Index

Sr. No.	Problem Statement	Date	Sign
1.	Python program to perform Array operations using Numpy package.		
2.	Python program to perform Data Manipulation operations using Pandas package.		
3.	Python program to display multiple types of charts using Matplotlib package.		
4.	Create a Python program that generates and displays a normal distribution curve.		
5.	Write a Python program that calculates and displays a frequency distribution of a given dataset.		
6.	Write a Python program that computes the correlation between two sets of data and visualizes the relationship using a scatter plot.		
7.	Develop a Python program to calculate the correlation coefficient between two sets of data.		
8.	Develop a Python program that performs Simple Linear Regression to model the relationship between two variables.		
9.	Write a Python program to convert data (e.g., JSON, CSV, XML, etc.) from one format to another.		

10.	Create the dashboard using Power BI.		
11.	Create the Dashboard using Tableau.		
12.	Project on Data Science.		

Practical -1 :- Python Program to perform Array operation using Numpy package.

Array Operations.

```
a = np.arange(15).reshape(3, 5)
a
```

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

```
a.shape
```

```
(3, 5)
```

```
a.ndim
```

```
2
```

```
a.dtype.name
type(a)
```

```
numpy.ndarray
```

```
print(np.__version__)
```

```
1.26.4
```

```
a.itemsize
```

```
8
```

```
a.size
```

```
15
```

```
a.tolist()
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
import numpy as np
a = np.array([2, 3, 4])
a
```

```
array([2, 3, 4])
```

```
a.dtype
```

```
dtype('int64')
```

```
np.ones((2, 3, 4), dtype=np.int16)
```

```
array([[[1, 1, 1, 1],
        [1, 1, 1, 1],
        [1, 1, 1, 1]],

       [[1, 1, 1, 1],
        [1, 1, 1, 1],
        [1, 1, 1, 1]]], dtype=int16)
```

```
np.zeros((3, 3))
```

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

```
a = np.arange(10)                    # 1d array
print(a)
```

```
[0 1 2 3 4 5 6 7 8 9]
```

```
b = np.arange(12).reshape(4, 3)     # 2d array
print(b)
```

```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
```

```
c = np.arange(24).reshape(2,3,4)  # 3d array
print(c)
```

```
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

```
print(np.arange(10000))
```

```
[   0    1    2 ... 9997 9998 9999]
```

```
print(np.arange(10000).reshape(100, 100))
```

```
[[   0    1    2 ...   97   98   99]
 [ 100  101  102 ...  197  198  199]
 [ 200  201  202 ...  297  298  299]
 ...
 [9700 9701 9702 ... 9797 9798 9799]
 [9800 9801 9802 ... 9897 9898 9899]
 [9900 9901 9902 ... 9997 9998 9999]]
```

```
b = np.arange(12).reshape(3, 4)
b
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

```
b.sum(axis=0)
```

```
array([12, 15, 18, 21])
```

```
b.min(axis=1)
```

```
array([0, 4, 8])
```

```
b.cumsum(axis=1)
```

```
array([[ 0,  1,  3,  6],
       [ 4,  9, 15, 22],
       [ 8, 17, 27, 38]])
```

```
a = np.matrix('1 2; 3 4')
a
```

```
matrix([[1, 2],
        [3, 4]])
```

```
np.matrix([[1, 2], [3, 4]])
```

```
matrix([[1, 2],
        [3, 4]])
```

```
x = np.matrix(np.arange(12).reshape((3,4))); x
```

```
matrix([[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]])
```

```
             [ 8,  9, 10, 11]])
```

```
#Max and Min in array
print("Maximum in Array",x.max())
print("Minimum in Array",x.min())
print("Mean in Array",x.mean())
```

```
Maximum in Array 11
Minimum in Array 0
Mean in Array 5.5
```

```
# Creating arrays
arr1 = np.array([1, 2, 3, 4, 5])
arr2 = np.array([6, 7, 8, 9, 10])

print("Array 1:", arr1)
print("Array 2:", arr2)
```

```
Array 1: [1 2 3 4 5]
Array 2: [ 6  7  8  9 10]
```

```
#Sorting Array
arr1.sort()
print(arr1)
```

```
[21 58 73 84 93]
```

```
arr = np.array([1, 2, 3, 4])
print("array + 5:", arr + 5)
print("array * 2:", arr * 2)
print("array squared:", arr ** 2)
```

```
array + 5: [6 7 8 9]
array * 2: [2 4 6 8]
array squared: [ 1  4  9 16]
```

```
# Array addition
sum_arr = np.add(arr1, arr2)
print("Sum of arrays:", sum_arr)
```

```
Sum of arrays: [ 27  65  81  93 103]
```

```
# Array subtraction
diff_arr = np.subtract(arr1, arr2)
print("Difference of arrays:", diff_arr)
```

```
Difference of arrays: [15 51 65 75 83]
```

```
# Array multiplication
prod_arr = np.multiply(arr1, arr2)
```

```
print("Product of arrays:", prod_arr)
```

```
Product of arrays: [126 406 584 756 930]
```

```
# Array division
div_arr = np.divide(arr1, arr2)
print("Division of arrays:", div_arr)
```

```
Division of arrays: [3.5        8.28571429 9.125      9.33333333 9.3
```

```
# Reshaping an array
reshaped_arr = np.arange(1, 10).reshape(3, 3)
print("Reshaped 3x3 array:")
print(reshaped_arr)
```

```
Reshaped 3x3 array:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

```
# Transposing an array
transposed_arr = reshaped_arr.T
print("Transposed array:")
print(transposed_arr)
```

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

```
# Concatenation of arrays
concatenated_arr = np.concatenate((arr1, arr2))
print("Concatenated array:", concatenated_arr)
```

```
Concatenated array: [21 58 73 84 93  6  7  8  9 10]
```

```
# Indexing and Slicing
print("First element of arr1:", arr1[0])
print("Last element of arr1:", arr1[-1])
print("Elements from index 1 to 3 of arr1:", arr1[1:4])
print("Every second element of arr1:", arr1[::2])
```

```
First element of arr1: 21
Last element of arr1: 93
Elements from index 1 to 3 of arr1: [58 73 84]
Every second element of arr1: [21 73 93]
```

```
# Slicing in a 2D array
print("First row of reshaped array:", reshaped_arr[0])
print("2*3 MATRIX array:", reshaped_arr[2,3])
```

```
print("2-3 matrix array: ", reshaped_arr[2,3])
print("First column of reshaped array:", reshaped_arr[:, 0])
print("Submatrix from reshaped array:", reshaped_arr[0:2, 1:3])
```

```
First row of reshaped array: [1 2 3]
First column of reshaped array: [1 4 7]
Submatrix from reshaped array: [[2 3]
 [5 6]]
```

```
arr = np.array([1,2,3,4,5,6,9,7,5])
newarr = np.array_split(arr, 3)
print(newarr)
```

Start coding or generate with AI.

Start coding or generate with AI.

Practical -2 :- Python Program to perform Data Manipulation operations using Pandas package.

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

```
df = pd.DataFrame(data=np.random.randint(0,10,(5,3)),
                  columns=['c1','c2','c3'],
                  index = ['A','B','C','D','E'])
df
```

	c1	c2	c3
A	3	5	3
B	9	6	9
C	9	3	5
D	8	2	3
E	4	9	8

```
# Properties
df.shape
```

```
(5, 3)
```

```
pip install pandas
```

```
df.dtypes
```

```
c1    int32
c2    int32
c3    int32
dtype: object
```

```
list(df.columns)
```

```
['c1', 'c2', 'c3']
```

```
df.index
```

```
Index(['A', 'B', 'C', 'D', 'E'], dtype='object')
```

```
df
```

	c1	c2	c3
A	3	5	3
B	9	6	9
C	9	3	5
D	8	2	3
E	4	9	8

```
# Selection of columns
df['c1']  # 1D
```

```
A    3
B    9
C    9
D    8
E    4
Name: c1, dtype: int32
```

```
df[['c1','c2']] # list of columns
```

	c1	c2
A	3	5
B	9	6
C	9	3
D	8	2
E	4	9

```
df
```

	c1	c2	c3
A	3	5	3
B	9	6	9
C	9	3	5
D	8	2	3
E	4	9	8

```
df.loc['A']
```

```
c1    3
c2    5
c3    3
Name: A, dtype: int32
```

```
df.loc[['A','D']]
```

	c1	c2	c3
A	3	5	3
D	8	2	3

```
df
```

	c1	c2	c3
A	3	5	3
B	9	6	9
C	9	3	5
D	8	2	3
E	4	9	8

```
df.iloc[[2,3]]
```

	c1	c2	c3
C	6	10	3
D	14	15	4

```
df.iloc[::2]
```

	c1	c2	c3
A	3	5	3
C	9	3	5
E	4	9	8

```
df
```

```
df
df.iloc[1:2:][0:1:]
```

	c1	c2	c3
B	9	6	9

```
df.iloc[1::2][['c2','c3']]
```

```
df.iloc[1:2][['c2','c3']]
```

	c2	c3
B	6	9
D	2	3

```
# Adding new columns
df['c4'] = [11,12,13,14,15]  # Same as of Dict column inserted
df
```

	c1	c2	c3	c4
A	3	5	3	11
B	9	6	9	12
C	9	3	5	13
D	8	2	3	14
E	4	9	8	15

```
df.loc['F'] = [1,2,3,4]   #row inserted
```

```
df
```

	c1	c2	c3	c4
A	3	5	3	11
B	9	6	9	12
C	9	3	5	13
D	8	2	3	14
E	4	9	8	15
F	1	2	3	4

```
# Delete row/record or column
df.drop('c4', axis=1)
```

	c1	c2	c3
A	3	5	3
B	9	6	9
C	9	3	5
D	8	2	3
E	4	9	8

F	1	2	3

```
df
```

	c1	c2	c3	c4
A	3	5	3	11
B	9	6	9	12
C	9	3	5	13
D	8	2	3	14
E	4	9	8	15
F	1	2	3	4

```
df.drop('c4', axis=1, inplace=True) # Permanent delete
```

```
df
```

	c1	c2	c3
A	3	5	3
B	9	6	9
C	9	3	5
D	8	2	3
E	4	9	8
F	1	2	3

```
df.drop('F', axis=0)
```

	c1	c2	c3
A	3	5	3
B	9	6	9
C	9	3	5
D	8	2	3
E	4	9	8

```
df
```

	c1	c2	c3

A	3	5	3
B	9	6	9
C	9	3	5
D	8	2	3
E	4	9	8
F	1	2	3

```
df.drop('F', axis=0, inplace=True)
```

```
df
```

	c1	c2	c3
A	3	5	3
B	9	6	9
C	9	3	5
D	8	2	3
E	4	9	8

```
df
```

	c1	c2	c3
A	3	5	3
B	9	6	9
C	9	3	5
D	8	2	3
E	4	9	8

```
df.drop(index=df.index[2])
```

	c1	c2	c3
A	3	5	3
B	9	6	9
D	8	2	3
E	4	9	8

```
df.index[2]
```

```
df.index[2]
```

```
'C'
```

```
df.iloc[[0,1,3,4]]
```

	c1	c2	c3
A	8	7	10
B	3	10	0
D	14	15	4
E	13	2	16

Reading Data From external files

