192224089

November 30, 2023

[ ]:

```
#1.         Write a Pandas program to select distinct department id from␣
 ↪employees file.
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 = pd.DataFrame(data)


dist_id = employees['DEPARTMENT_ID'].unique()

# Create a new DataFrame with distinct department IDs
dist_df = pd.DataFrame({'DEPARTMENT_ID': dist_id})
```

```
# Display the result
print(dist_df)
```

```
    DEPARTMENT_ID
0              10
1              20
2              30
3              40
4              50
5              60
6              70
7              80
8              90
9             100
10            110
11            120
12            130
13            140
14            150
15            160
16            170
17            180
18            190
19            200
20            210
21            220
22            230
23            240
24            250
25            260
26            270
```

[ ]:

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

# Count the number of jobs each employee has done
job_counts = df.groupby('EMPLOYEE_ID')['JOB_ID'].count()

# Select employees who did two or more jobs
result = job_counts[job_counts >= 2]

# Display the employee IDs
print(result)
```

```
EMPLOYEE_ID
101    2
176    2
200    2
Name: JOB_ID, dtype: int64
```

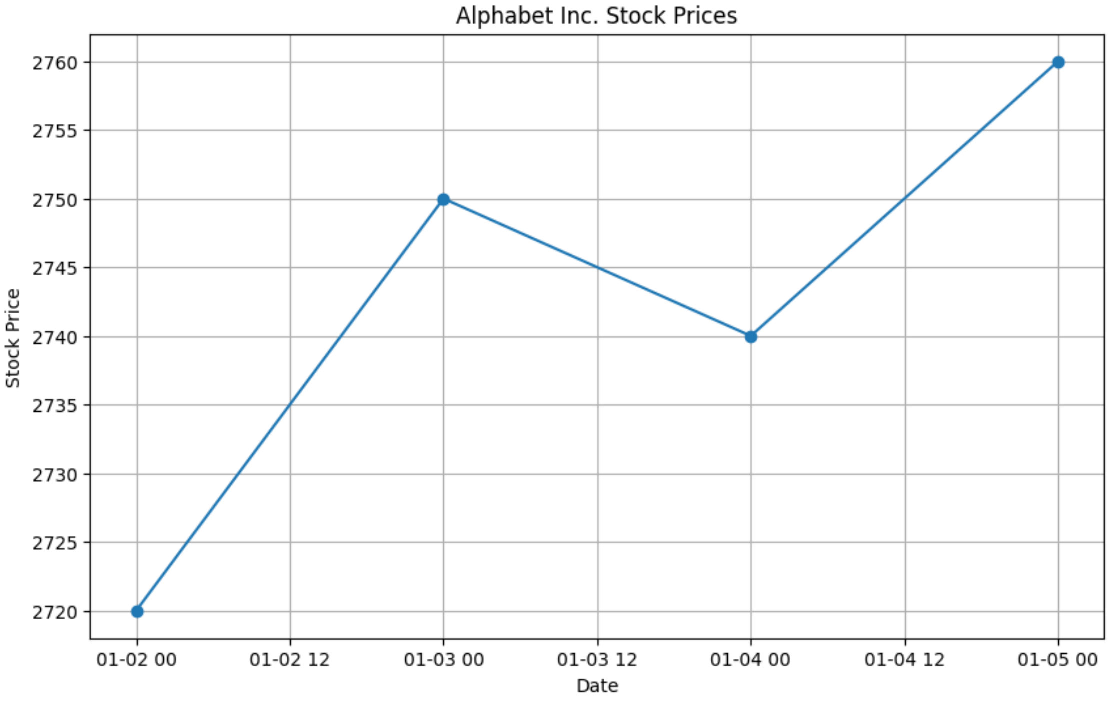
```
[ ]: #3.        Write a Pandas program to display the details of jobs in descending␣
      ↪sequence on job title.
     import pandas as pd

     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)

     # Sort the DataFrame by job title in descending order
     sorted_df = df.sort_values(by='JOB_TITLE', ascending=False)

     # Display the result
     print(sorted_df)
```

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

# Sample data for Alphabet Inc. stock prices with date as the index
data = {
    'Date': ['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04',␣
 ↪'2023-01-05', '2023-01-06'],
    'Price': [2700, 2720, 2750, 2740, 2760, 2770]
}

# Create a DataFrame from the sample data
df = pd.DataFrame(data)

# Convert the 'Date' column to a datetime format
df['Date'] = pd.to_datetime(df['Date'])

# Set 'Date' as the index
df.set_index('Date', inplace=True)

# Filter the data for the specific date range
start_date = '2023-01-02'
end_date = '2023-01-05'
filtered_df = df[start_date:end_date]
```

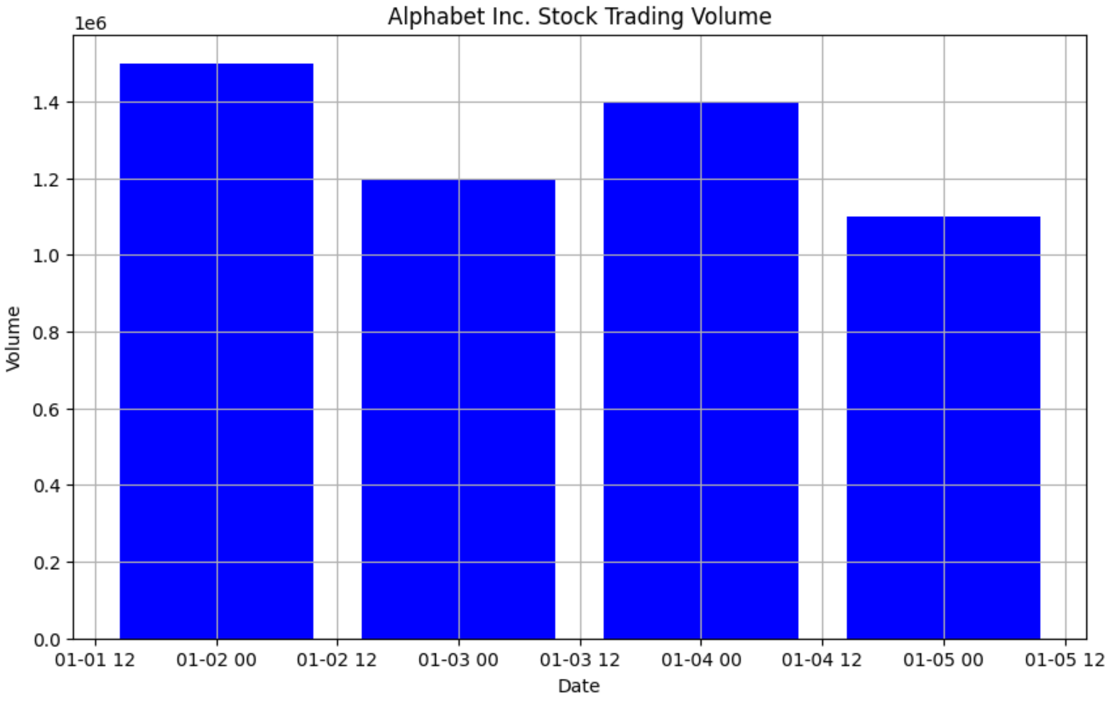
```
# Create a line plot
plt.figure(figsize=(10, 6))
plt.plot(filtered_df.index, filtered_df['Price'], marker='o', linestyle='-')
plt.title('Alphabet Inc. Stock Prices')
plt.xlabel('Date')
plt.ylabel('Stock Price')
plt.grid(True)

# Show the plot
plt.show()
```



