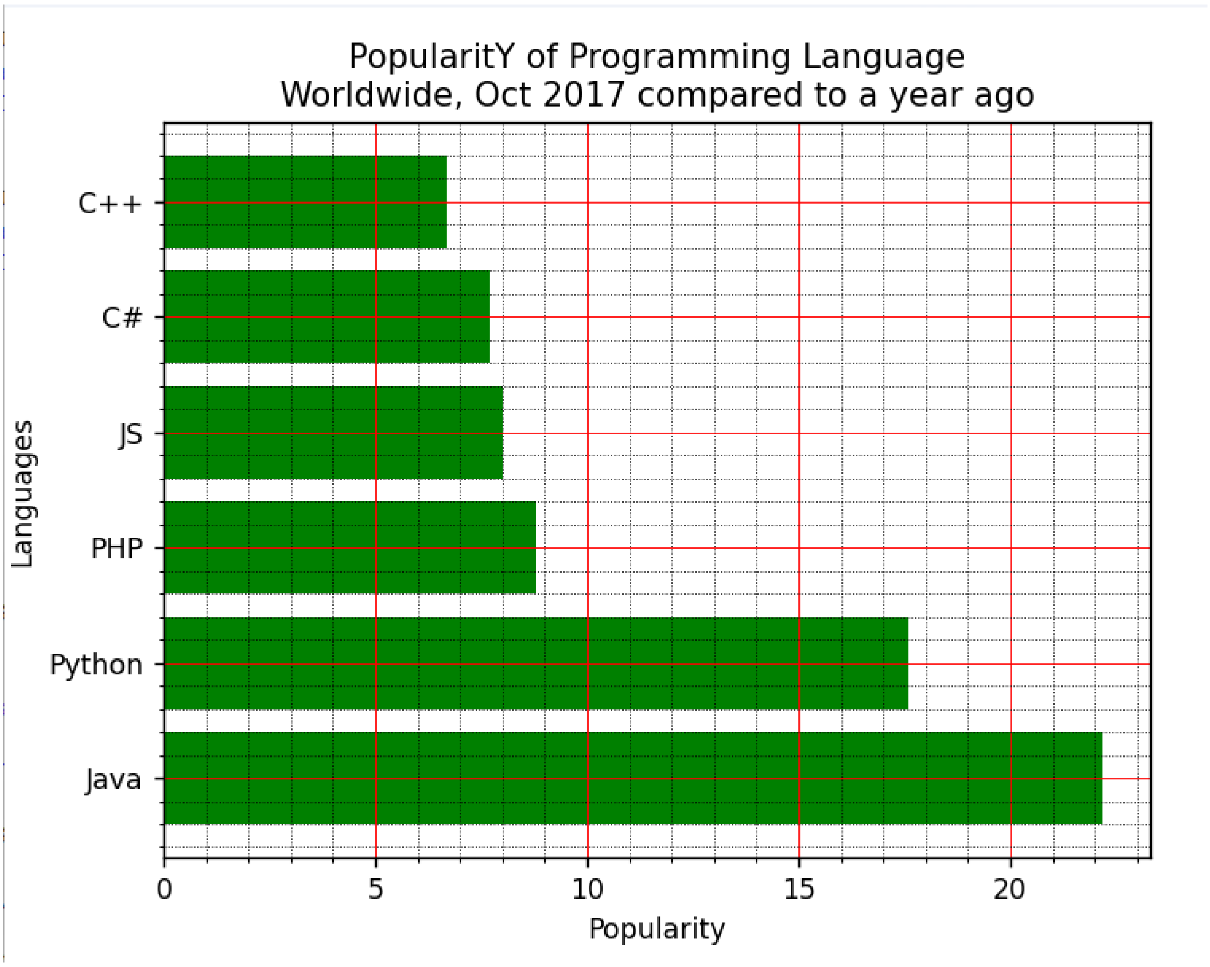
Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7



Program:

```
import matplotlib.pyplot as plt

x = ['Java', 'Python', 'PHP', 'JS', 'C#', 'C++']

popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]

x_pos = [i for i, _ in enumerate(x)]

plt.barh(x_pos, popularity, color='green')

plt.xlabel("Popularity")

plt.ylabel("Languages")

plt.title("PopularitY of Programming Language\n" + "Worldwide, Oct 2017 compared to a year ago")

plt.yticks(x_pos, x)

plt.minorticks_on()

plt.grid(which='major', linestyle='-', linewidth='0.5', color='red')

plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')

plt.show()
```

Output:

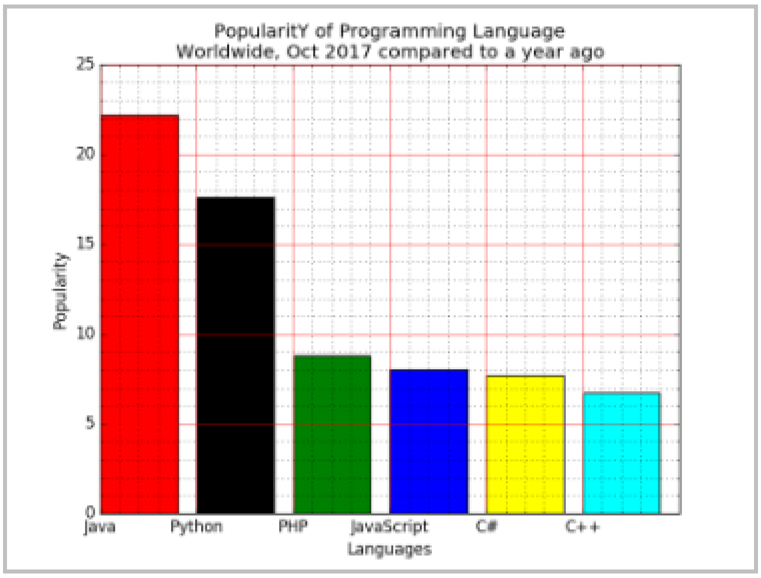


29.Write a Python programming to display a bar chart of the popularity of programming Languages. Use different color for each bar.

Sample data:

Programming languages: Java, Python, PHP, JavaScript, C#, C++

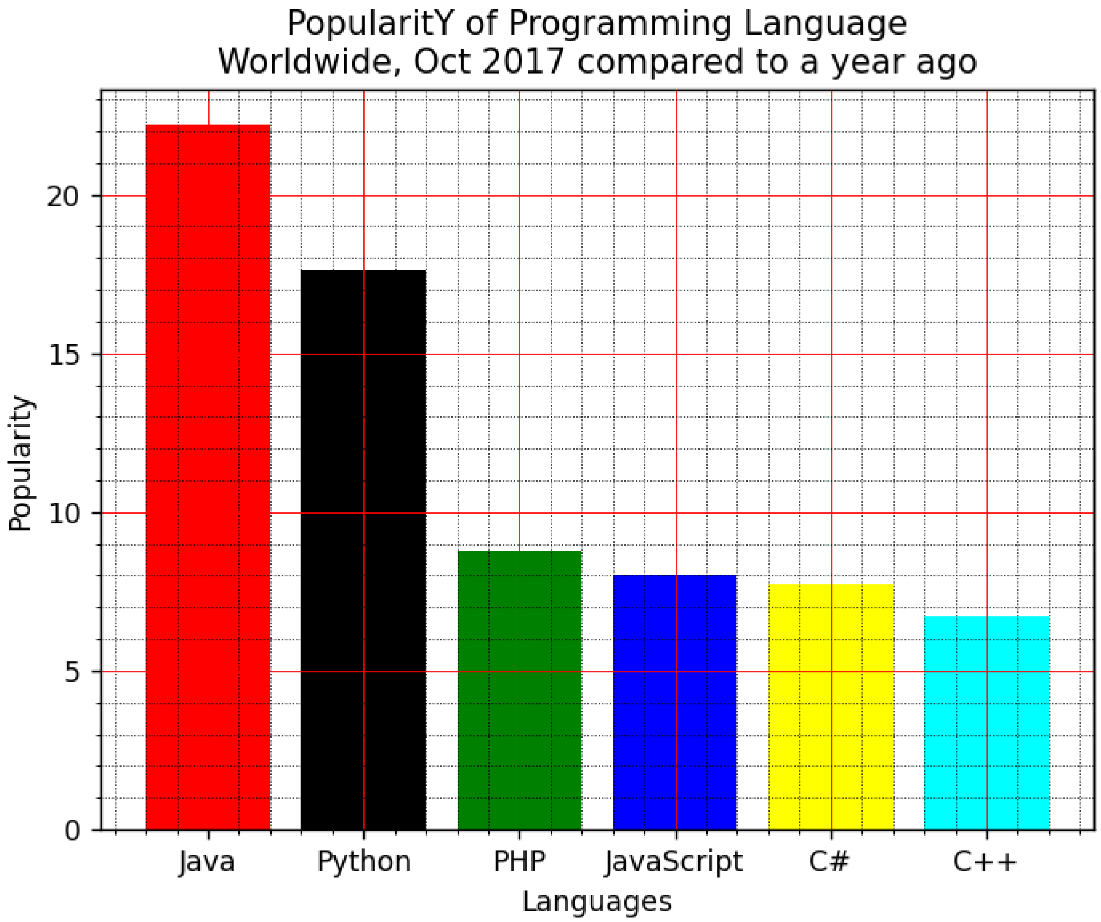
Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7



Program:

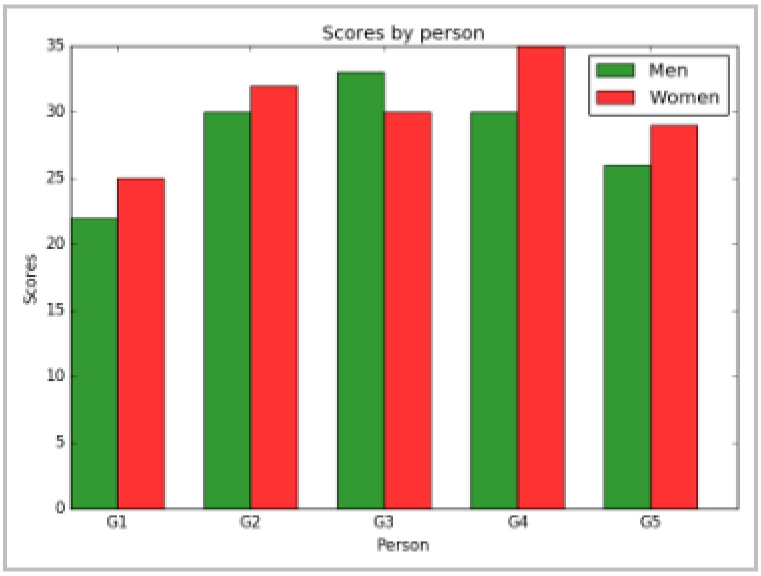
```
import matplotlib.pyplot as plt

x = ['Java', 'Python', 'PHP', 'JavaScript', 'C#', 'C++']

popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]

x_pos = [i for i, _ in enumerate(x)]


plt.bar(x_pos, popularity, color=['red', 'black', 'green', 'blue', 'yellow', 'cyan'])


plt.xlabel("Languages")

plt.ylabel("Popularity")

plt.title("PopularitY of Programming Language\n" + "Worldwide, Oct 2017 compared to a year ago")

plt.xticks(x_pos, x)

plt.minorticks_on()

plt.grid(which='major', linestyle='-', linewidth='0.5', color='red')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')

plt.show()
```

Output:



30.Write a Python program to create bar plot of scores by group and gender. Use multiple X values on the same chart for men and women.

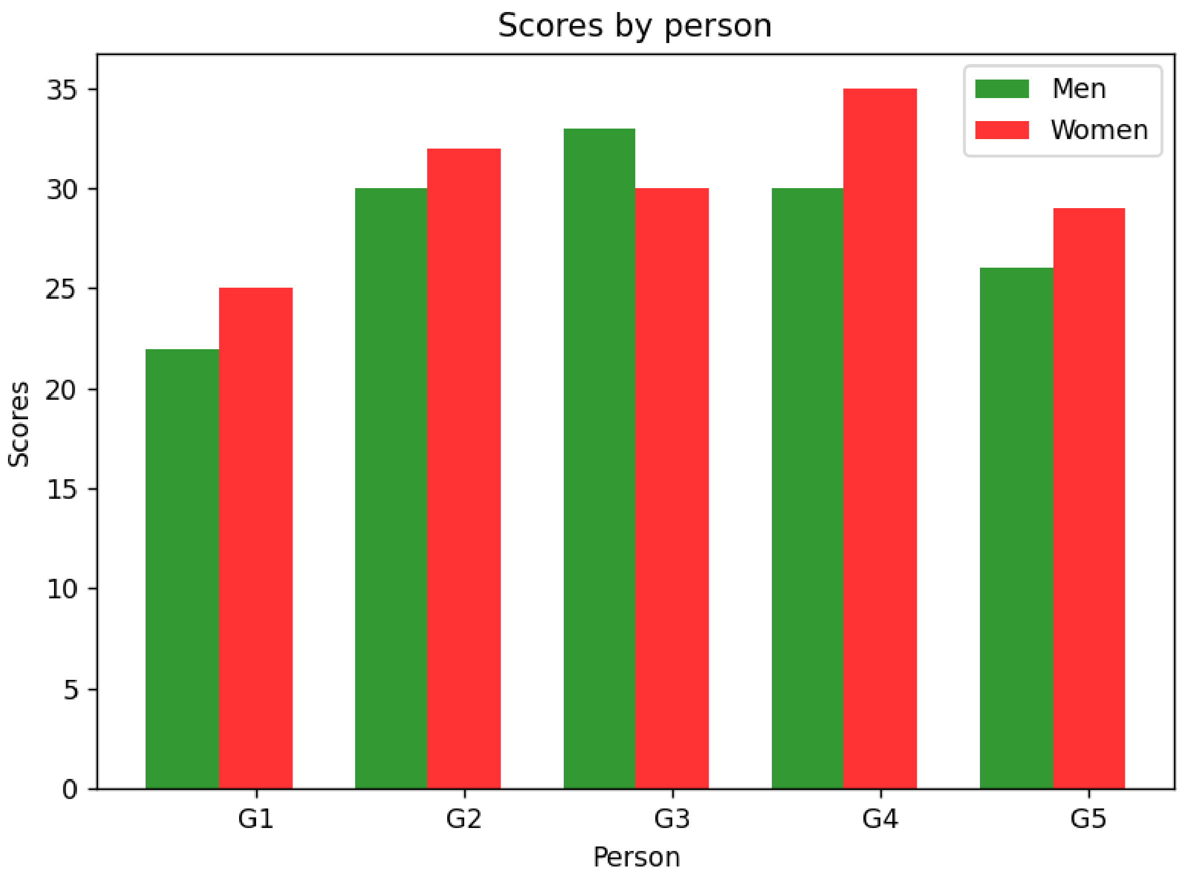
Sample Data:
Means (men) = (22, 30, 35, 35, 26)
Means (women) = (25, 32, 30, 35, 29)