```
df = pd.read_excel('stud_db.xlsx', sheet_name='XII')
df
```

	Roll	Name	Marks	Address	Dept
0	1	qw	76	Pune	1
1	2	df	45	Mumbai	1
2	3	vc	56	Nagpur	1
3	4	bn	78	Pune	2
4	5	hj	74	Mumbai	2
5	6	yu	49	Nagpur	2
6	7	fg	68	Pune	2
7	8	tr	84	Mumbai	1

```
df.shape
```

```
(8, 5)
```

```
df.columns
```

```
Index(['Roll', 'Name', 'Marks', 'Address', 'Dept'], dtype='object')
```

```
df['Marks'].sum()
```

```
530
```

```
df['Marks'].mean()
```

```
df['Marks'].mean()
```

```
66.25
```

```
df['Marks']
```

```
0    76
1    45
2    56
3    78
4    74
5    49
6    68
7    84
Name: Marks, dtype: int64
```

```
df['Marks']<60 # Broadcasting
```

	Roll	Name	Marks	Address	Dept
1	2	df	45	Mumbai	1
2	3	vc	56	Nagpur	1
5	6	yu	49	Nagpur	2

```
df[df['Marks']<60] # Selection
```

	Roll	Name	Marks	Address	Dept
1	2	df	45	Mumbai	1
2	3	vc	56	Nagpur	1
5	6	yu	49	Nagpur	2

```
df_pune = df[df['Address']=='Pune']
df_pune
df_pune['Marks'].mean()
```

```
74.0
```

```
df_mumbai = df[df['Address']=='Mumbai']
df_mumbai['Marks'].mean()
```

```
67.66666666666667
```

```
df_nagpur = df[df['Address']=='Nagpur']
df_nagpur['Marks'].mean()
```

```
52.5
```

```
df
```

	Roll	Name	Marks	Address	Dept
0	1	qw	76	Pune	1
1	2	df	45	Mumbai	1
2	3	vc	56	Nagpur	1
3	4	bn	78	Pune	2
4	5	hj	74	Mumbai	2
5	6	yu	49	Nagpur	2
6	7	fg	68	Pune	2
7	8	tr	84	Mumbai	1

```
df['Address'].unique()
```

```
array(['Pune', 'Mumbai', 'Nagpur'], dtype=object)
```

```
df['Address'].nunique()
```

```
3
```

```
df['Address'].value_counts()
```

```
Address
Pune      3
Mumbai    3
Nagpur    2
Name: count, dtype: int64
```

```
df
```

	Roll	Name	Marks	Address	Dept
0	1	qw	76	Pune	1
1	2	df	45	Mumbai	1
2	3	vc	56	Nagpur	1
3	4	bn	78	Pune	2
4	5	hj	74	Mumbai	2
5	6	yu	49	Nagpur	2
6	7	fg	68	Pune	2
7	8	tr	84	Mumbai	1

```
df['Dept'].value_counts()
```

```
1    4
2    4
Name: Dept, dtype: int64
```

```
# To find name of topper
df[df['Marks'].max()==df['Marks']]['Name']
```

```
7    tr
Name: Name, dtype: object
```

```
# To find city of topper
df[df['Marks'].max()==df['Marks']]['Address'].iloc[0]
```

```
'Mumbai'
```

```
# To find Name second topper
temp = df[df['Marks'] < df['Marks'].max()] # All students except topper
marks_2nd_topper = temp['Marks'].max()
df[df['Marks']==marks_2nd_topper]['Name'].iloc[0]
```

```
'bn'
```

```
df.sort_values('Marks', ascending=False)['Name'].iloc[3] # 4th topper
```

	Roll	Name	Marks	Address	Dept
7	8	tr	84	Mumbai	1
3	4	bn	78	Pune	2
0	1	qw	76	Pune	1
4	5	hj	74	Mumbai	2
6	7	fg	68	Pune	2
2	3	vc	56	Nagpur	1
5	6	yu	49	Nagpur	2
1	2	df	45	Mumbai	1

```
df
```

	Roll	Name	Marks	Address	Dept
0	1	qw	76	Pune	1
1	2	df	45	Mumbai	1
2	3	vc	56	Nagpur	1

3	4	bn	78	Pune	2
4	5	hj	74	Mumbai	2
5	6	yu	49	Nagpur	2
6	7	fg	68	Pune	2
7	8	tr	84	Mumbai	1

```
df['City_Code'] = df['Address'].apply(lambda x:x[:2])
```

```
df
```

	Roll	Name	Marks	Address	Dept	City_Code
0	1	qw	76	Pune	1	Pu
1	2	df	45	Mumbai	1	Mu
2	3	vc	56	Nagpur	1	Na
3	4	bn	78	Pune	2	Pu
4	5	hj	74	Mumbai	2	Mu
5	6	yu	49	Nagpur	2	Na
6	7	fg	68	Pune	2	Pu
7	8	tr	84	Mumbai	1	Mu

```
df['Marks'].apply(lambda x:x/10)
```

```
0    7.6
1    4.5
2    5.6
3    7.8
4    7.4
5    4.9
6    6.8
7    8.4
Name: Marks, dtype: float64
```

GROUPBY

```
df
```

	Roll	Name	Marks	Address	Dept	City_Code
0	1	qw	76	Pune	1	Pu
1	2	df	45	Mumbai	1	Mu

2	3	vc	56	Nagpur	1	Na
3	4	bn	78	Pune	2	Pu
4	5	hj	74	Mumbai	2	Mu
5	6	yu	49	Nagpur	2	Na
6	7	fg	68	Pune	2	Pu
7	8	tr	84	Mumbai	1	Mu

```
# City-wise Avg marks
df.groupby('Address').mean()['Marks']
```

```
Address
Mumbai    67.666667
Nagpur    52.500000
Pune      74.000000
Name: Marks, dtype: float64
```

```
df.groupby('Dept').max()['Roll']
```

```
Dept
1    8
2    7
Name: Roll, dtype: int64
```

Merging of DF's

```
df_student = pd.read_excel('stud_db.xlsx',sheet_name='XII')
df_dept = pd.read_excel('stud_db.xlsx',sheet_name='Depts')
```

```
df_student
```

	Roll	Name	Marks	Address	Dept
0	1	qw	76	Pune	1
1	2	df	45	Mumbai	1
2	3	vc	56	Nagpur	1
3	4	bn	78	Pune	2
4	5	hj	74	Mumbai	2
5	6	yu	49	Nagpur	2
6	7	fg	68	Pune	2
7	8	tr	84	Mumbai	1

```
df_dept
```

	dept_id	Name	HOD
0	1	Comp Science	XYZ
1	2	Statistics	PQR
2	3	Maths	MNO

```
pd.merge(df_student,df_dept,left_on='Dept', right_on='dept_id')
```

	Roll	Name_x	Marks	Address	Dept	dept_id	Name_y	HOD
0	1	qw	76	Pune	1	1	Comp Science	XYZ
1	2	df	45	Mumbai	1	1	Comp Science	XYZ
2	3	vc	56	Nagpur	1	1	Comp Science	XYZ
3	8	tr	84	Mumbai	1	1	Comp Science	XYZ
4	4	bn	78	Pune	2	2	Statistics	PQR
5	5	hj	74	Mumbai	2	2	Statistics	PQR
6	6	yu	49	Nagpur	2	2	Statistics	PQR
7	7	fg	68	Pune	2	2	Statistics	PQR

```
pd.merge(df_student,df_dept,left_on='Dept', right_on='dept_id', how='right')
```

	Roll	Name_x	Marks	Address	Dept	dept_id	Name_y	HOD
0	1.0	qw	76.0	Pune	1.0	1	Comp Science	XYZ
1	2.0	df	45.0	Mumbai	1.0	1	Comp Science	XYZ
2	3.0	vc	56.0	Nagpur	1.0	1	Comp Science	XYZ
3	8.0	tr	84.0	Mumbai	1.0	1	Comp Science	XYZ
4	4.0	bn	78.0	Pune	2.0	2	Statistics	PQR
5	5.0	hj	74.0	Mumbai	2.0	2	Statistics	PQR
6	6.0	yu	49.0	Nagpur	2.0	2	Statistics	PQR
7	7.0	fg	68.0	Pune	2.0	2	Statistics	PQR
8	NaN	NaN	NaN	NaN	NaN	3	Maths	MNO

```
import pandas as pd
data = {
    'ID': [101, 102, 103, 104, 105],
    'Name': ['Sonal', 'Tanvi', 'Anuja', 'Rutuja', 'Eve'],
    'Age': [20, 22, 23, 20, 15],
```

```
    'Salary': [50000, 60000, 70000, 80000, 90000],
    'Department': ['HR' , 'IT', 'IT', 'Finance', 'HR']
}

df = pd.DataFrame(data)
print("Original DataFrame:")
print(df)


print("\nPrint Data (Name):")
print(df['Age'])


print("\nFiltering employees with Salary < 70000:")
filtered_df = df[df['Salary'] <70000]
print(filtered_df)


print("\nSorting employees by Age:")
sorted_df = df.sort_values(by='Age', ascending=False)
print(sorted_df)


print("\nAverage Salary by Department:")
grouped_df = df.groupby('Department')['Salary'].mean()
print(grouped_df)


# Updating Values
df.loc[df['Department'] == 'IT', 'Salary'] += 5000
print("\nDataFrame after Salary increment for IT Department:\n ")
print(df)


# Dropping a Column
df.drop(columns=['Bonus'], inplace=True)
print("\nDataFrame after dropping Bonus column:")
print(df)


# Reset Index
df.reset_index(drop=True, inplace=True)
print("\nDataFrame after resetting index:")
print(df)



# Sample data
data = {
    'Name': ['Alice', 'Bob', 'Charlie', 'David'],
    'Age': [25, 30, 35, 40],
    'City': ['New York', 'Los Angeles', 'Chicago', 'Houston']
}

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

```
# Save to CSV
df.to_csv('dataset.csv', index=False)

print("Dataset saved as dataset.csv")


ds_loaded = pd.read_csv('dataset.csv')
print(ds_loaded)
```

Start coding or generate with AI.