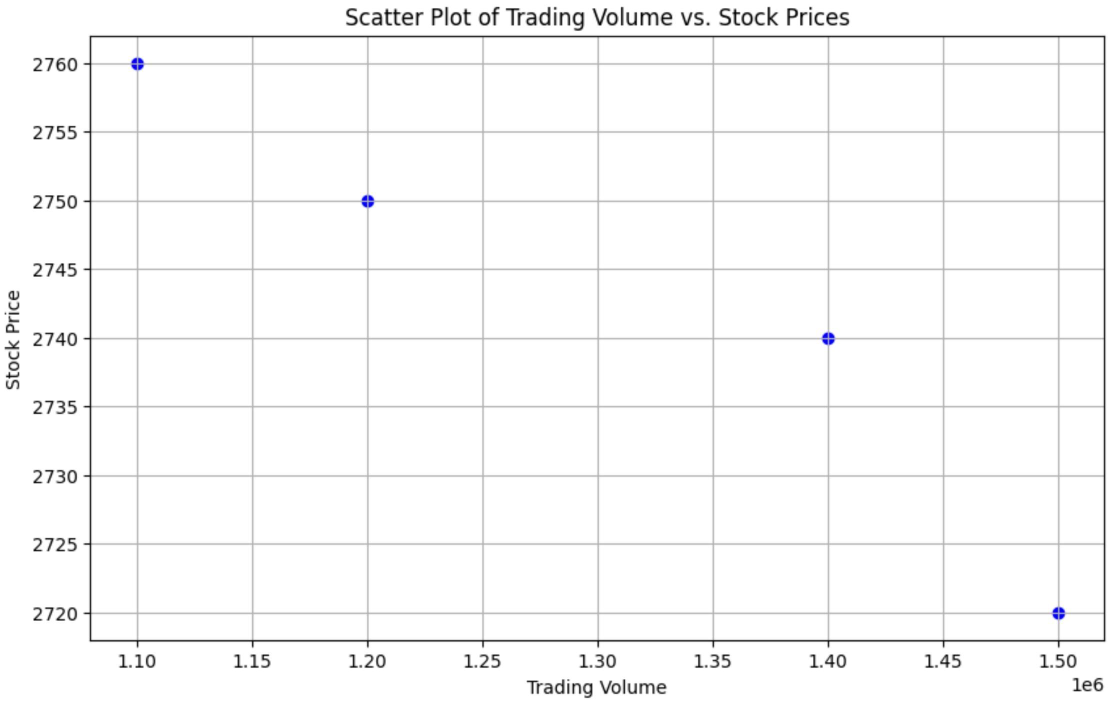
[ ]:

```
#5.        Write a Pandas program to create a bar plot of the trading volume of␣
 ↪Alphabet Inc. stock between two specific dates.
import pandas as pd
import matplotlib.pyplot as plt

# Sample data for Alphabet Inc. stock trading volume with date as the index
data = {
    'Date': ['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04',␣
 ↪'2023-01-05', '2023-01-06'],
    'Volume': [1000000, 1500000, 1200000, 1400000, 1100000, 1600000]
}

# Create a DataFrame from the sample data
```

```python
df = pd.DataFrame(data)

# Convert the 'Date' column to a datetime format
df['Date'] = pd.to_datetime(df['Date'])

# Set 'Date' as the index
df.set_index('Date', inplace=True)

# Filter the data for the specific date range
start_date = '2023-01-02'
end_date = '2023-01-05'
filtered_df = df[start_date:end_date]

# Create a bar plot
plt.figure(figsize=(10, 6))
plt.bar(filtered_df.index, filtered_df['Volume'], color='blue')
plt.title('Alphabet Inc. Stock Trading Volume')
plt.xlabel('Date')
plt.ylabel('Volume')
plt.grid(True)

# Show the plot
plt.show()
```



```
[ ]: #6.        Write a Pandas program to create a scatter plot of the trading␣
      ↪volume/stock prices of Alphabet Inc. stock between two specific dates.
     import pandas as pd
     import matplotlib.pyplot as plt

     # Sample data for Alphabet Inc. stock including Date, Volume, and Price columns
     data = {
         'Date': ['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04',␣
      ↪'2023-01-05', '2023-01-06'],
         'Volume': [1000000, 1500000, 1200000, 1400000, 1100000, 1600000],
         'Price': [2700, 2720, 2750, 2740, 2760, 2770]
     }

     # Create a DataFrame from the sample data
     alphabet_stock_data = pd.DataFrame(data)

     # Convert the 'Date' column to a datetime format
     alphabet_stock_data['Date'] = pd.to_datetime(alphabet_stock_data['Date'])

     # Set 'Date' as the index
     alphabet_stock_data.set_index('Date', inplace=True)

     # Filter the data for the specific date range
     start_date = '2023-01-02'
     end_date = '2023-01-05'
     filtered_data = alphabet_stock_data[start_date:end_date]

     # Create a scatter plot
     plt.figure(figsize=(10, 6))
     plt.scatter(filtered_data['Volume'], filtered_data['Price'], color='blue',␣
      ↪marker='o')
     plt.title('Scatter Plot of Trading Volume vs. Stock Prices')
     plt.xlabel('Trading Volume')
     plt.ylabel('Stock Price')
     plt.grid(True)

     # Show the plot
     plt.show()
```



```
[ ]: #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)
import pandas as pd

# Sample sales_data table
data = {
    'Item': ['A', 'B', 'A', 'C', 'B', 'A', 'C', 'B', 'A', 'C'],
    'Sale': [100, 200, 150, 300, 250, 180, 320, 220, 170, 310]
}

# Create a DataFrame from the sample data
sales_data = pd.DataFrame(data)

# Create a pivot table to find the maximum and minimum sale values for each item
pivot_table = pd.pivot_table(sales_data, values='Sale', index='Item', aggfunc=[max, min])

# Rename the columns for clarity
pivot_table.columns = ['Max Sale', 'Min Sale']

# Display the pivot table
print(pivot_table)
```

```
     Max Sale  Min Sale
```

```
Item
A           180        100
B           250        200
C           320        300
```

[ ]:

```
#8.        Write a Pandas program to create a Pivot table and find the item␣
 ↪wise unit sold. .(refer sales_data table)
import pandas as pd

# Sample sales_data table
data = {
    'Item': ['A', 'B', 'A', 'C', 'B', 'A', 'C', 'B', 'A', 'C'],
    'Unit Sold': [5, 10, 8, 12, 15, 9, 16, 11, 7, 13]
}

# Create a DataFrame from the sample data
sales_data = pd.DataFrame(data)

# Create a pivot table to find the total unit sold for each item
pivot_table = pd.pivot_table(sales_data, values='Unit Sold', index='Item',␣
 ↪aggfunc='sum')

# Display the pivot table
print(pivot_table)
```

```
      Unit Sold
Item
A            29
B            36
C            41
```

[ ]:

```
#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)
import pandas as pd

# Create the Sales_data DataFrame
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', 'East',␣
 ↪'Central', 'East', 'Central', 'East'],
    'Manager': ['Martha', 'Hermann', 'Hermann', 'Timothy', 'Timothy', 'Martha',␣
 ↪'Martha', 'Hermann', 'Douglas', 'Martha', 'Hermann', 'Martha', 'Douglas',␣
 ↪'Martha', 'Douglas', '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)

# Create a pivot table to find total sale amount region-wise, manager-wise, and␣
 ↪salesman-wise
pivot_table = pd.pivot_table(sales_data, values='Sale_amt', index=['Region',␣
 ↪'Manager', 'SalesMan'], aggfunc='sum')

print(pivot_table)
```

```
                           Sale_amt
Region  Manager SalesMan
Central Douglas John          250.0
        Hermann Luis       150948.0
                Shelli      25000.0
                Sigal      121820.0
        Martha  Steven      89850.0
        Timothy David        6075.0
East    Douglas Karen       40500.0
        Martha  Alexander  231076.0
                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.
import pandas as pd
import numpy as np
```

```
# Create a DataFrame with random values
data = np.random.randn(10, 4)

df = pd.DataFrame(data, columns=['Column1', 'Column2', 'Column3', 'Column4'])

# Apply conditional formatting to highlight negative numbers in red and␣
↪positive numbers in green
styled_df = df.style.applymap(lambda val: f'color: {"red" if val < 0 else␣
↪"black"}')

# Display the styled DataFrame
styled_df
```

```
[ ]: <pandas.io.formats.style.Styler at 0x78c0865aba60>
```

[ ]:

```
#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.
import pandas as pd
import numpy as np

# Create a DataFrame with random values
data = np.random.randn(10, 4)
df = pd.DataFrame(data, columns=['Column1', 'Column2', 'Column3', 'Column4'])

# Convert some random values to NaN
rows, cols = np.random.choice(10, size=5), np.random.choice(4, size=5)
df.iloc[rows, cols] = np.nan

# Apply conditional formatting to highlight NaN values
styled_df = df.style.applymap(lambda val: f'background-color: red' if pd.
↪isna(val) else '')

# Display the styled DataFrame
styled_df
```

```
[ ]: <pandas.io.formats.style.Styler at 0x78c08743caf0>
```

[ ]:

```
#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.

import pandas as pd
import numpy as np

# Create a DataFrame with random values
data = np.random.randn(10, 4)
df = pd.DataFrame(data, columns=['Column1', 'Column2', 'Column3', 'Column4'])
```

```
# Create a Styler object to set the background color to black and font color to␣
↪yellow
styled_df = df.style.set_properties(**{'background-color': 'black', 'color':␣
↪'yellow'})

# Display the styled DataFrame
styled_df
```

[ ]: <pandas.io.formats.style.Styler at 0x78c0582230d0>

[ ]:
```
#13.Write a Pandas program to detect missing values of a given DataFrame.␣
↪Display True or False.
import pandas as pd
import numpy as np

# Create a DataFrame with missing values
data = {
    'A': [1, 2, np.nan, 4, 5],
    'B': [np.nan, 2, 3, 4, np.nan],
    'C': [1, 2, 3, np.nan, 5]
}

df = pd.DataFrame(data)

# Detect missing values and display True for missing values, and False for␣
↪non-missing values
missing_values = df.isnull()

# Display the result
print(missing_values)
```

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

[ ]:
```
#14. Write a Pandas program to find and replace the missing values in a given␣
↪DataFrame which do not have any valuable information.
import pandas as pd
import numpy as np

# Create a DataFrame with missing values
data = {
    'A': [1, 2, np.nan, 4, 5],
    'B': [np.nan, 2, 3, 4, np.nan],
    'C': [1, 2, 3, np.nan, 5]
}

df = pd.DataFrame(data)

# Replace missing values with a specific value (e.g., -1) that does not have valuable information
value_to_replace = -1
df_filled = df.fillna(value_to_replace)

# Display the DataFrame with missing values replaced
print(df_filled)
```

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

[ ]:

```
#15. Write a Pandas program to keep the rows with at least 2 NaN values in a given DataFrame.
import pandas as pd
import numpy as np

# Create a sample DataFrame
data = {'A': [1, 2, np.nan, 4, 5],
        'B': [np.nan, 2, 3, np.nan, 5],
        'C': [1, 2, np.nan, np.nan, 5]}

df = pd.DataFrame(data)

# Keep rows with at least 2 NaN values
result = df[df.isna().sum(axis=1) >= 2]

# Display the resulting DataFrame
print(result)
```

```
     A    B   C
2  NaN  3.0 NaN
3  4.0  NaN NaN
```