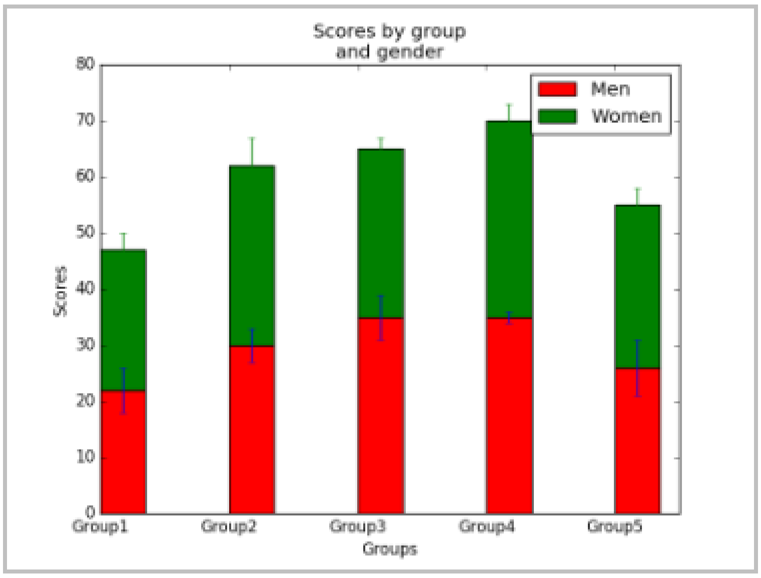
Program:

```
import numpy as np

import matplotlib.pyplot as plt

n_groups = 5

men_means = (22, 30, 33, 30, 26)

women_means = (25, 32, 30, 35, 29)

fig, ax = plt.subplots()

index = np.arange(n_groups)

bar_width = 0.35

opacity = 0.8

rects1 = plt.bar(index, men_means, bar_width,

alpha=opacity,

color='g',

label='Men')

rects2 = plt.bar(index + bar_width, women_means, bar_width,
alpha=opacity,

color='r',

label='Women')

plt.xlabel('Person')

plt.ylabel('Scores')

plt.title('Scores by person')

plt.xticks(index + bar_width, ('G1', 'G2', 'G3', 'G4', 'G5'))

plt.legend()

plt.tight_layout()

plt.show()
```

Output:



31.Write a Python program to create a stacked bar plot with error bars.
Note: Use bottom to stack the women?s bars on top of the men?s bars.
Sample Data:
Means (men) = (22, 30, 35, 35, 26)
Means (women) = (25, 32, 30, 35, 29)
Men Standard deviation = (4, 3, 4, 1, 5)
Women Standard deviation = (3, 5, 2, 3, 3)

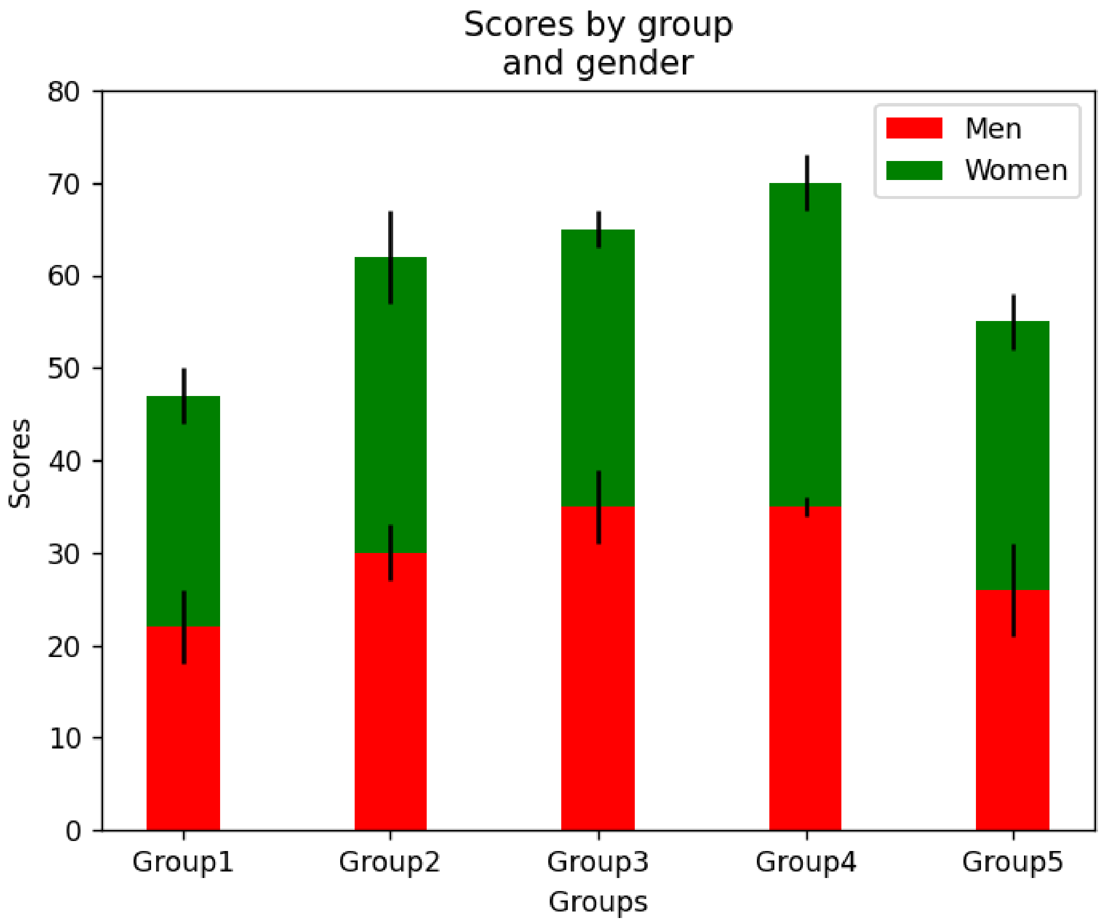


Program:

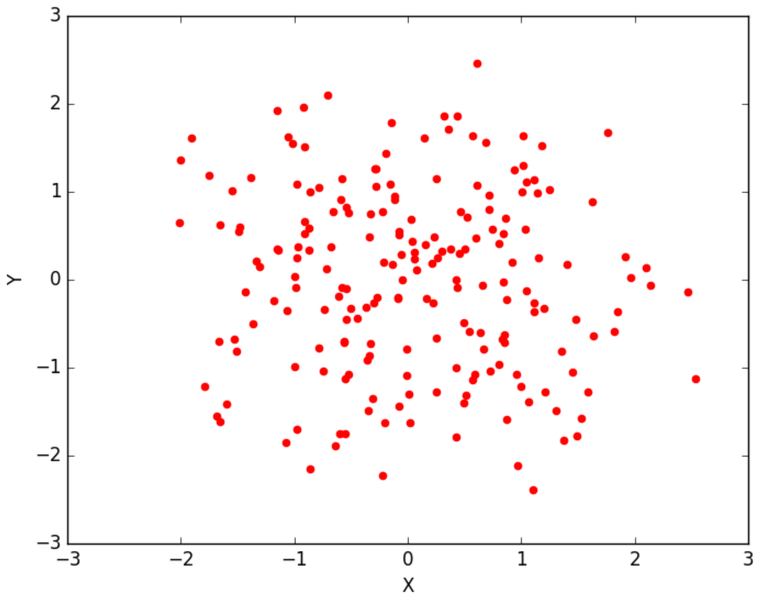
```
import numpy as np

import matplotlib.pyplot as plt

N = 5

menMeans = (22, 30, 35, 35, 26)

womenMeans = (25, 32, 30, 35, 29)

menStd = (4, 3, 4, 1, 5)

womenStd = (3, 5, 2, 3, 3)

ind = np.arange(N)

width = 0.35
```

```
p1 = plt.bar(ind, menMeans, width, yerr=menStd, color='red')

p2 = plt.bar(ind, womenMeans, width,

bottom=menMeans, yerr=womenStd, color='green')

plt.ylabel('Scores')

plt.xlabel('Groups')

plt.title('Scores by group\n' + 'and gender')

plt.xticks(ind, ('Group1', 'Group2', 'Group3', 'Group4', 'Group5'))

plt.yticks(np.arange(0, 81, 10))

plt.legend((p1[0], p2[0]), ('Men', 'Women'))

plt.show()
```

Output:



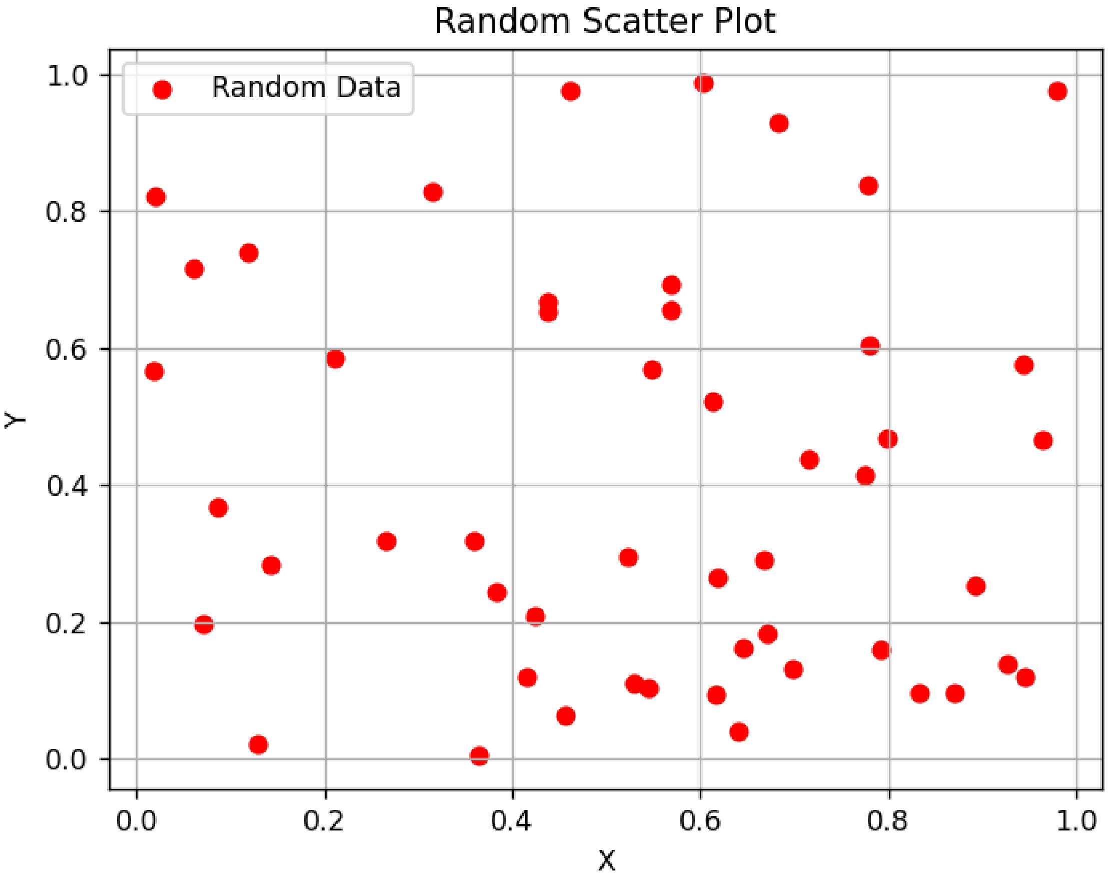
32.Write a Python program to draw a scatter graph taking a random distribution in X and Y and plotted against each other.



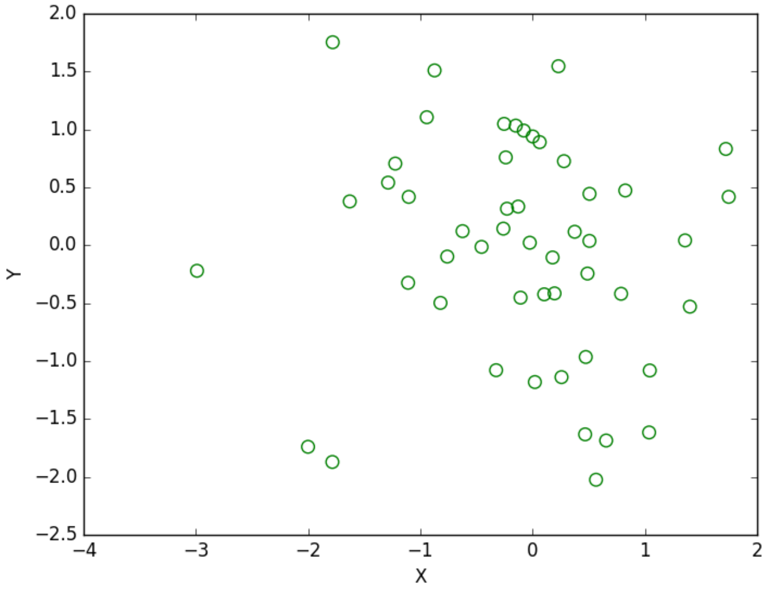
Program:

```
import matplotlib.pyplot as plt

import numpy as np

np.random.seed(0)

n_points = 50

x = np.random.rand(n_points)

y = np.random.rand(n_points)

plt.scatter(x, y, label='Random Data', color='red', marker='o')

plt.xlabel('X')

plt.ylabel('Y')

plt.title('Random Scatter Plot')

plt.legend()
plt.grid(True)

plt.show()
```

Output:



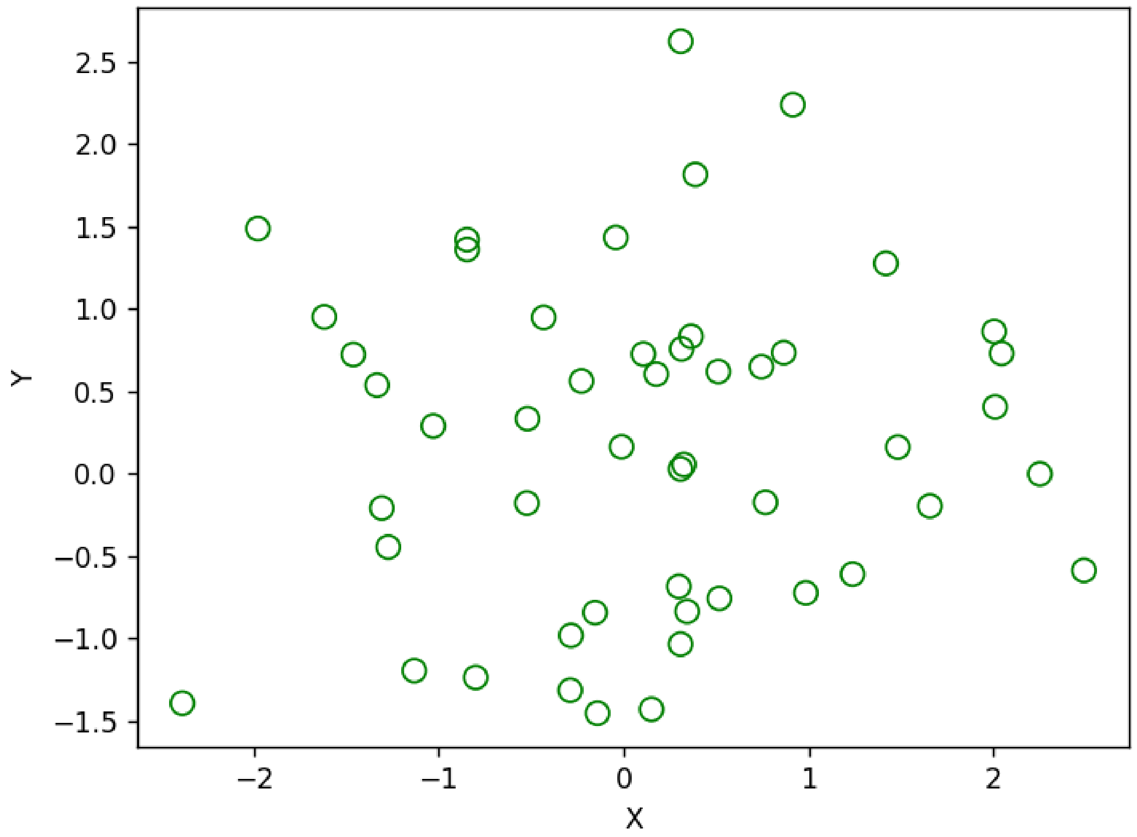
33.Write a Python program to draw a scatter plot with empty circles taking a random distribution in X and Y and plotted against each other.



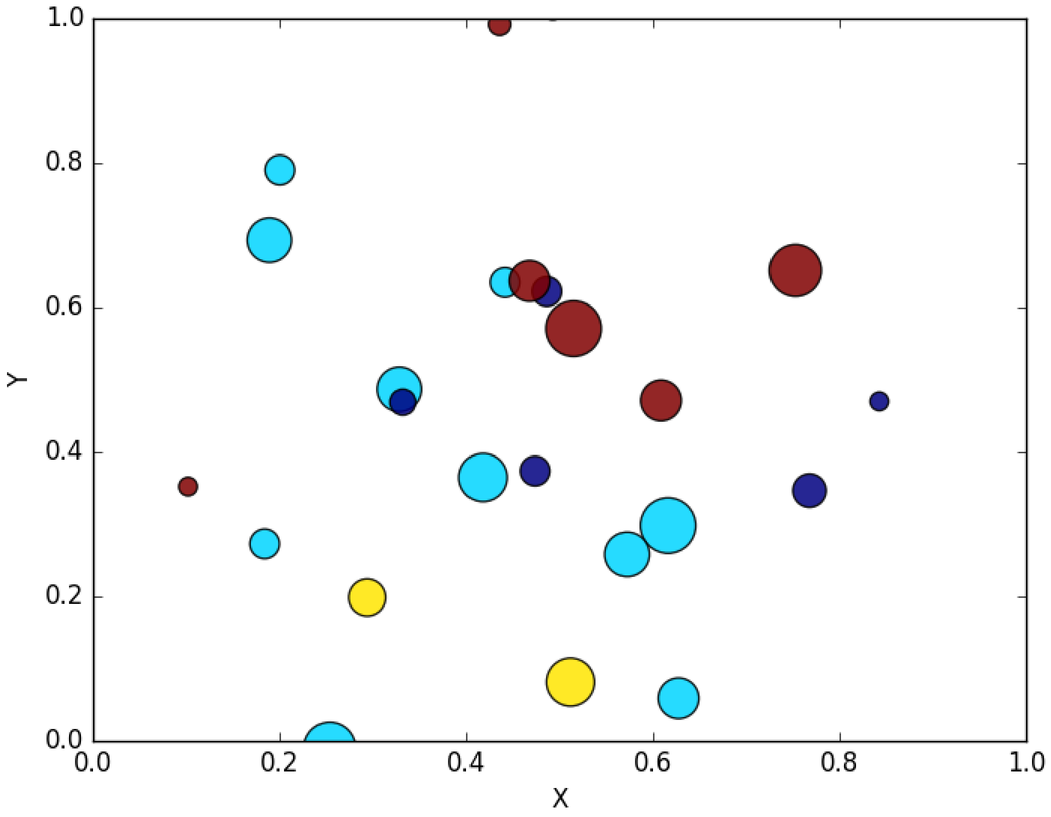
Program:

```
import matplotlib.pyplot as plt

import numpy as np

x = np.random.randn(50)

y = np.random.randn(50)

plt.scatter(x, y, s=70, facecolors='none', edgecolors='g')

plt.xlabel("X")

plt.ylabel("Y")

plt.show()
```

Output:



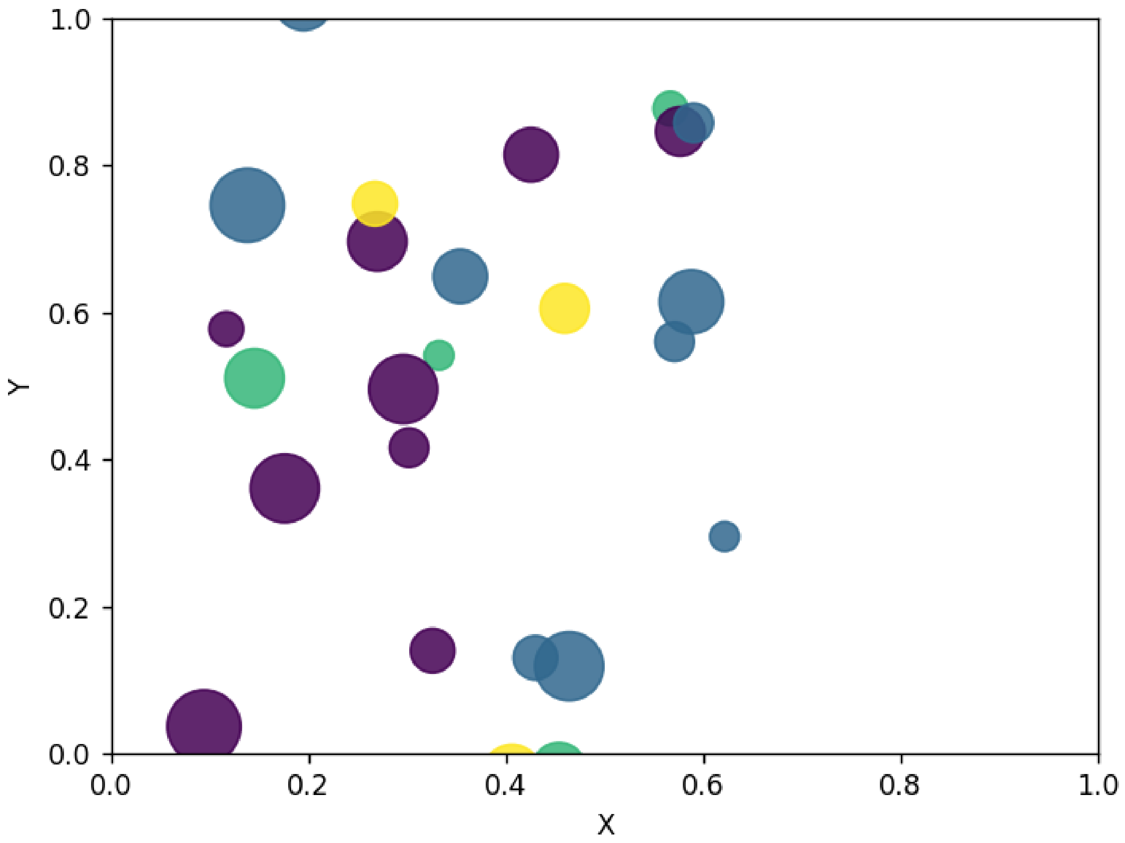
34.Write a Python program to draw a scatter plot using random distributions to generate balls of different sizes.



Program:

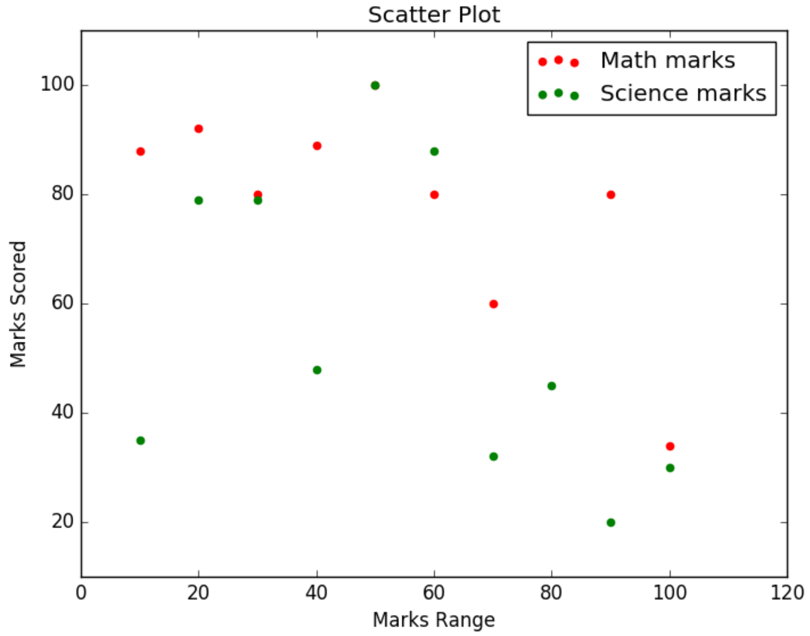
```
import math

import random

import matplotlib.pyplot as plt

no_of_balls = 25

x = [random.triangular() for i in range(no_of_balls)]

y = [random.gauss(0.5, 0.25) for i in range(no_of_balls)]

colors = [random.randint(1, 4) for i in range(no_of_balls)]

areas = [math.pi * random.randint(5, 15)**2 for i in range(no_of_balls)]

plt.figure()

plt.scatter(x, y, s=areas, c=colors, alpha=0.85)
plt.axis([0.0, 1.0, 0.0, 1.0])

plt.xlabel("X")

plt.ylabel("Y")

plt.show()
```

Output:



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

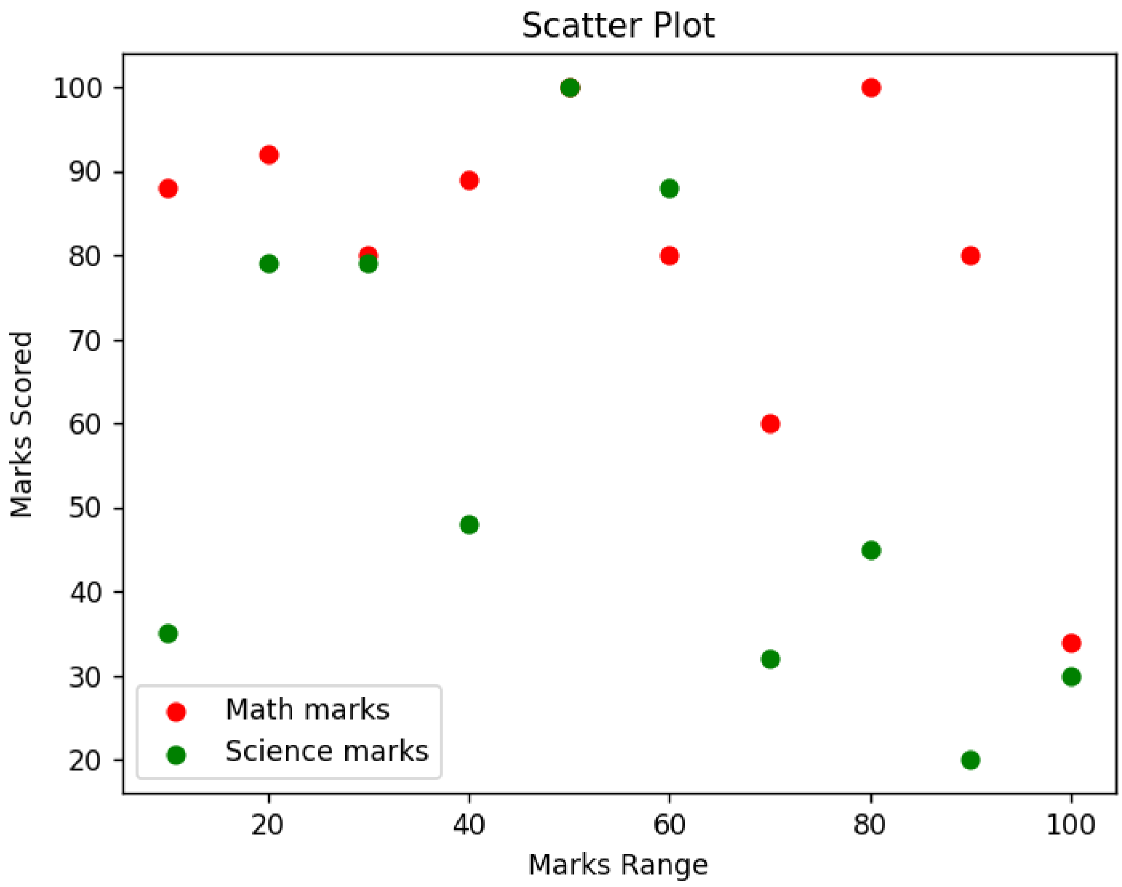
Test Data:
math_marks = [88, 92, 80, 89, 100, 80, 60, 100, 80, 34]
science_marks = [35, 79, 79, 48, 100, 88, 32, 45, 20, 30]
marks_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]