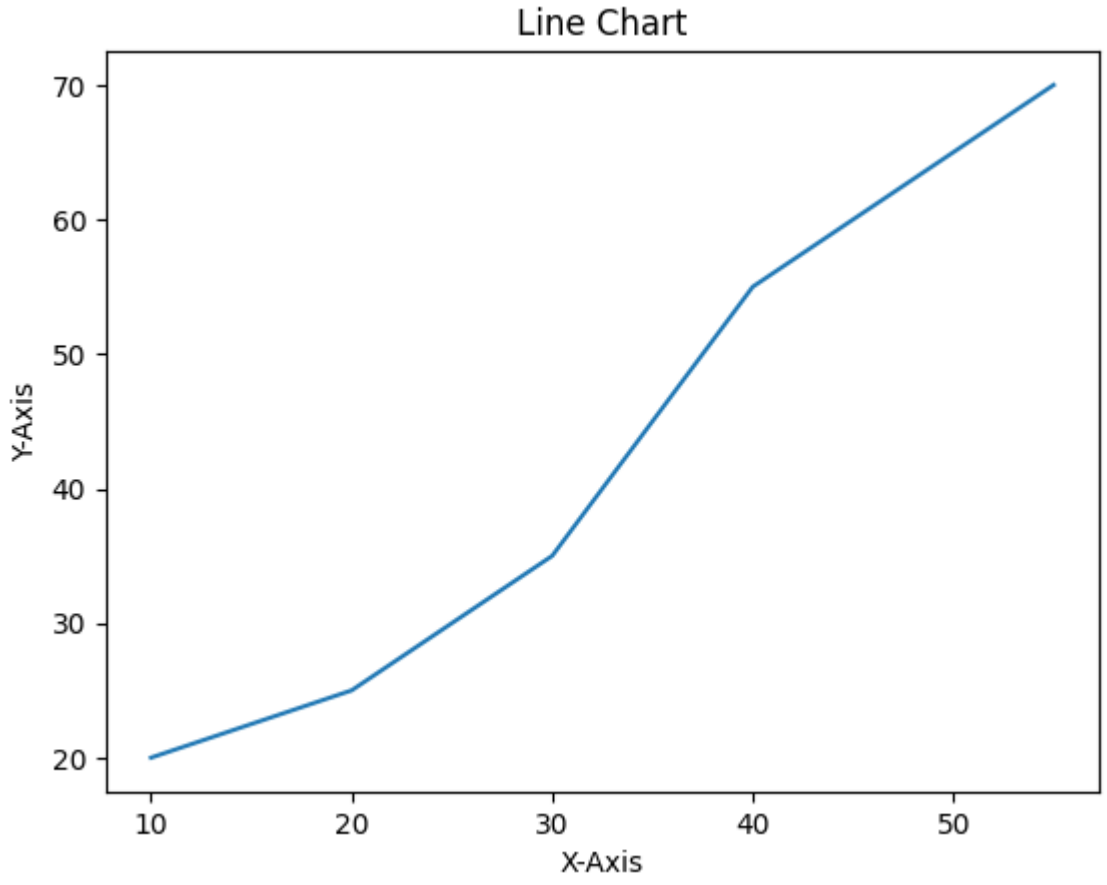
Start coding or generate with AI.

Start coding or generate with AI.

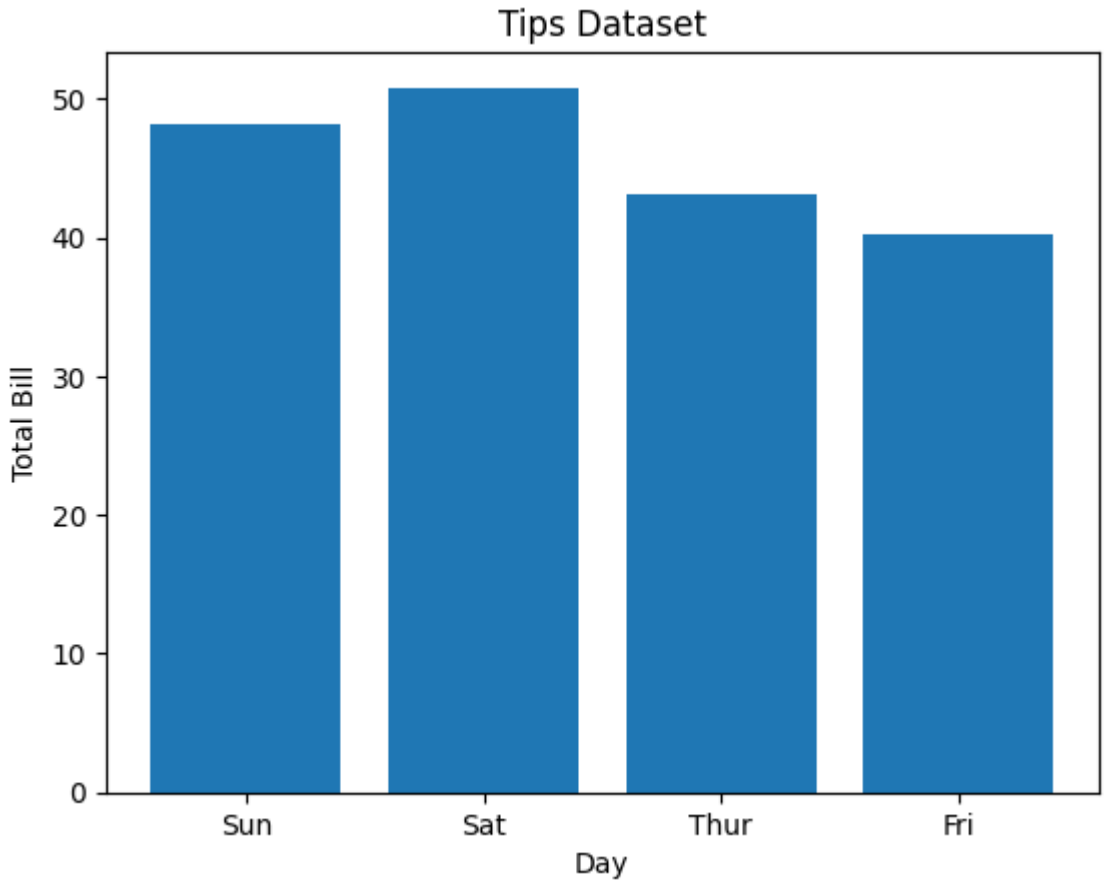
```
pip install matplotlib
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/di
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.11/dis
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.1
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/pytho
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-p
```

```
import matplotlib.pyplot as plt

x = [10, 20, 30, 40,55]
y = [20, 25, 35, 55,70]

plt.plot(x, y)

plt.title("Line Chart")

plt.ylabel('Y-Axis')

plt.xlabel('X-Axis')
plt.show()
```

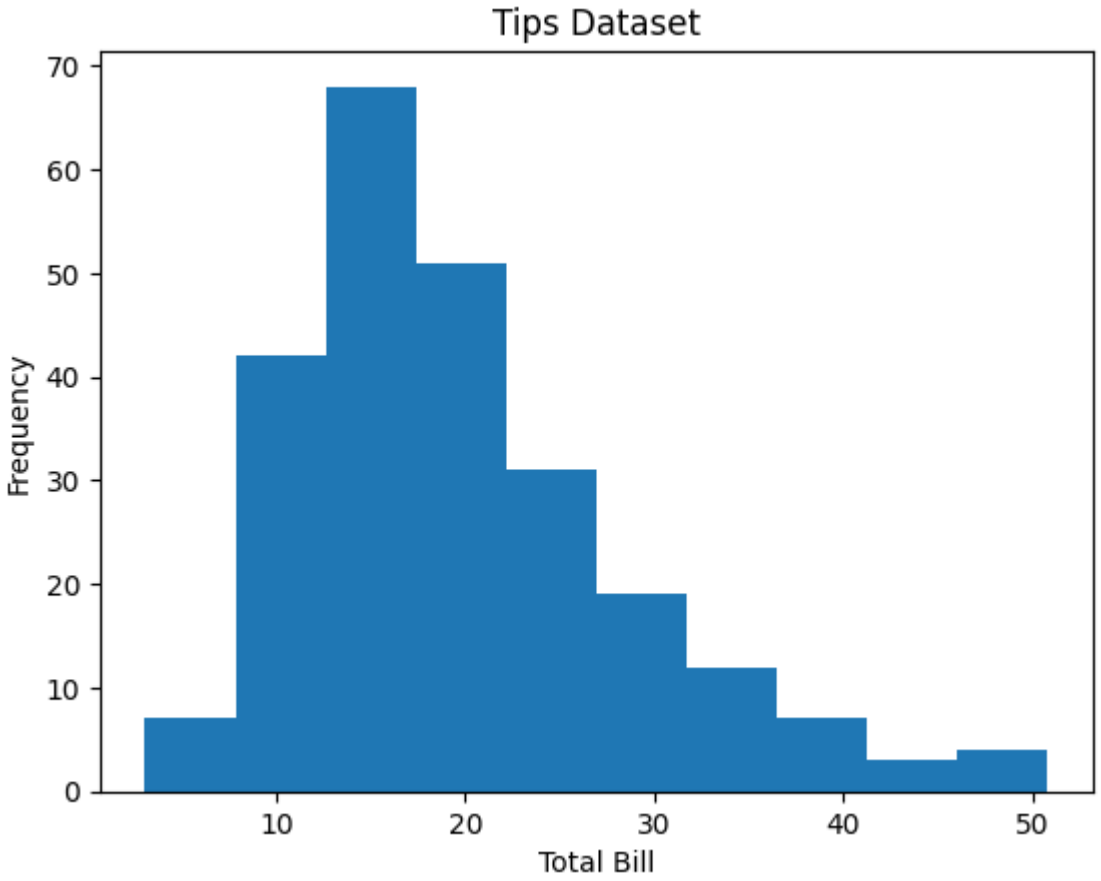


```
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
import pandas as pd

data = pd.read_csv("tip.csv")

x = data['day']
y = data['total_bill']

plt.bar(x, y)

plt.title("Tips Dataset")

plt.ylabel('Total Bill')

plt.xlabel('Day')

plt.show()
```




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

data = pd.read_csv('tip.csv')

x = data['total_bill']
```

```
plt.hist(x)


plt.title("Tips Dataset")
2)
plt.ylabel('Frequency')

plt.xlabel('Total Bill')