[ ]:

```
#16.Write a Pandas program to split the following dataframe into groups based on school code. Also check the type of GroupBy object.
import pandas as pd
```

```
# Create a sample DataFrame
data = {'Student': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
        'School Code': [101, 102, 101, 103, 102],
        'Grade': [85, 92, 78, 88, 95]}

df = pd.DataFrame(data)

# Group the DataFrame by 'School Code'
grouped = df.groupby('School Code')

# Check the type of GroupBy object
print(type(grouped))
```

```
<class 'pandas.core.groupby.generic.DataFrameGroupBy'>
```

```
[ ]: #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.
     import pandas as pd

     # Create a sample DataFrame
     data = {'Student': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
             'School Code': [101, 102, 101, 103, 102],
             'Age': [15, 17, 14, 16, 18]}

     df = pd.DataFrame(data)

     # Split the DataFrame by 'School Code' and calculate statistics
     grouped = df.groupby('School Code')['Age'].agg(['mean', 'min', 'max'])

     # Reset the index for a cleaner output
     grouped.reset_index(inplace=True)

     print(grouped)
```

```
   School Code  mean  min  max
0          101  14.5   14   15
1          102  17.5   17   18
2          103  16.0   16   16
```

```
[ ]: #18.Write a Pandas program to split the following given dataframe into groups␣
      ↪based on school code and class.
     import pandas as pd

     # Create a sample DataFrame
     data = {'Student': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
             'School Code': [101, 102, 101, 103, 102],
             'Class': ['A', 'B', 'A', 'C', 'B'],
             'Age': [15, 17, 14, 16, 18]}

     df = pd.DataFrame(data)

     # Split the DataFrame by 'School Code' and 'Class'
     grouped = df.groupby(['School Code', 'Class'])

     for (school_code, class_), group in grouped:
         print(f"School Code: {school_code}, Class: {class_}")
         print(group)
         print()
```

```
School Code: 101, Class: A
   Student  School Code Class   Age
0    Alice          101     A   15
2  Charlie          101     A   14

School Code: 102, Class: B
  Student  School Code Class   Age
1      Bob          102     B   17
4      Eva          102     B   18

School Code: 103, Class: C
  Student  School Code Class   Age
3    David          103     C   16
```

[ ]:

```
#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.
import pandas as pd

# Sample data for illustration
data = {
    'Country': ['USA', 'Canada', 'UK', 'Australia', 'Germany'],
    'Beer': [25, 20, 15, 22, 18],
    'Spirit': [10, 8, 7, 12, 9],
    'Wine': [5, 7, 8, 4, 7],
}

# Create a DataFrame
df = pd.DataFrame(data)

# Display the dimensions (shape) of the dataset
dimensions = df.shape
print(f"Dimensions of the dataset: {dimensions}")

# Extract the column names
```

```
column_names = df.columns
print("Column names:")
for column in column_names:
    print(column)
```

```
Dimensions of the dataset: (5, 4)
Column names:
Country
Beer
Spirit
Wine
```

[ ]:

```
#20. Write a Pandas program to find the index of a given substring of a␣
 ↪DataFrame column.
import pandas as pd

# Sample data for illustration
data = {
    'Text': ['Hello, world', 'This is a test', 'Pandas is great', 'DataFrames␣
 ↪are useful']
}

# Create a DataFrame
df = pd.DataFrame(data)

# Find the index of the substring 'is' in the 'Text' column
substring = 'is'
result = df['Text'].str.find(substring)

# Display the result
print("Index of substring 'is' in the 'Text' column:")
print(result)
```

```
Index of substring 'is' in the 'Text' column:
0   -1
1    2
2    7
3   -1
Name: Text, dtype: int64
```

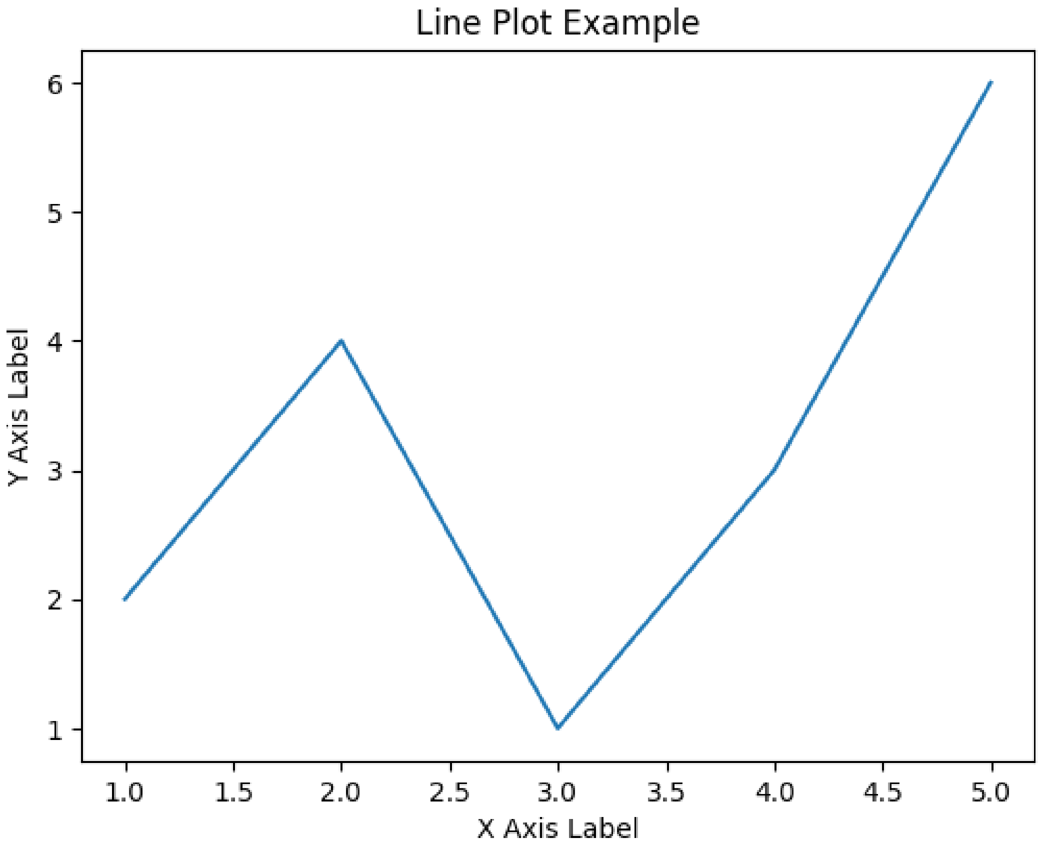
[ ]:

```
#21. Write a Pandas program to swap the cases of a specified character column␣
 ↪in a given DataFrame.
import pandas as pd

# Sample data for illustration
data = {
    'Text': ['Hello, World', 'This is a Test', 'Pandas is Great', 'DataFrames␣
  ↪are Useful']
}

# Create a DataFrame
df = pd.DataFrame(data)

# Specify the column to swap cases
column_to_swap = 'Text'

# Swap the cases in the specified column
df[column_to_swap] = df[column_to_swap].str.swapcase()

# Display the updated DataFrame
print("DataFrame with Cases Swapped:")
print(df)
```

```
DataFrame with Cases Swapped:
                    Text
0           hELLO, wORLD
1         tHIS IS A tEST
2        pANDAS IS gREAT
3   dATAfRAMES ARE uSEFUL
```

[ ]:

```
#22.Write a Python program to draw a line with suitable label in the x axis, y␣
  ↪axis and a title.
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4, 5]
y = [2, 4, 1, 3, 6]

# Create a line plot
plt.plot(x, y)

# Add labels for x and y axes
plt.xlabel("X Axis Label")
plt.ylabel("Y Axis Label")

# Add a title
plt.title("Line Plot Example")

# Display the plot
plt.show()
```



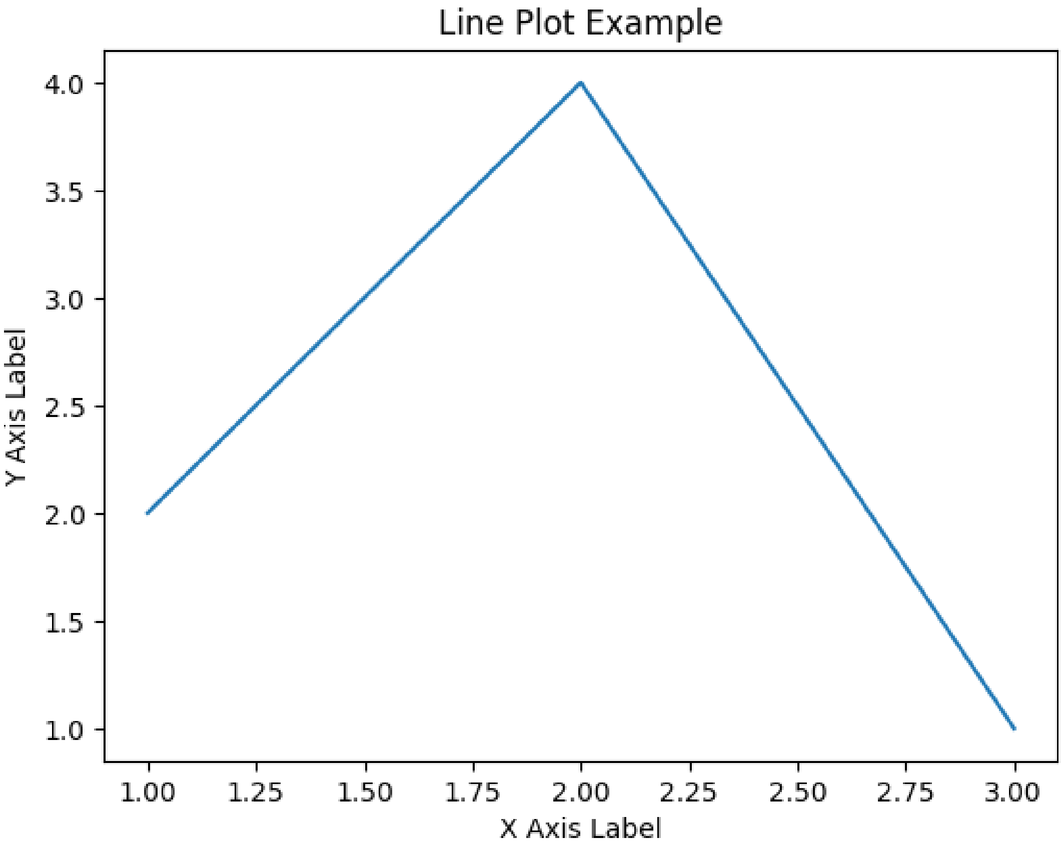
[ ]:

```
#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.
#file:
#1 2
#2 4
#3 1

import matplotlib.pyplot as plt
from google.colab import files

# Upload the text file with data
uploaded = files.upload()

# Assuming you've uploaded the file as 'test.txt'
file_name = 'test.txt'

# Read the data from the uploaded file
with open(file_name, 'r') as file:
    lines = file.readlines()
```

```python
# Extract x and y values from the text file
x = []
y = []
for line in lines:
    values = line.split()
    x.append(float(values[0]))
    y.append(float(values[1]))

# Create a line plot
plt.plot(x, y)

# Add labels for x and y axes
plt.xlabel("X Axis Label")
plt.ylabel("Y Axis Label")

# Add a title
plt.title("Line Plot Example")

# Display the plot
plt.show()
```

```
<IPython.core.display.HTML object>
```