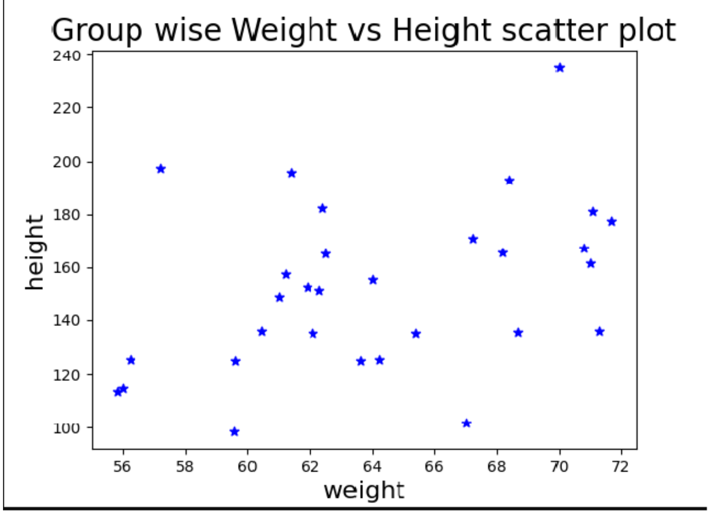
Program:

```
import matplotlib.pyplot as plt

import pandas as pd

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='r')

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()
```

Output:



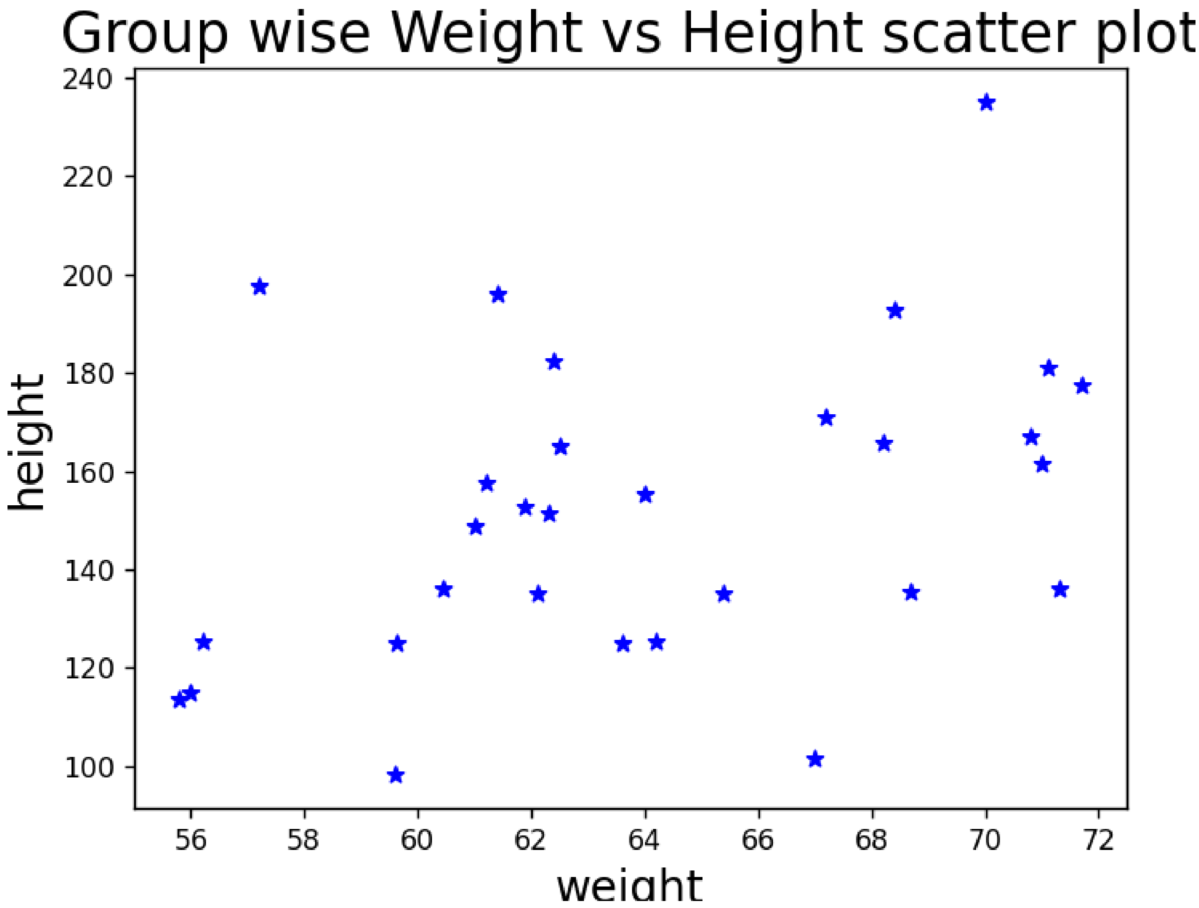
36.Write a Python program to draw a scatter plot for three different groups comparing weights and heights.



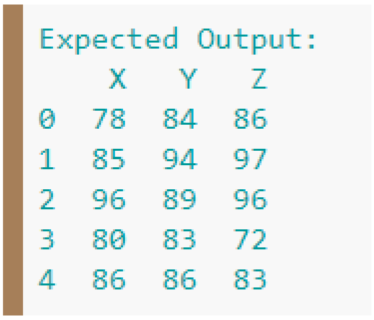
Program:

```
import matplotlib.pyplot as plt

import numpy as np

weight1=[67,57.2,59.6,59.64,55.8,61.2,60.45,61,56.23,56]

height1=[101.7,197.6,98.3,125.1,113.7,157.7,136,148.9,125.3,114.9]

weight2=[61.9,64,62.1,64.2,62.3,65.4,62.4,61.4,62.5,63.6]

height2=[152.8,155.3,135.1,125.2,151.3,135,182.2,195.9,165.1,125.1]

weight3=[68.2,67.2,68.4,68.7,71,71.3,70.8,70,71.1,71.7]

height3=[165.8,170.9,192.8,135.4,161.4,136.1,167.1,235.1,181.1,177.3]

weight=np.concatenate((weight1,weight2,weight3))

height=np.concatenate((height1,height2,height3))

plt.scatter(weight, height, marker='*', color=['blue'])

plt.xlabel('weight', fontsize=16)

plt.ylabel('height', fontsize=16)

plt.title('Group wise Weight vs Height scatter plot',fontsize=20)

plt.show()
```