plt.show()
```

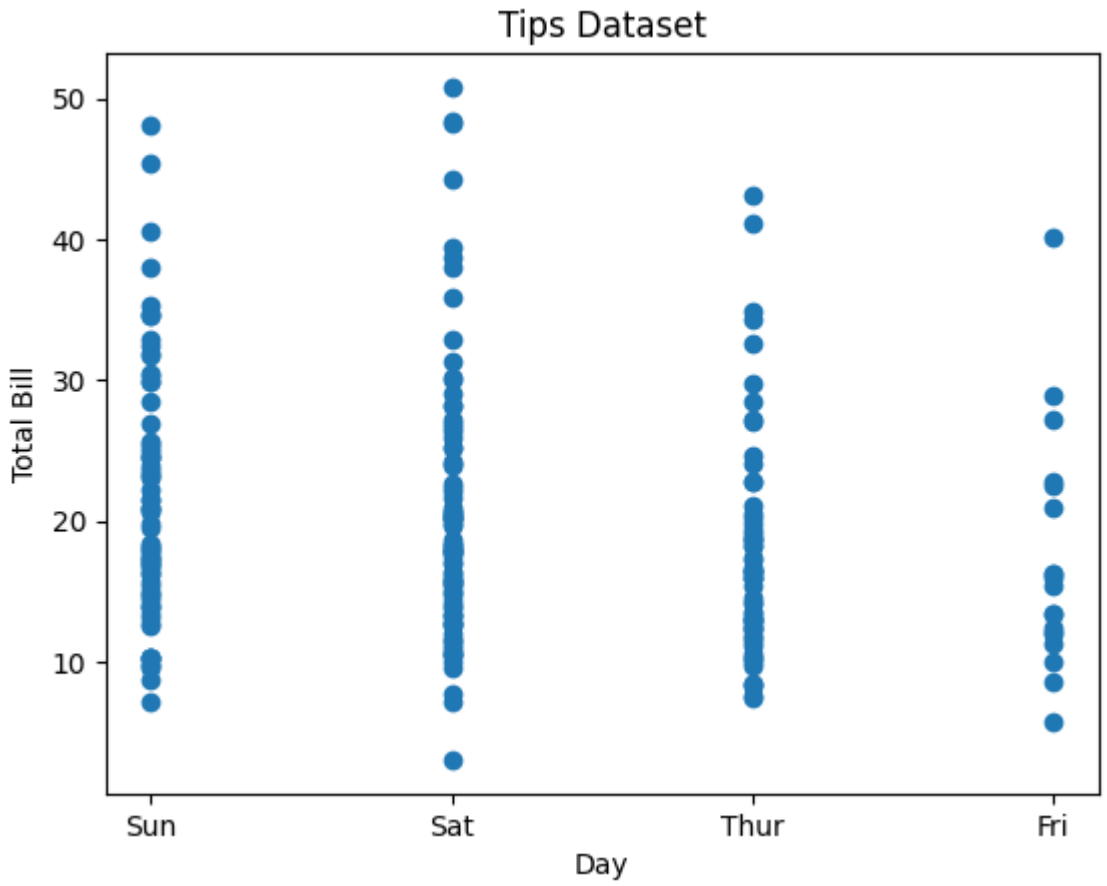


New Section

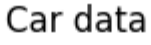
New Section

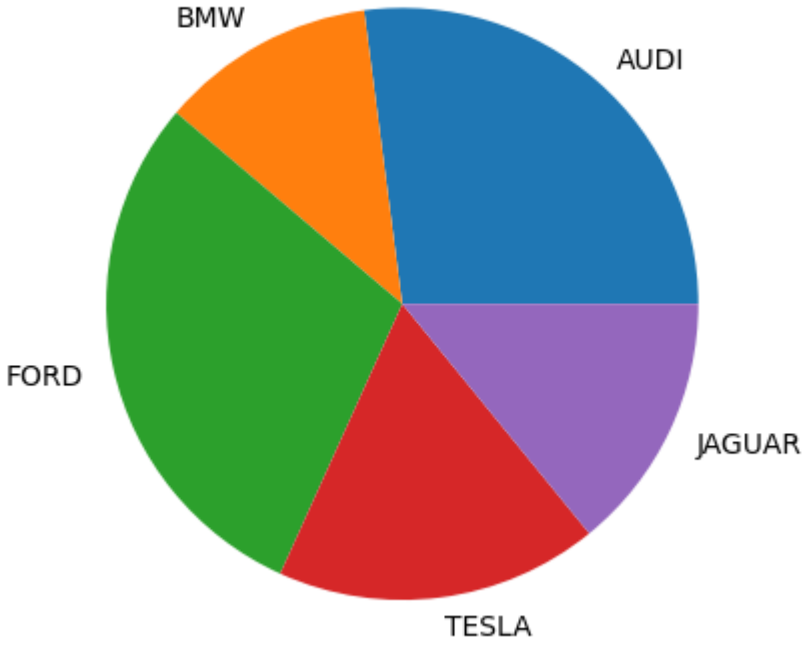
```
import matplotlib.pyplot as plt
import pandas as pd

data = pd.read_csv('tip.csv')

x = data['day']
y = data['total_bill']
```

```
plt.scatter(x, y)

plt.title("Tips Dataset")

plt.ylabel('Total Bill')

plt.xlabel('Day')

plt.show()
```

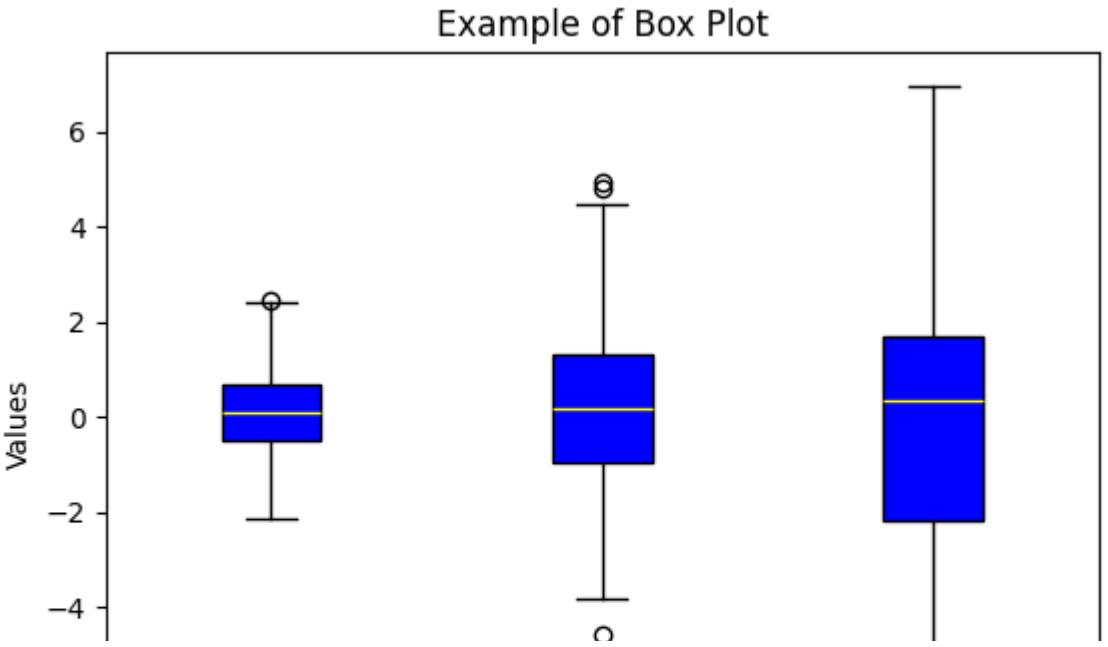



```
import matplotlib.pyplot as plt
import pandas as pd

data = pd.read_csv('tip.csv')

cars = ['AUDI', 'BMW', 'FORD',
        'TESLA', 'JAGUAR',]
data = [23, 10, 25, 15, 12]

plt.pie(data, labels=cars)

plt.title("Car data")

plt.show()
```

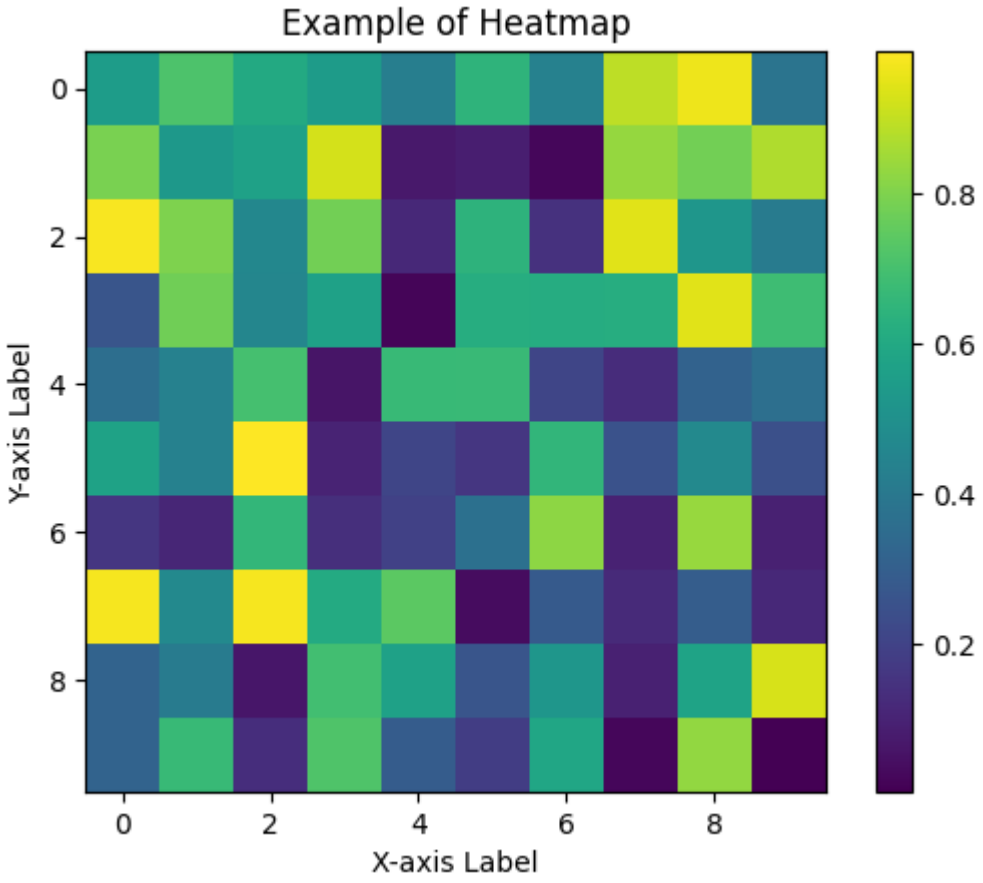





```
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(10)
data = [np.random.normal(0, std, 100) for std in range(1, 4)]

# Create a box plot
plt.boxplot(data, vert=True, patch_artist=True,
            boxprops=dict(facecolor='blue'),
            medianprops=dict(color='yellow'))

plt.xlabel('Data Set')
plt.ylabel('Values')
plt.title('Example of Box Plot')
plt.show()
```



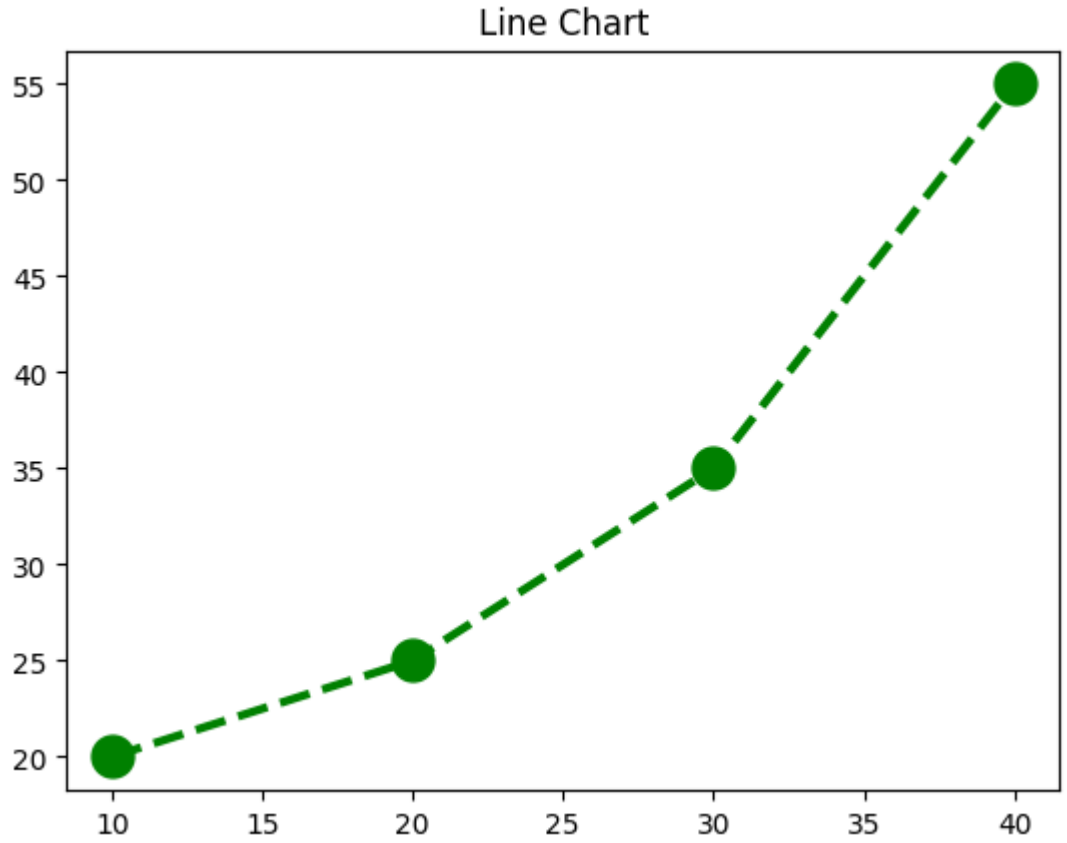