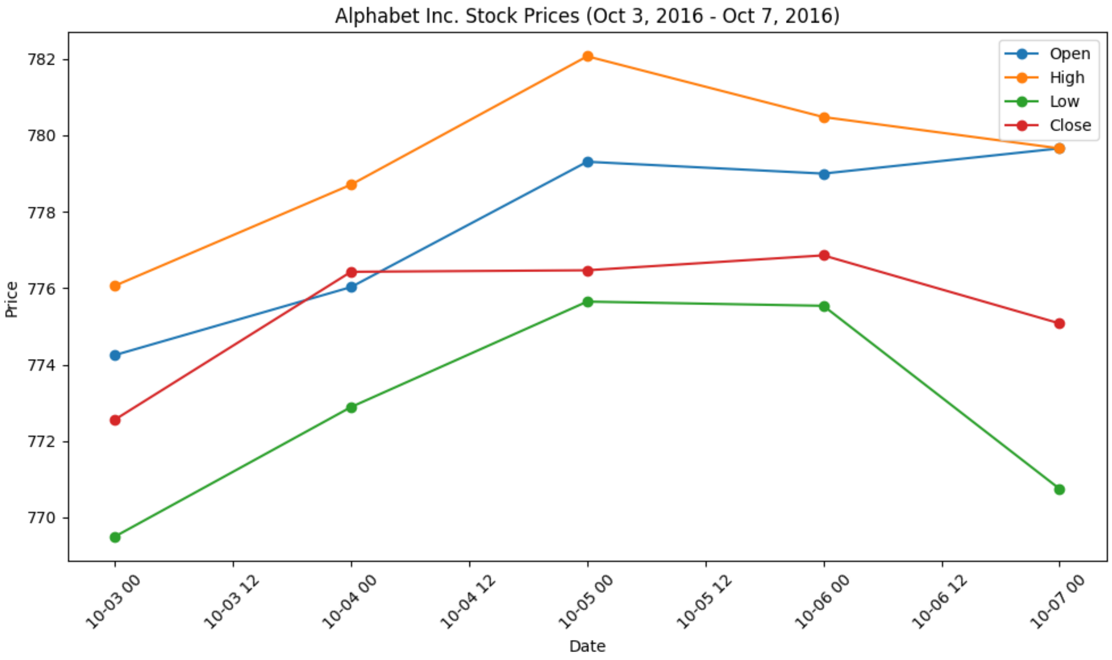
```
Saving test.txt to test.txt
```



[ ]:

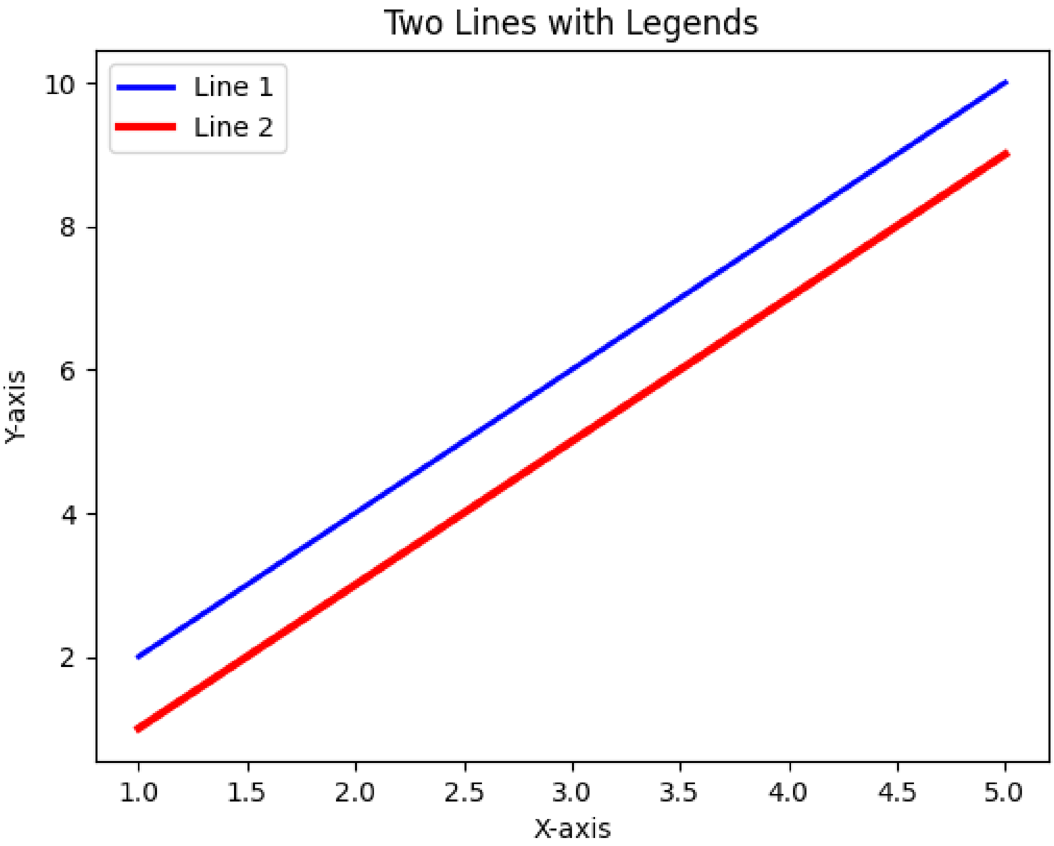
```
#24. Write a Python program to draw line charts of the financial data of␣
 ↪Alphabet Inc. between October 3, 2016 to October 7, 2016.
import pandas as pd
import matplotlib.pyplot as plt

# Sample financial data
data = {
    'Date': ['10-03-16', '10-04-16', '10-05-16', '10-06-16', '10-07-16'],
    'Open': [774.25, 776.030029, 779.309998, 779, 779.659973],
    'High': [776.065002, 778.710022, 782.070007, 780.47998, 779.659973],
    'Low': [769.5, 772.890015, 775.650024, 775.539978, 770.75],
    'Close': [772.559998, 776.429993, 776.469971, 776.859985, 775.080017]
}

# Create a DataFrame from the data
df = pd.DataFrame(data)

# Convert the 'Date' column to datetime format
df['Date'] = pd.to_datetime(df['Date'])
```

```python
# Set the 'Date' column as the index
df.set_index('Date', inplace=True)

# Create line charts for Open, High, Low, and Close prices
plt.figure(figsize=(10, 6))
plt.plot(df.index, df['Open'], label='Open', marker='o')
plt.plot(df.index, df['High'], label='High', marker='o')
plt.plot(df.index, df['Low'], label='Low', marker='o')
plt.plot(df.index, df['Close'], label='Close', marker='o')

# Add labels and title
plt.xlabel('Date')
plt.ylabel('Price')
plt.title('Alphabet Inc. Stock Prices (Oct 3, 2016 - Oct 7, 2016)')

# Rotate x-axis labels for better visibility
plt.xticks(rotation=45)

# Add a legend
plt.legend()

# Show the plot
plt.tight_layout()
plt.show()
```

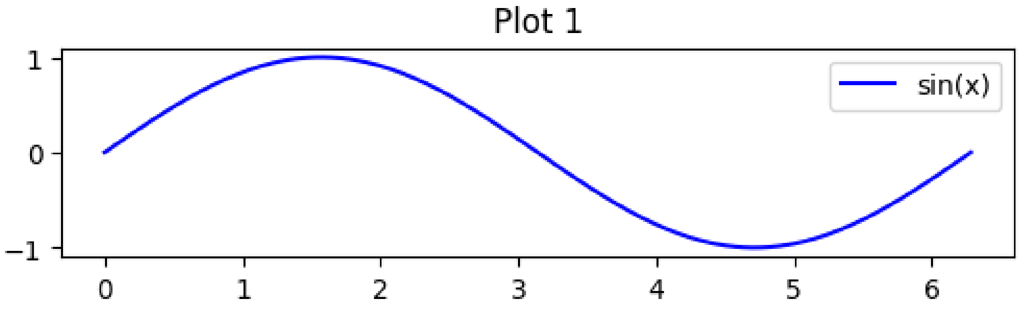


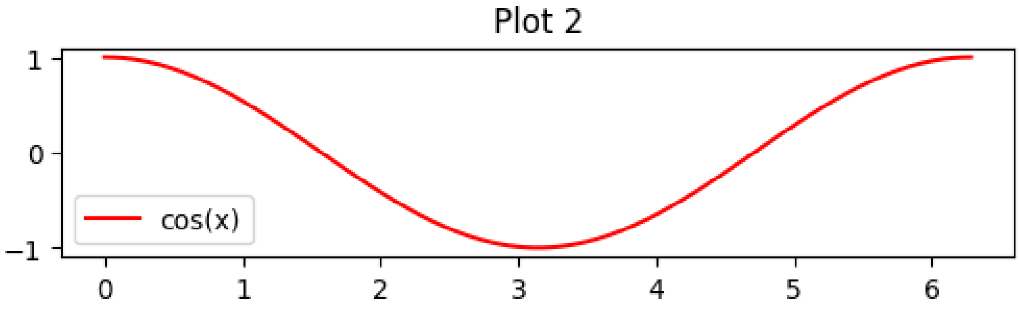
[ ]:

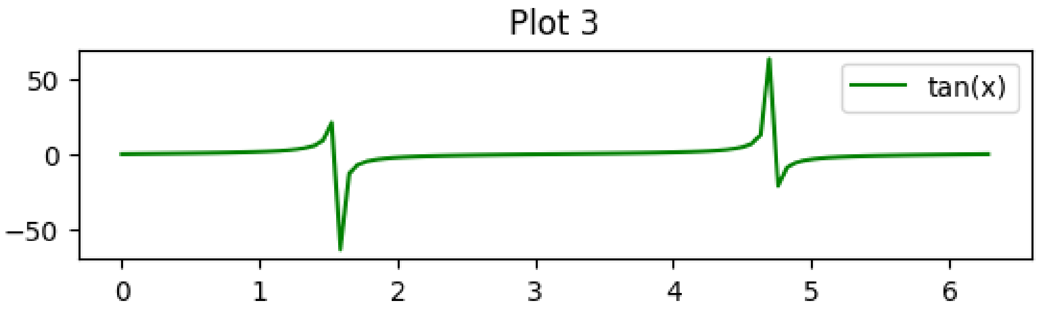
```python
#25.Write a Python program to plot two or more lines with legends, different widths and colors.
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4, 5]
y1 = [2, 4, 6, 8, 10]
y2 = [1, 3, 5, 7, 9]

# Plot two lines with legends, colors, and widths
plt.plot(x, y1, label='Line 1', color='blue', linewidth=2)
plt.plot(x, y2, label='Line 2', color='red', linewidth=3)

# Add labels and title
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Two Lines with Legends')

# Add a legend
plt.legend()

# Show the plot
plt.show()
```



```
[ ]: #26 Write a Python program to create multiple plots.
     import matplotlib.pyplot as plt
     import numpy as np

     # Create some sample data
     x = np.linspace(0, 2 * np.pi, 100)
     y1 = np.sin(x)
     y2 = np.cos(x)
     y3 = np.tan(x)

     # Create the first plot
     plt.figure(1)
     plt.subplot(311)
     plt.plot(x, y1, label='sin(x)', color='blue')
     plt.title('Plot 1')
     plt.legend()

     # Create the second plot
     plt.figure(2)
     plt.subplot(312)
     plt.plot(x, y2, label='cos(x)', color='red')
     plt.title('Plot 2')
     plt.legend()

     # Create the third plot
     plt.figure(3)
     plt.subplot(313)
     plt.plot(x, y3, label='tan(x)', color='green')
     plt.title('Plot 3')
     plt.legend()

     # Show the plots
     plt.show()
```

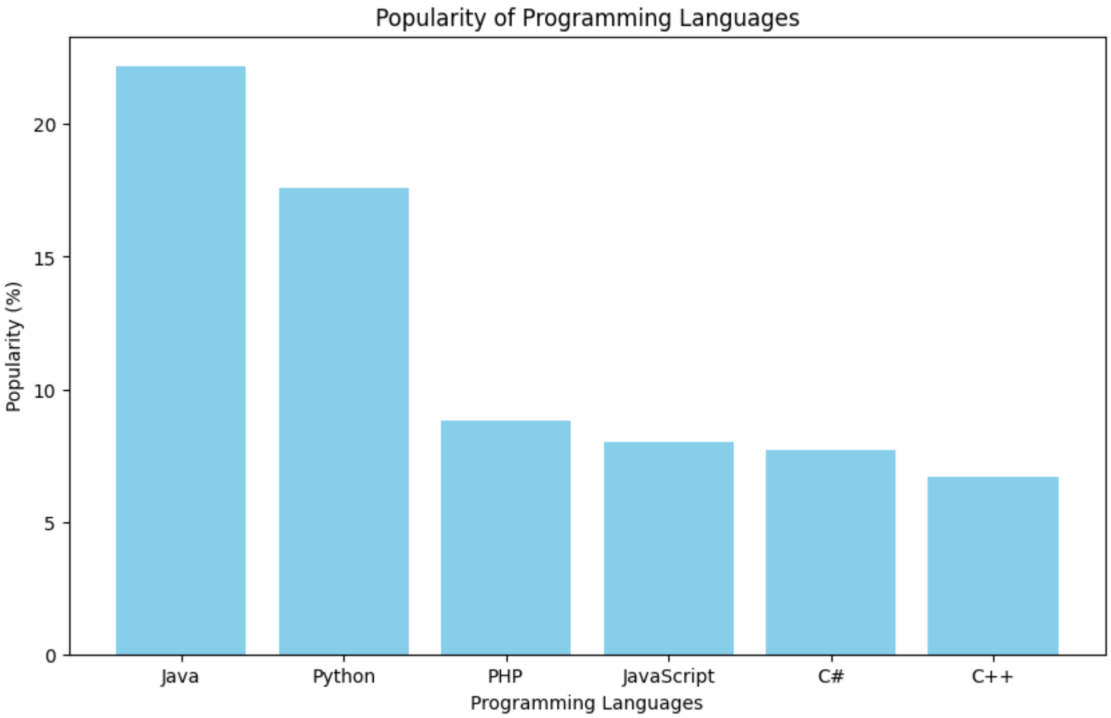






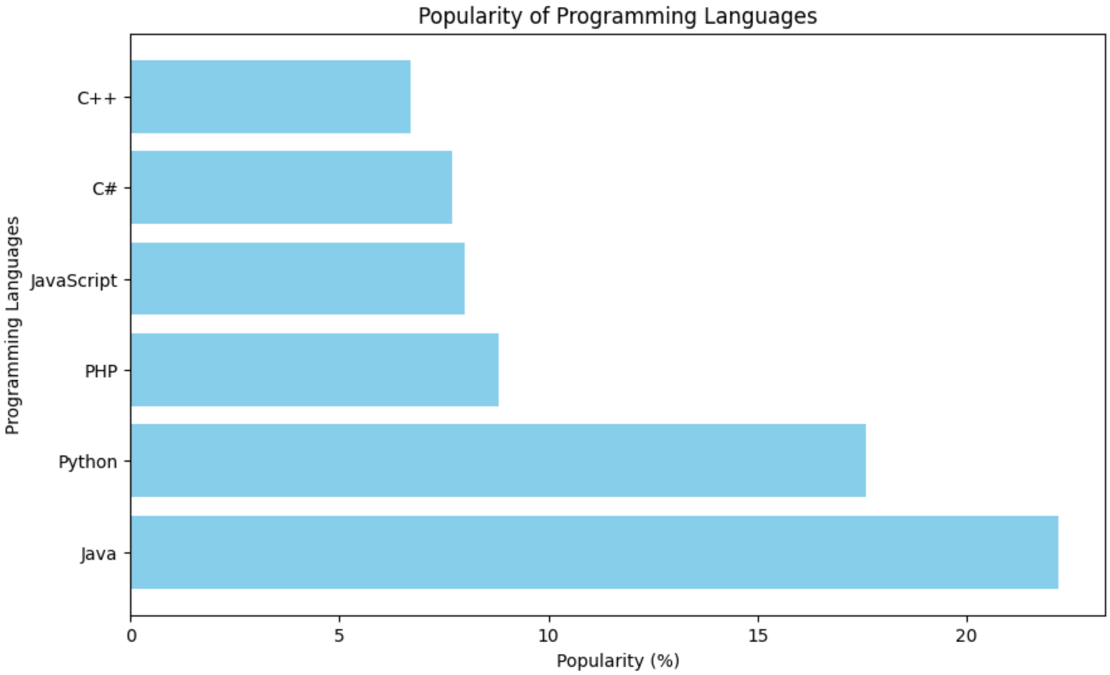
[ ]:

```
#27. Write a Python programming to display a bar chart of the popularity of programming Languages.
import matplotlib.pyplot as plt

# Sample data
languages = ["Java", "Python", "PHP", "JavaScript", "C#", "C++"]
popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]

# Create a bar chart
plt.figure(figsize=(10, 6))
plt.bar(languages, popularity, color='skyblue')

# Add labels and title
plt.xlabel("Programming Languages")
plt.ylabel("Popularity (%)")
plt.title("Popularity of Programming Languages")

# Show the bar chart
plt.show()
```

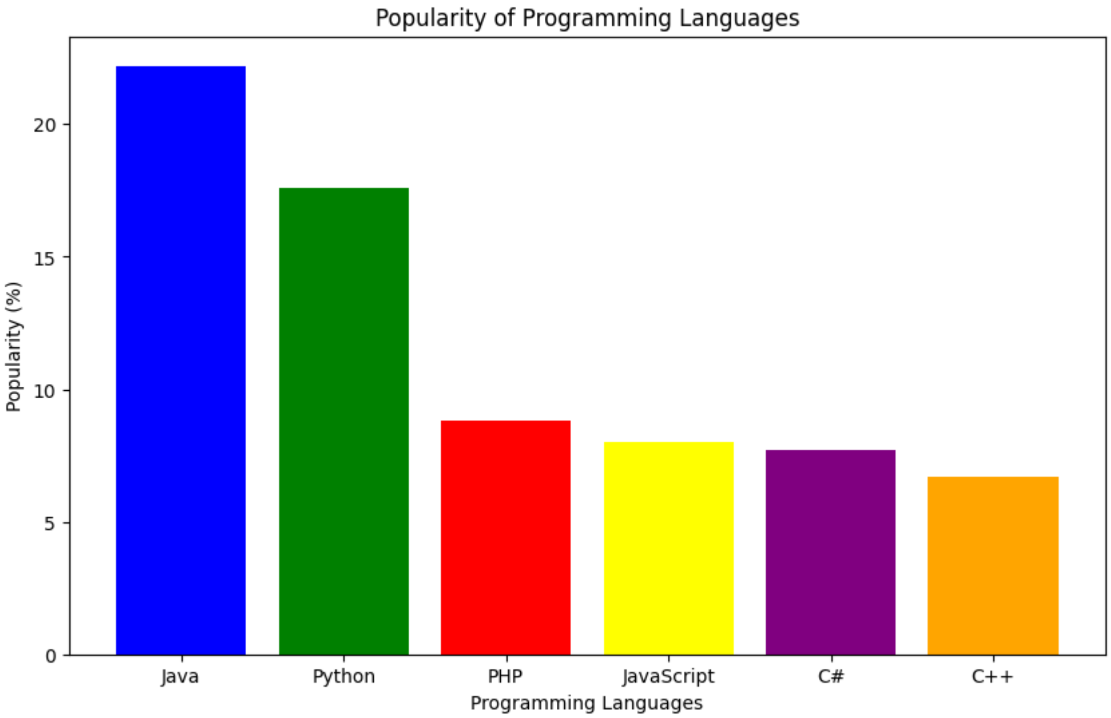


[ ]:

```python
#28.Write a Python programming to display a horizontal bar chart of the popularity of programming Languages.
import matplotlib.pyplot as plt

# Sample data
languages = ["Java", "Python", "PHP", "JavaScript", "C#", "C++"]
popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]

# Create a horizontal bar chart
plt.figure(figsize=(10, 6))
plt.barh(languages, popularity, color='skyblue')

# Add labels and title
plt.xlabel("Popularity (%)")
plt.ylabel("Programming Languages")
plt.title("Popularity of Programming Languages")

# Show the horizontal bar chart
plt.show()
```