Output:



37.Write a Pandas program to create a dataframe from a dictionary and display it.
Sample data: {'X':[78,85,96,80,86], 'Y':[84,94,89,83,86],'Z':[86,97,96,72,83]}



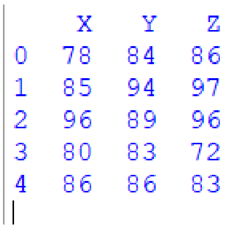
```
Expected Output:
    X   Y   Z
0  78  84  86
1  85  94  97
2  96  89  96
3  80  83  72
4  86  86  83
```

Program:

```
import pandas as pd

df = pd.DataFrame({'X':[78,85,96,80,86], 'Y':[84,94,89,83,86],'Z':[86,97,96,72,83]});
```

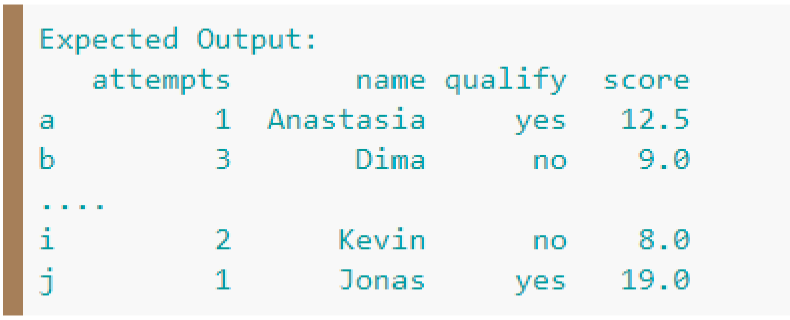
print(df)

Output:



```
    X   Y   Z
0  78  84  86
1  85  94  97
2  96  89  96
3  80  83  72
4  86  86  83
```

38.Write a Pandas program to create and display a DataFrame from a specified dictionary data which has the index labels.
Sample Python dictionary data and list labels:
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']



```
Expected Output:
   attempts       name qualify  score
a         1  Anastasia     yes   12.5
b         3       Dima      no    9.0
....
i         2      Kevin      no    8.0
j         1      Jonas     yes   19.0
```

Program:

import pandas as pd

import numpy as np

exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'], 'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

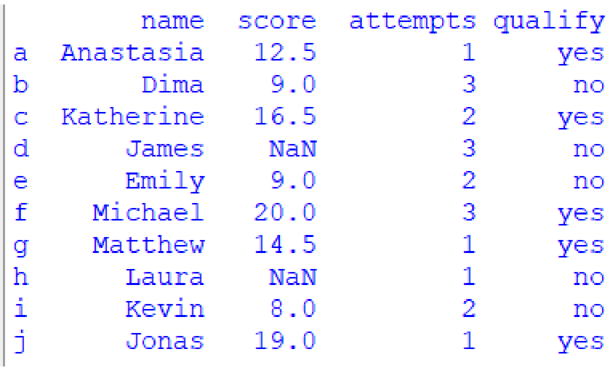
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)

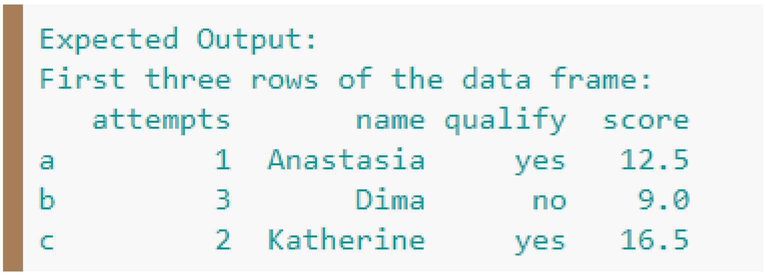
print(df)

Output:



```
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
d      James    NaN         3      no
e      Emily    9.0         2      no
f    Michael   20.0         3     yes
g    Matthew   14.5         1     yes
h      Laura    NaN         1      no
i      Kevin    8.0         2      no
j      Jonas   19.0         1     yes
```

39.Write a Pandas program to get the first 3 rows of a given DataFrame.
Sample Python dictionary data and list labels:
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

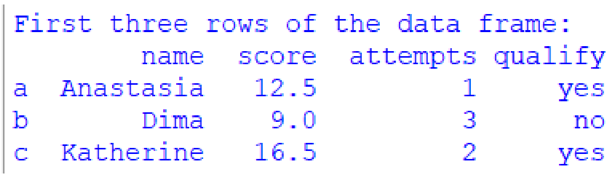


```
Expected Output:
First three rows of the data frame:
   attempts       name qualify  score
a         1  Anastasia     yes   12.5
b         3       Dima      no    9.0
c         2  Katherine     yes   16.5
```

Program:

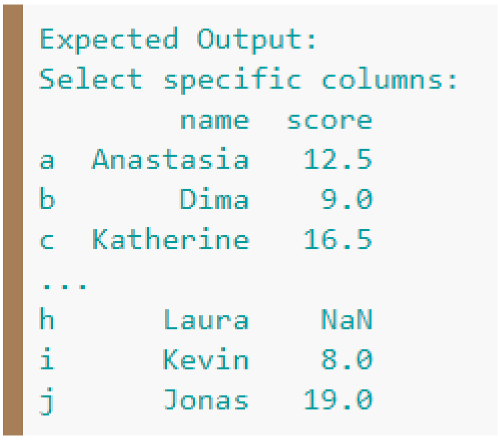
```
import pandas as pd

import numpy as np

exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael',
'Matthew', 'Laura', 'Kevin', 'Jonas'],

        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)

print("First three rows of the data frame:")

print(df.iloc[:3])
```

Output:



```
First three rows of the data frame:
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
```

40. Write a Pandas program to select the 'name' and 'score' columns from the following DataFrame.
Sample Python dictionary data and list labels:
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],
'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

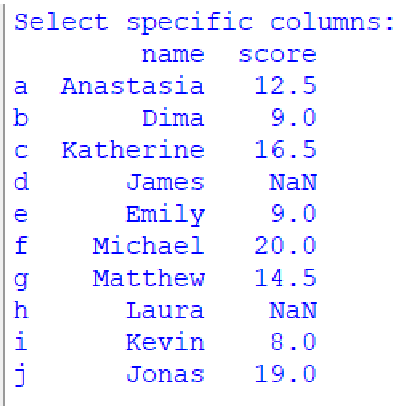


```
Expected Output:
Select specific columns:
        name  score
a  Anastasia   12.5
b       Dima    9.0
c  Katherine   16.5
...
h      Laura    NaN
i      Kevin    8.0
j      Jonas   19.0
```

Program:

```
import pandas as pd

import numpy as np

exam_data  = {'name': ['Anastasia', 'Dima', 'Katherine', 'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin', 'Jonas'],

        'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],

        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],

        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}

labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)

print("Select specific columns:")

print(df[['name', 'score']])
```

Output:



```
Select specific columns:
        name  score
a  Anastasia   12.5
b       Dima    9.0
c  Katherine   16.5
d      James    NaN
e      Emily    9.0
f    Michael   20.0
g    Matthew   14.5
h      Laura    NaN
i      Kevin    8.0
j      Jonas   19.0
```