```
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(0)
data = np.random.rand(10, 10)

plt.imshow(data, cmap='viridis', interpolation='nearest')

plt.colorbar()

plt.xlabel('X-axis Label')
plt.ylabel('Y-axis Label')
plt.title('Example of Heatmap')

plt.show()
```




```
import matplotlib.pyplot as plt

x = [10, 20, 30, 40]
y = [20, 25, 35, 55]
```

```
plt.plot(x, y, color='green', linewidth=3, marker='o',
         markersize=15, linestyle='--')

plt.title("Line Chart")
```

```
Text(0.5, 1.0, 'Line Chart')
```



Start coding or generate with AI.

Practical 3:- Python program to display multiple types of charts using Matplotlib package

```
pip install matplotlib
```

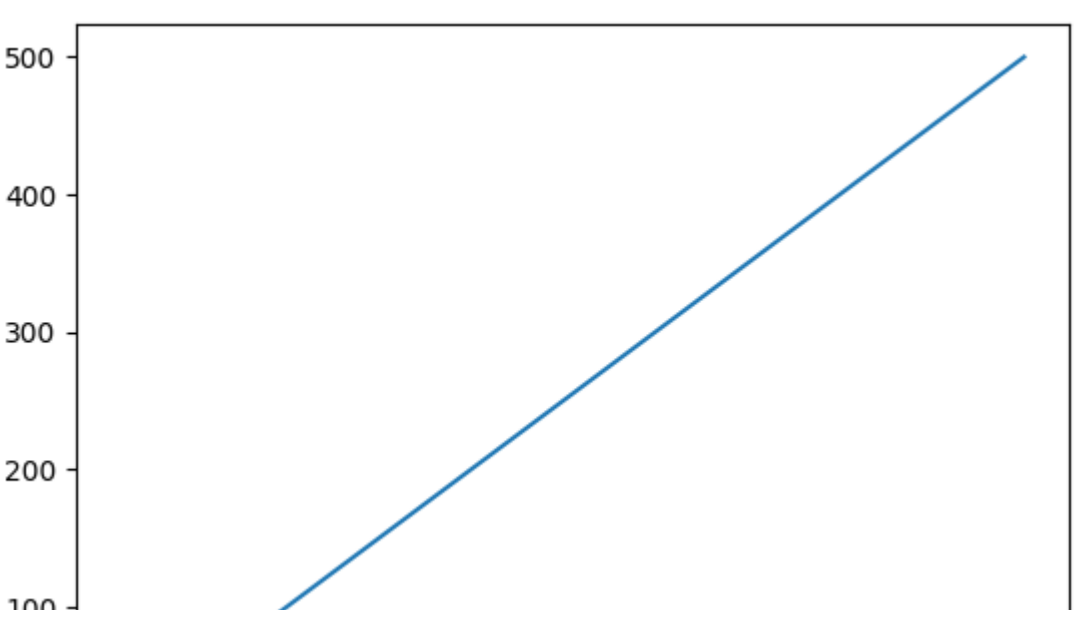
```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/di
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.
Requirement already satisfied: numpy>=1.23 in /usr/local/lib/python3.11/dis
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.1
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/pytho
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-p
```

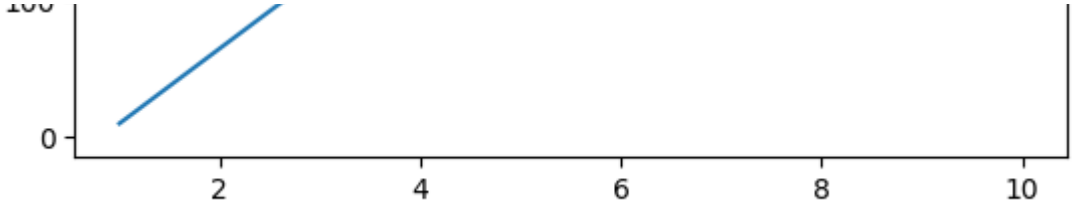
```
import matplotlib
print(matplotlib.__version__)
```

```
3.10.0
```

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

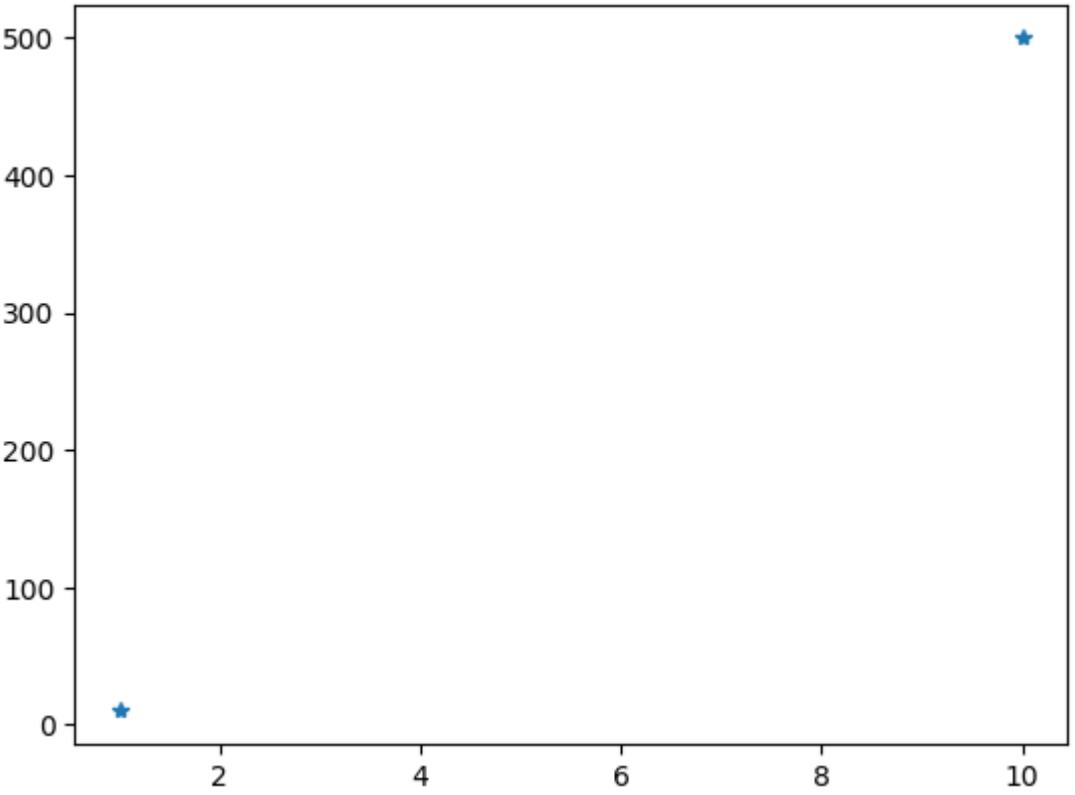
```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([1, 10])
ypoints = np.array([10, 500])

plt.plot(xpoints, ypoints)
plt.show()
```



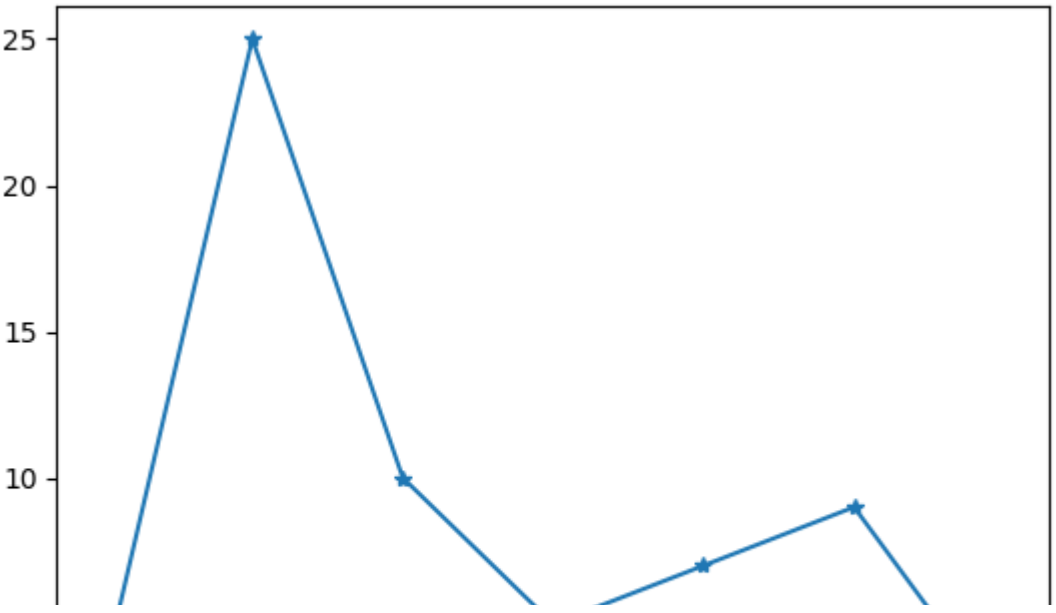


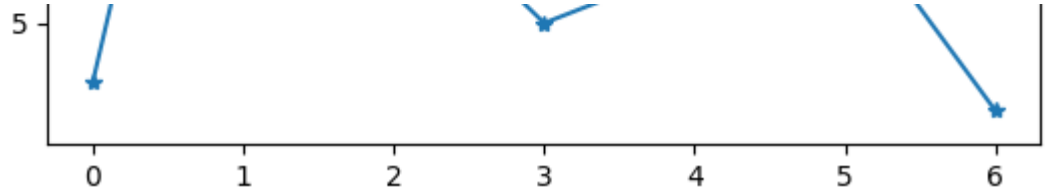