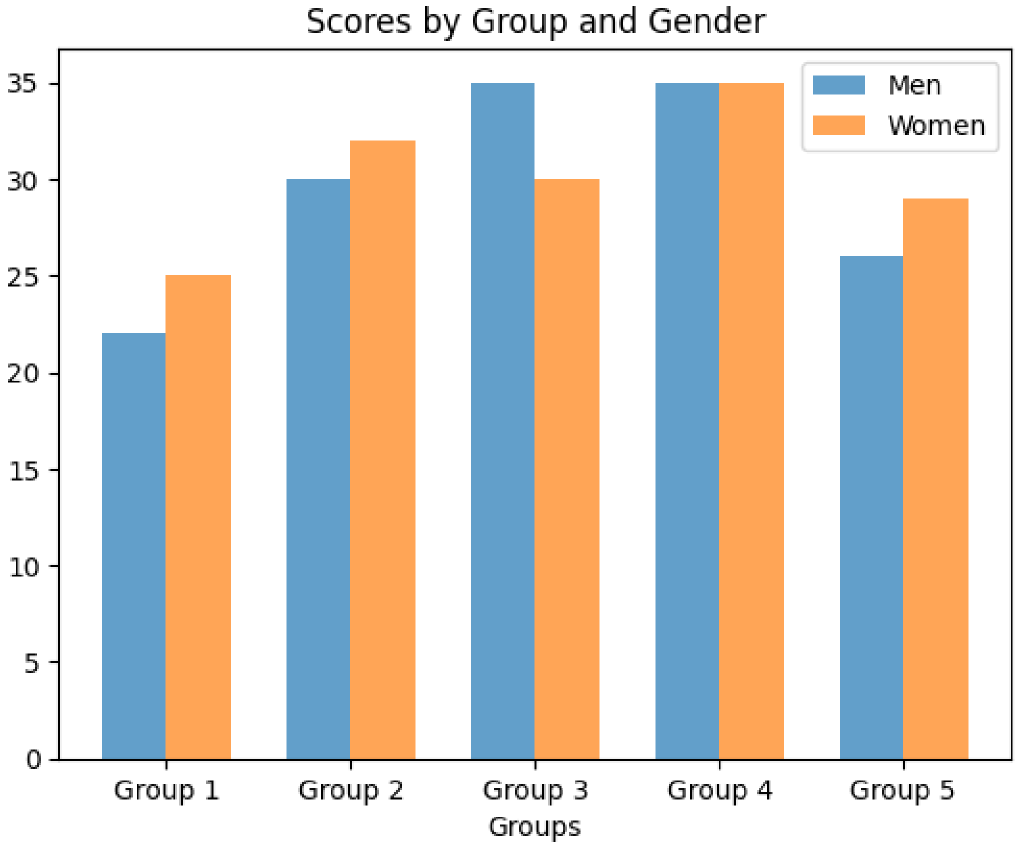
```
[ ]: # 29.Write a Python programming to display a bar chart of the popularity of␣
      ↪programming Languages. Use different color for each bar.
     import matplotlib.pyplot as plt

     # Sample data
     languages = ["Java", "Python", "PHP", "JavaScript", "C#", "C++"]
     popularity = [22.2, 17.6, 8.8, 8, 7.7, 6.7]
     colors = ['blue', 'green', 'red', 'yellow', 'purple', 'orange']

     # Create a bar chart with different colors
     plt.figure(figsize=(10, 6))
     plt.bar(languages, popularity, color=colors)

     # Add labels and title
     plt.xlabel("Programming Languages")
     plt.ylabel("Popularity (%)")
     plt.title("Popularity of Programming Languages")

     # Show the bar chart
     plt.show()
```



[ ]:

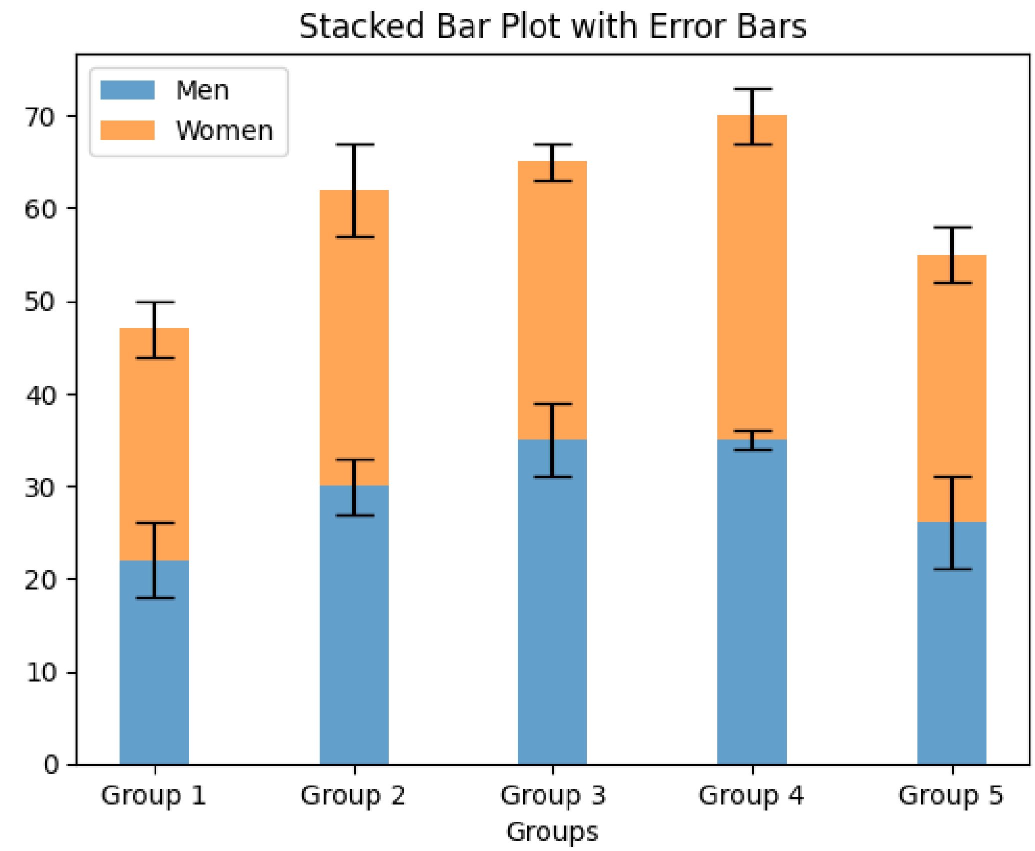
```
#30.Write a Python program to create bar plot of scores by group and gender.␣
  ↪Use multiple X values on the same chart for men and women.
import matplotlib.pyplot as plt
import numpy as np

# Sample data
groups = ['Group 1', 'Group 2', 'Group 3', 'Group 4', 'Group 5']
means_men = [22, 30, 35, 35, 26]
means_women = [25, 32, 30, 35, 29]

# Define the width of the bars
bar_width = 0.35
index = np.arange(len(groups))

# Create the bar plot for men
plt.bar(index, means_men, bar_width, label='Men', alpha=0.7)

# Create the bar plot for women with an offset
plt.bar(index + bar_width, means_women, bar_width, label='Women', alpha=0.7)

# Set X-axis labels and title
plt.xlabel('Groups')
plt.xticks(index + bar_width / 2, groups)
plt.title('Scores by Group and Gender')
```

```
# Add legend
plt.legend()

# Show the bar plot
plt.show()
```