```
plt.plot(xpoints, ypoints, '*')
```

```
[<matplotlib.lines.Line2D at 0x7ff308774690>]
```



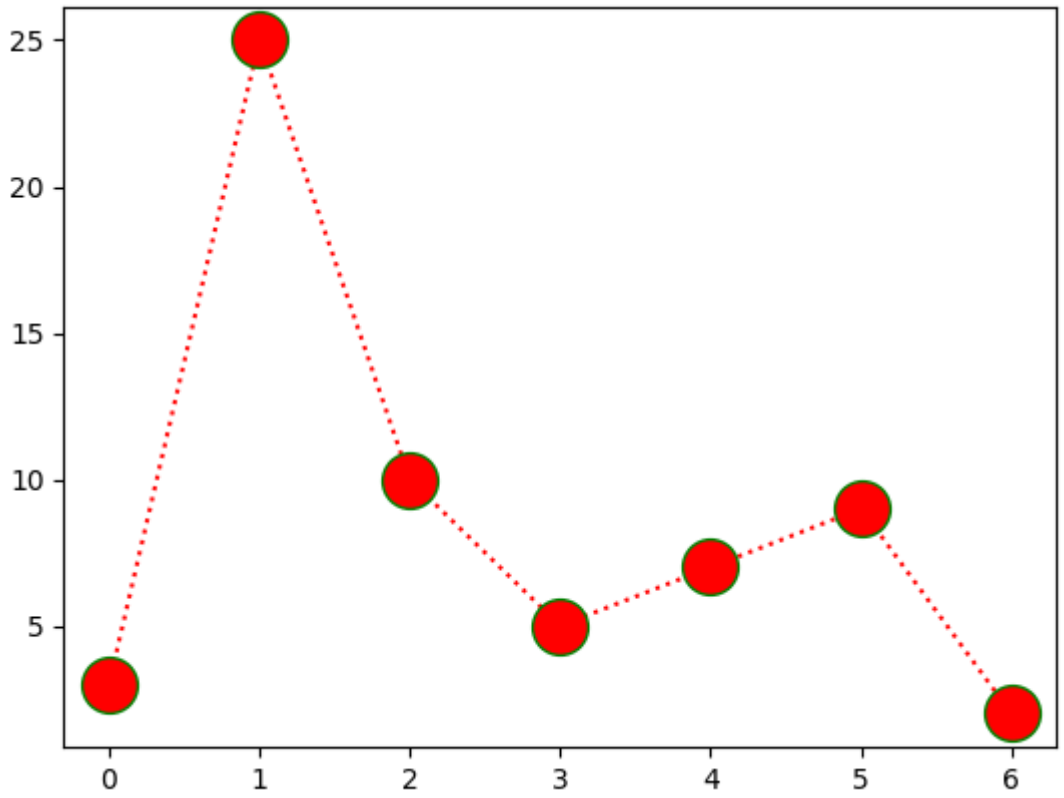

```
ypoints = np.array([3, 25, 10, 5, 7,9,2])
plt.plot(ypoints, marker = '*')
plt.show()
```



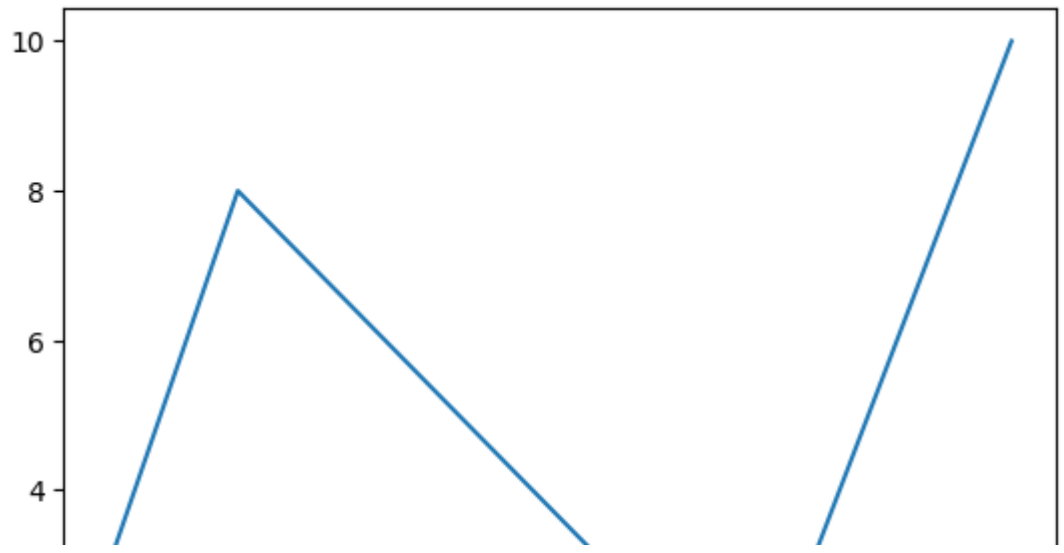


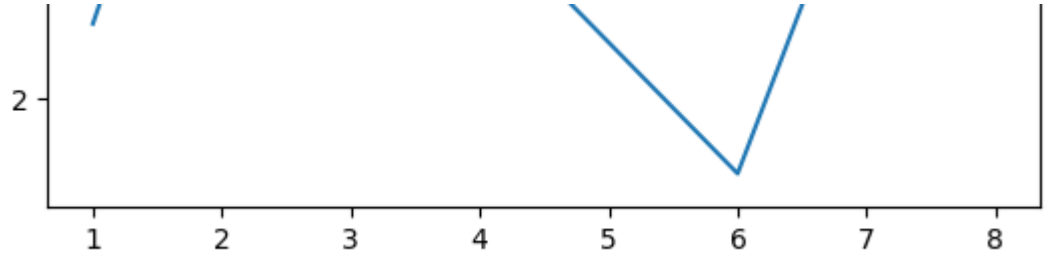

```
plt.plot(ypoints, 'o:r' ,ms = 20, mec = 'g')
```

```
[<matplotlib.lines.Line2D at 0x7ff325206150>]
```



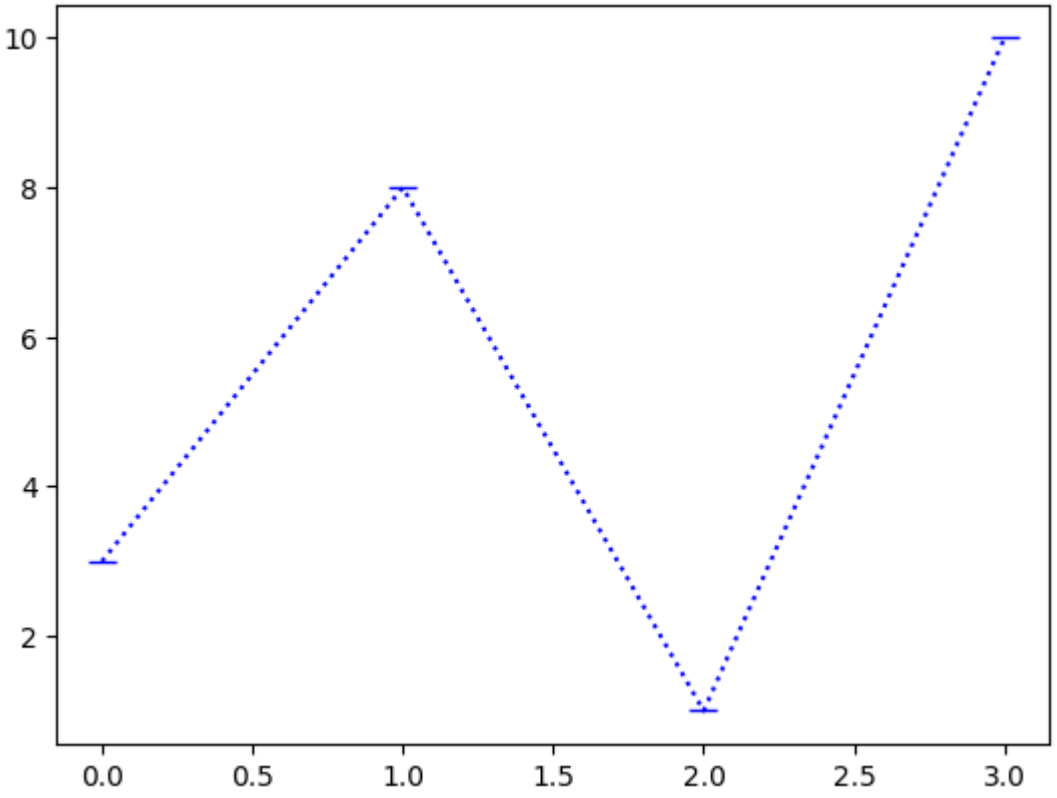

```
xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])

plt.plot(xpoints, ypoints)
plt.show()
```





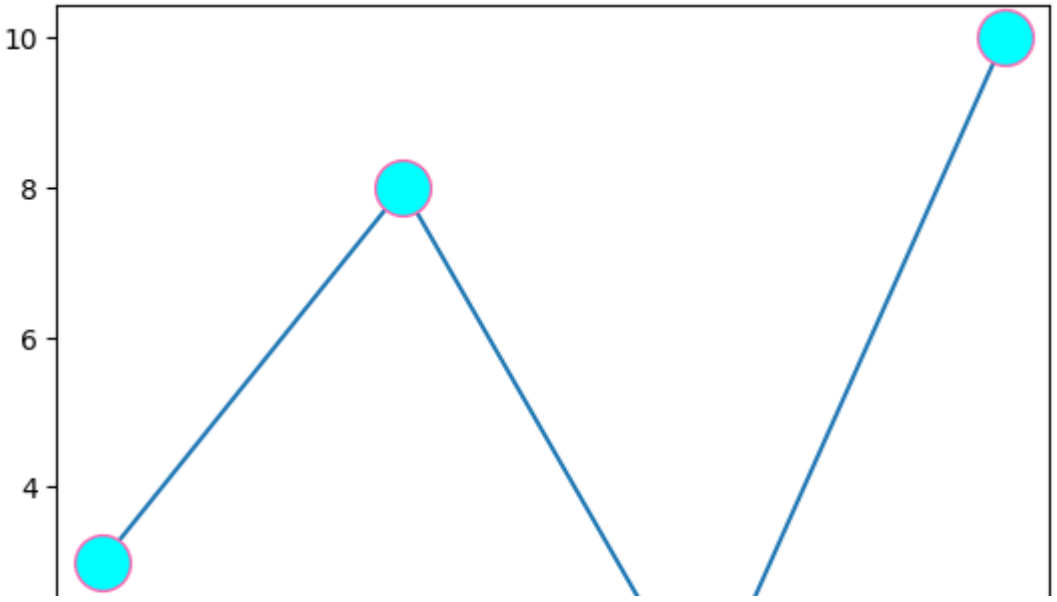

```
plt.plot(ypoints, '_:b' , ms = 10)
```

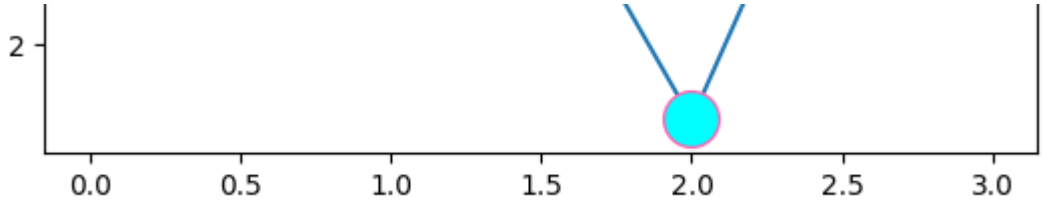
```
[<matplotlib.lines.Line2D at 0x7ff308661ad0>]
```



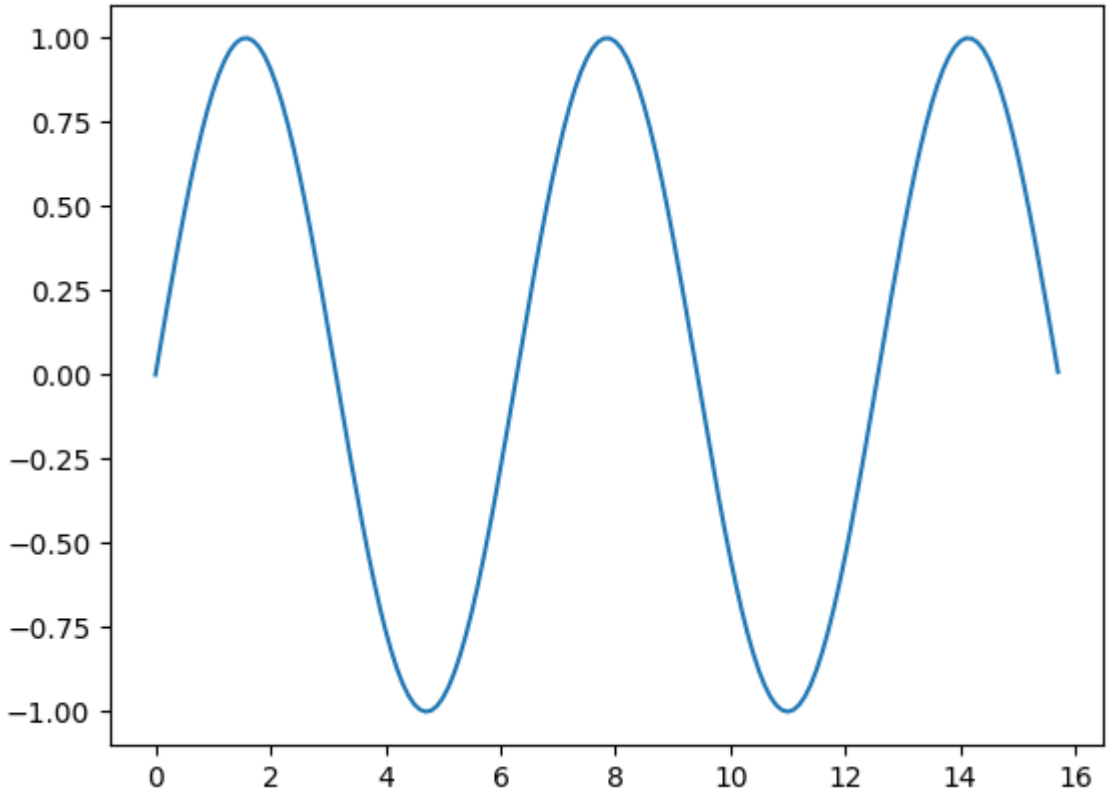

```
plt.plot(ypoints, marker = 'o', ms = 20, mec = 'hotpink', mfc = 'cyan')
```

```
[<matplotlib.lines.Line2D at 0x7ff3086efe10>]
```

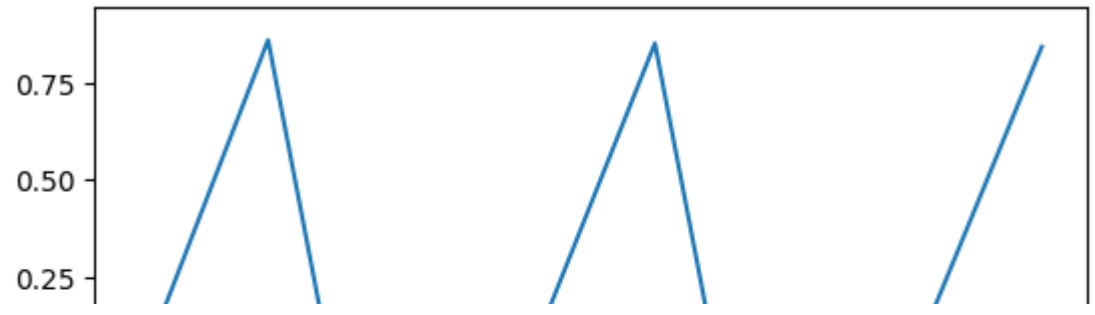


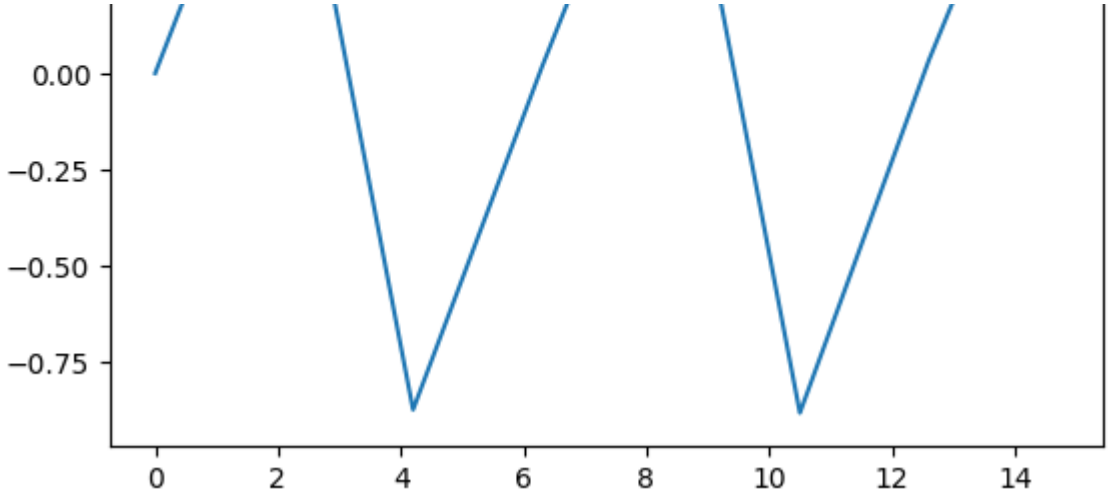



```
import matplotlib.pyplot as plt
import numpy as np
x=np.arange(0,5*np.pi,0.1);
y=np.sin(x);
plt.plot(x,y);
plt.show;
```

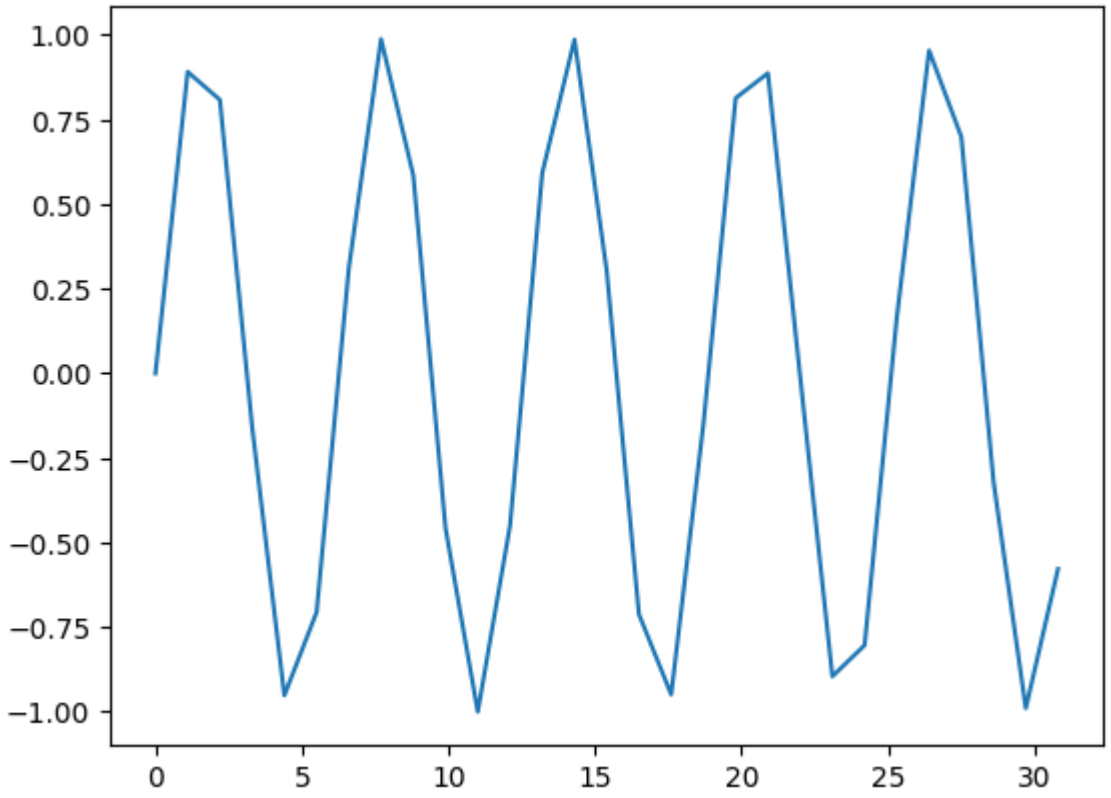



```
x=np.arange(0,5*np.pi,2.1);
y=np.sin(x);
plt.plot(x,y);
plt.show;
```

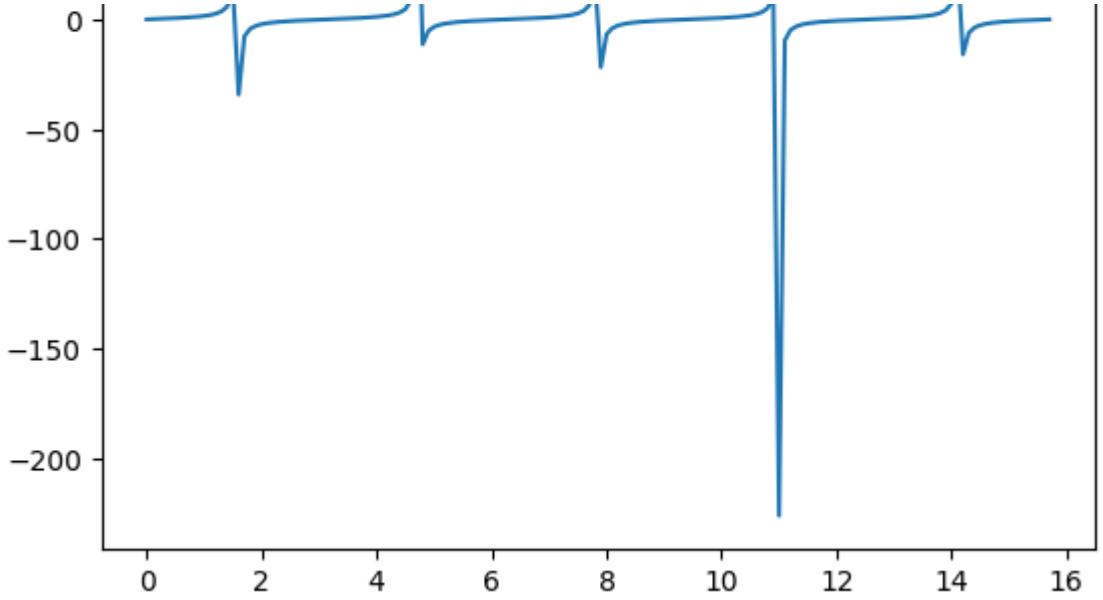