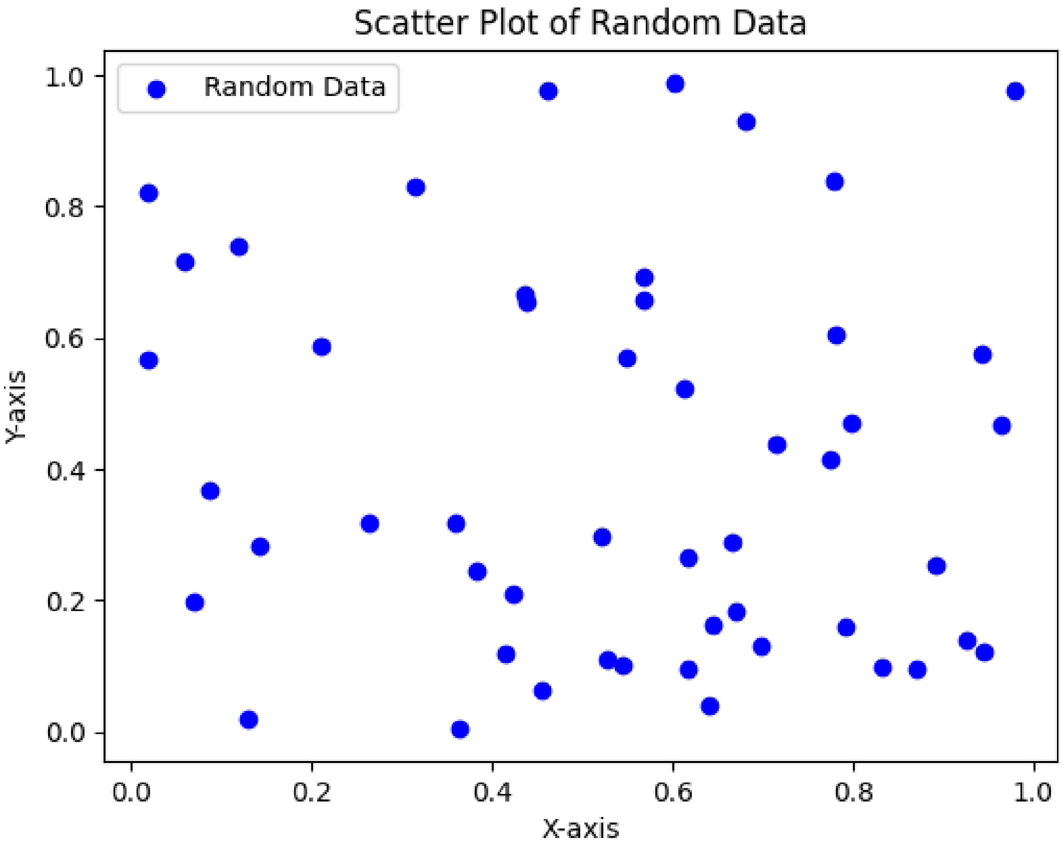
[ ]:
```
#31.Write a Python program to create a stacked bar plot with error bars.
import matplotlib.pyplot as plt
import numpy as np

# Sample data
groups = ['Group 1', 'Group 2', 'Group 3', 'Group 4', 'Group 5']
means_men = [22, 30, 35, 35, 26]
means_women = [25, 32, 30, 35, 29]
std_men = [4, 3, 4, 1, 5]
std_women = [3, 5, 2, 3, 3]

# Define the width of the bars
bar_width = 0.35
```

```
index = np.arange(len(groups))

# Create the stacked bar plot
plt.bar(index, means_men, bar_width, yerr=std_men, label='Men', alpha=0.7,␣
 ↪capsize=7)
plt.bar(index, means_women, bar_width, yerr=std_women, bottom=means_men,␣
 ↪label='Women', alpha=0.7, capsize=7)

# Set X-axis labels and title
plt.xlabel('Groups')
plt.xticks(index, groups)
plt.title('Stacked Bar Plot with Error Bars')

# Add legend
plt.legend()

# Show the bar plot
plt.show()
```



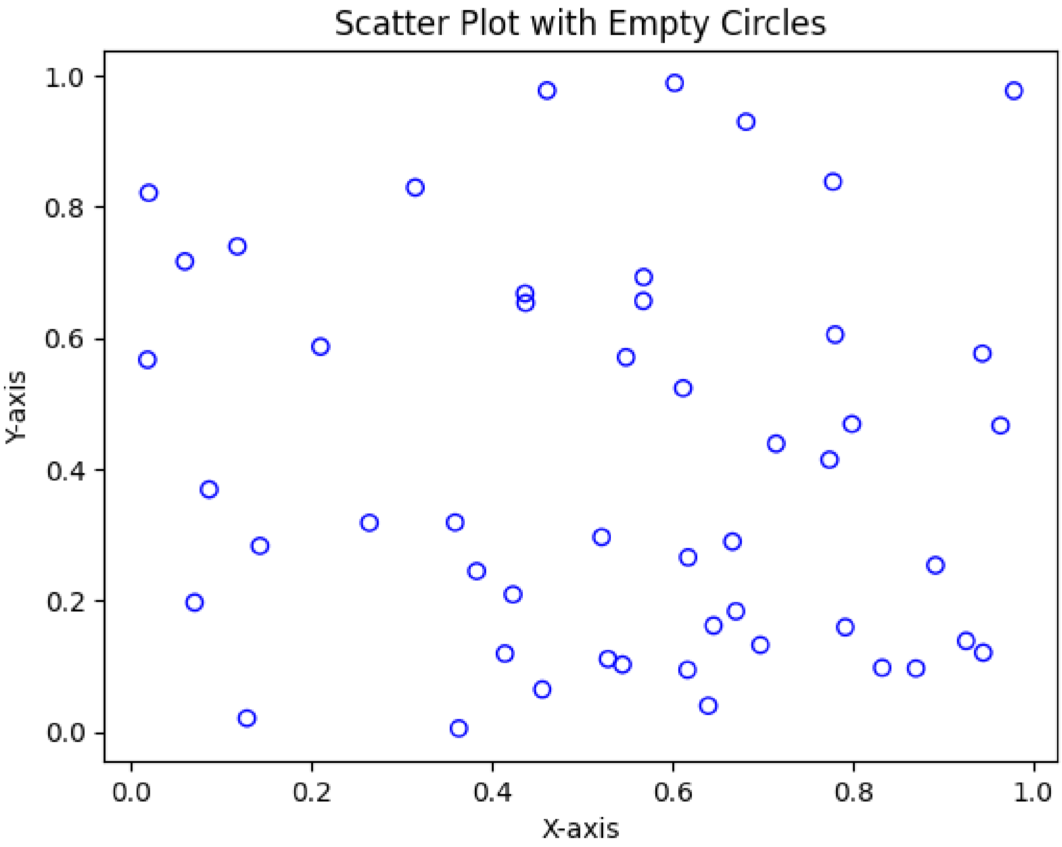
[ ]:

```
#32.Write a Python program to draw a scatter graph taking a random distribution␣
 ↪in X and Y and plotted against each other.
import matplotlib.pyplot as plt
import numpy as np

# Generate random data for X and Y
np.random.seed(0)  # For reproducibility
x = np.random.rand(50)  # 50 random values for X
y = np.random.rand(50)  # 50 random values for Y

# Create a scatter plot
plt.scatter(x, y, label='Random Data', color='blue', marker='o')

# Set labels for X and Y axes
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# Set a title for the scatter plot
plt.title('Scatter Plot of Random Data')

# Show a legend
plt.legend()

# Show the scatter plot
plt.show()
```



```
[ ]: #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.
     import matplotlib.pyplot as plt
     import numpy as np

     # Generate random data for X and Y
     np.random.seed(0)  # For reproducibility
     x = np.random.rand(50)  # 50 random values for X
     y = np.random.rand(50)  # 50 random values for Y

     # Create a scatter plot with empty circles (hollow markers)
     plt.scatter(x, y, label='Random Data', color='blue', marker='o',␣
      ↪facecolors='none', edgecolors='blue')

     # Set labels for X and Y axes
     plt.xlabel('X-axis')
     plt.ylabel('Y-axis')

     # Set a title for the scatter plot
     plt.title('Scatter Plot with Empty Circles')
```

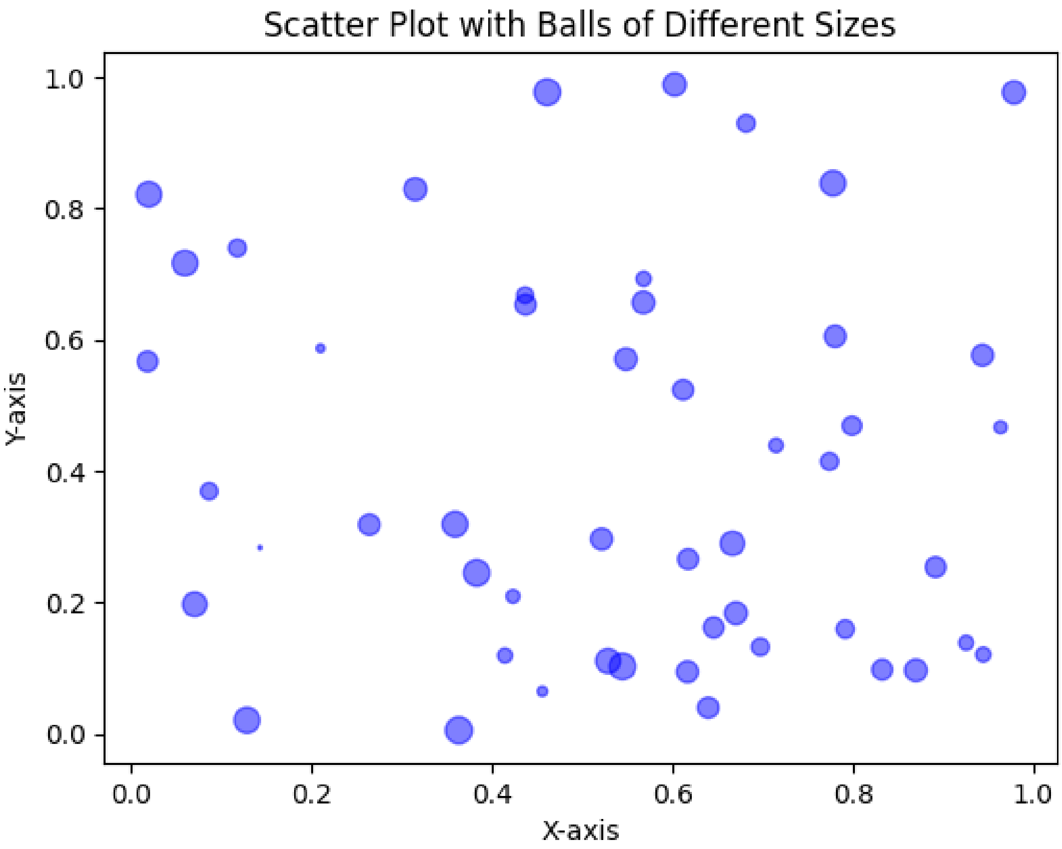
```
# Show the scatter plot
plt.show()
```



[ ]:

```
#34.Write a Python program to draw a scatter plot using random distributions to␣
 ↪generate balls of different sizes.
import matplotlib.pyplot as plt
import numpy as np

# Generate random data for X and Y
np.random.seed(0)  # For reproducibility
x = np.random.rand(50)  # 50 random values for X
y = np.random.rand(50)  # 50 random values for Y

# Generate random sizes for the balls
sizes = np.random.rand(50) * 100

# Create a scatter plot with balls of different sizes
plt.scatter(x, y, s=sizes, label='Random Data', color='blue', alpha=0.5)
```

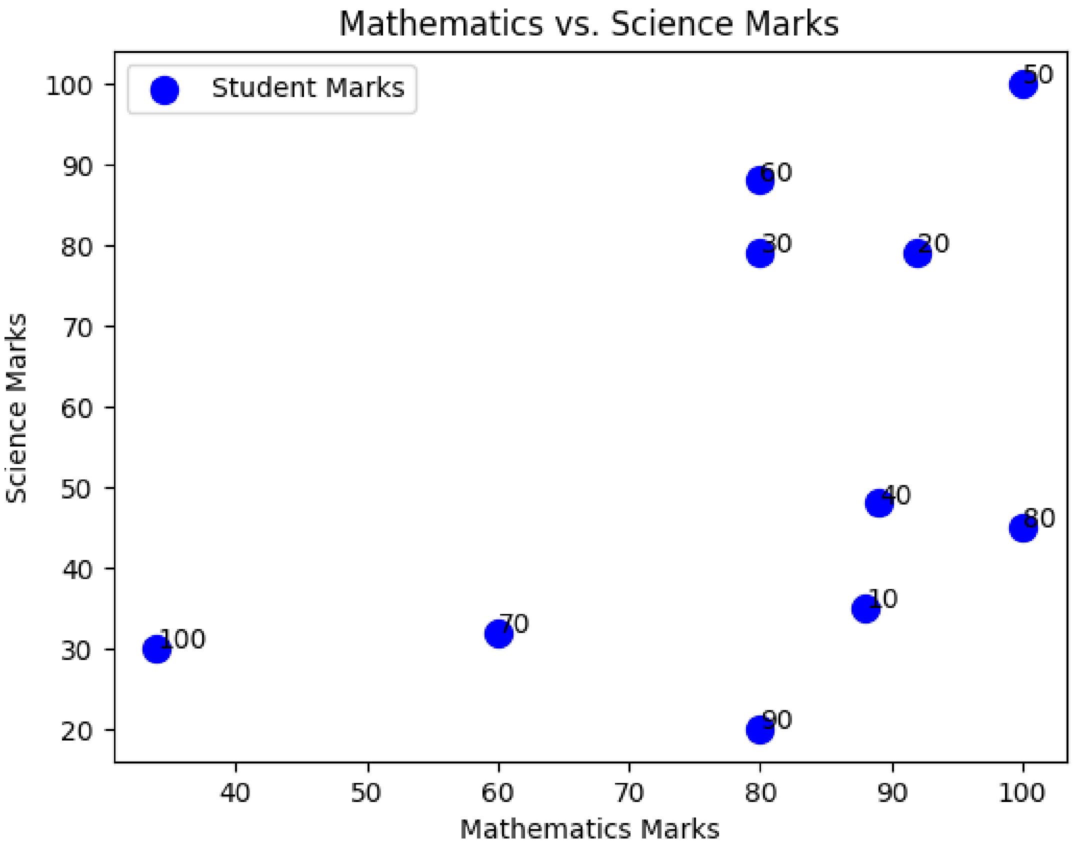
```
# Set labels for X and Y axes
plt.xlabel('X-axis')
plt.ylabel('Y-axis')

# Set a title for the scatter plot
plt.title('Scatter Plot with Balls of Different Sizes')

# Show the scatter plot
plt.show()
```



[ ]:

```
#35.Write a Python program to draw a scatter plot comparing two subject marks␣
 ↪of Mathematics and Science. Use marks of 10 students.
import matplotlib.pyplot as plt

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

# Create a scatter plot
```

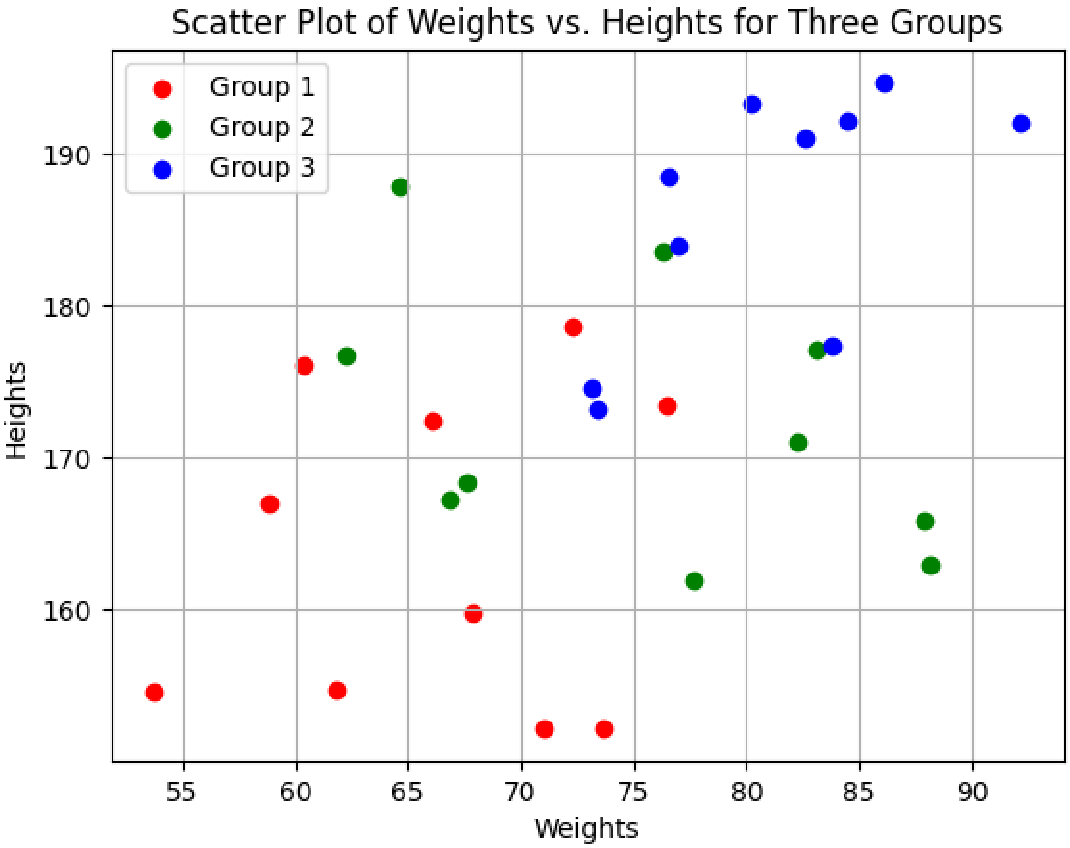
```
plt.scatter(math_marks, science_marks, s=100, c='blue', marker='o',␣
 ↪label='Student Marks')

# Set labels for X and Y axes
plt.xlabel('Mathematics Marks')
plt.ylabel('Science Marks')

# Set a title for the scatter plot
plt.title('Mathematics vs. Science Marks')

# Display the marks range on the plot
for i, txt in enumerate(marks_range):
    plt.annotate(txt, (math_marks[i], science_marks[i]))

plt.legend()
plt.show()
```



[ ]:

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

```
import matplotlib.pyplot as plt
import random

# Generate random data for three groups
group1_weights = [random.uniform(50, 80) for _ in range(10)]
group1_heights = [random.uniform(150, 180) for _ in range(10)]

group2_weights = [random.uniform(60, 90) for _ in range(10)]
group2_heights = [random.uniform(160, 190) for _ in range(10)]

group3_weights = [random.uniform(70, 100) for _ in range(10)]
group3_heights = [random.uniform(170, 200) for _ in range(10)]

# Create a scatter plot for each group
plt.scatter(group1_weights, group1_heights, label='Group 1', color='red')
plt.scatter(group2_weights, group2_heights, label='Group 2', color='green')
plt.scatter(group3_weights, group3_heights, label='Group 3', color='blue')

# Set labels for X and Y axes
plt.xlabel('Weights')
plt.ylabel('Heights')

# Set a title for the scatter plot
plt.title('Scatter Plot of Weights vs. Heights for Three Groups')

# Show legend
plt.legend()

# Show the scatter plot
plt.grid(True)
plt.show()
```



```
[ ]: #37.Write a Pandas program to create a dataframe from a dictionary and display␣
       ↪it.
     import pandas as pd

     # Sample data as a dictionary
     data = {'X': [78, 85, 96, 80, 86], 'Y': [84, 94, 89, 83, 86], 'Z': [86, 97, 96,␣
       ↪72, 83]}

     # Create a DataFrame from the dictionary
     df = pd.DataFrame(data)

     # Display the DataFrame
     print(df)
```

```
    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.
import pandas as pd
import numpy as np

# Sample data and 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']

# Create a DataFrame
df = pd.DataFrame(exam_data, index=labels)

# Display the DataFrame
print(df)
```

```
        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.
import pandas as pd
import numpy as np

# Sample data and 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']

# Create a DataFrame
df = pd.DataFrame(exam_data, index=labels)

# Get the first 3 rows
first_3_rows = df.head(3)

# Display the first 3 rows
print(first_3_rows)
```

```
        name  score  attempts qualify
a  Anastasia   12.5         1     yes
b       Dima    9.0         3      no
c  Katherine   16.5         2     yes
```

```
[ ]: #40 40.        Write a Pandas program to select the 'name' and 'score' columns␣
      ↪from the following DataFrame.
     import pandas as pd
     import numpy as np

     # Sample data and 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']

     # Create a DataFrame
     df = pd.DataFrame(exam_data, index=labels)

     # Select 'name' and 'score' columns
     selected_columns = df[['name', 'score']]

     # Display the selected columns
     print(selected_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
```