```
x=np.arange(0,10*np.pi,1.1);
y=np.sin(x);
plt.plot(x,y);
plt.show;
```

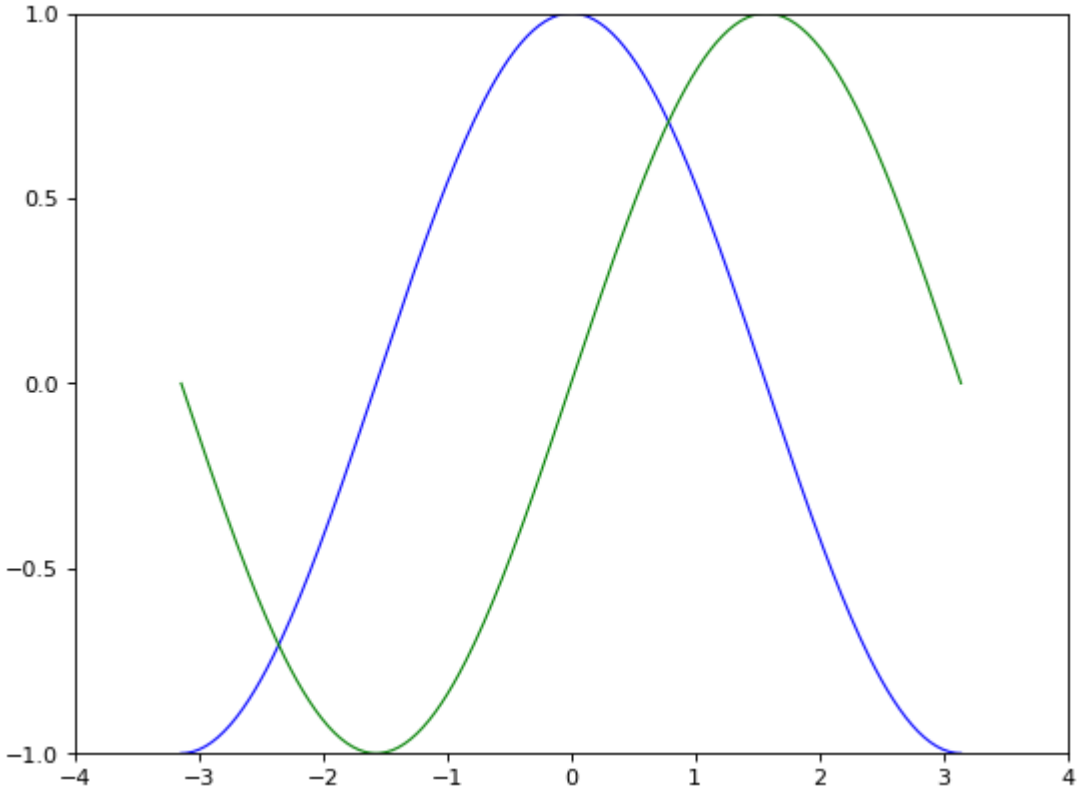



```
x=np.arange(0,5*np.pi,0.1);
y=np.tan(x);
plt.plot(x,y);
plt.show;
```

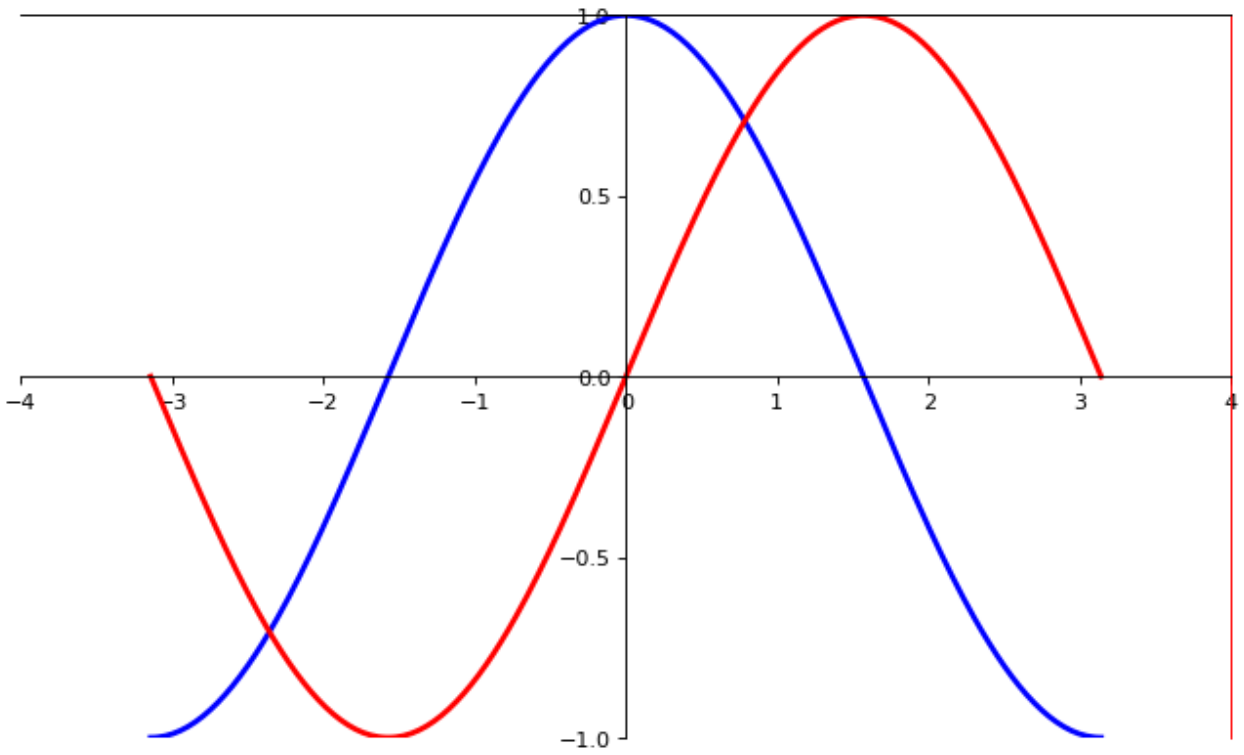


Start coding or generate with AI.

```
plt.figure(figsize=(8,6), dpi=80)
plt.subplot(1, 1, 1)
X = np.linspace(-np.pi, np.pi, 256)
C, S = np.cos(X), np.sin(X)
plt.plot(X, C, color="blue", linewidth=1.0, linestyle="-")
plt.plot(X, S, color="green", linewidth=1.0, linestyle="-")
plt.xlim(-4.0, 4.0)
plt.xticks(np.linspace(-4, 4, 9))
plt.ylim(-1.0, 1.0)
plt.yticks(np.linspace(-1, 1, 5))
plt.show()
```

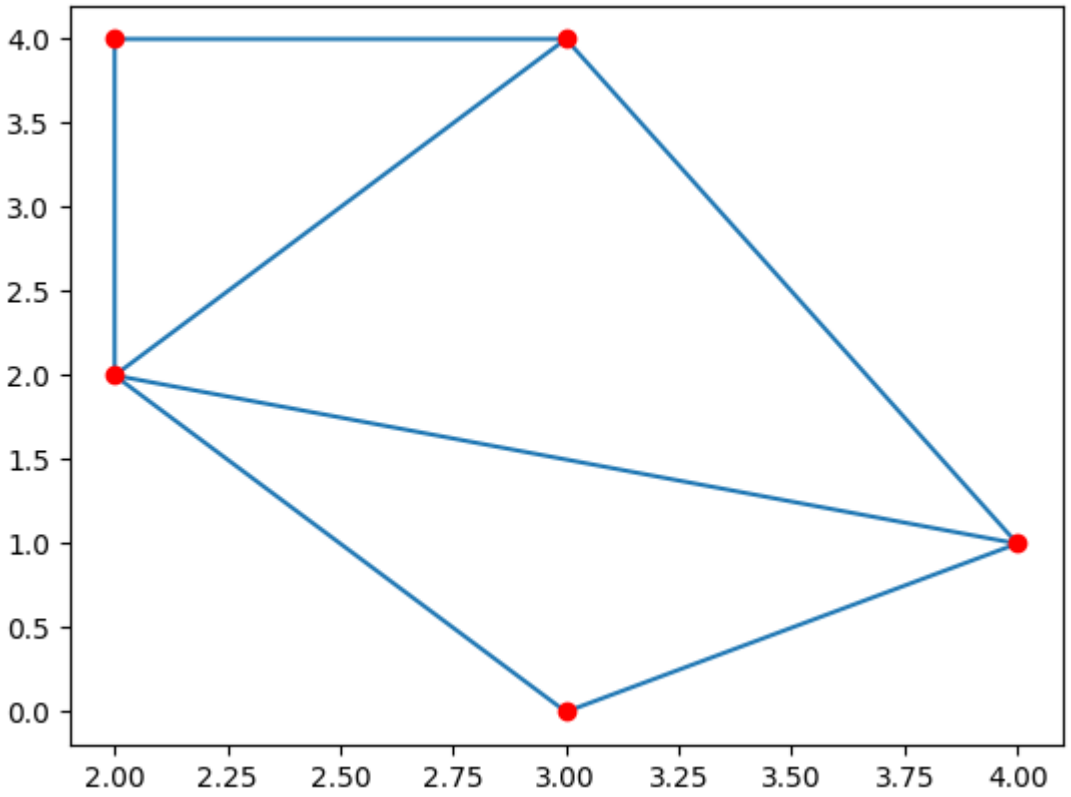


```
plt.figure(figsize=(10, 6), dpi=80)
plt.subplot(1, 1, 1)
X = np.linspace(-np.pi, np.pi, 256)
C, S = np.cos(X), np.sin(X)
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
plt.plot(X, S, color="RED", linewidth=2.5, linestyle="-")
plt.xlim(-4.0, 4.0)
plt.xticks(np.linspace(-4, 4, 9))
plt.ylim(-1.0, 1.0)
plt.yticks(np.linspace(-1, 1, 5))

ax = plt.gca() # gca stands for get current axis;
ax.spines['right'].set_color('red')
ax.spines['top'].set_color('black')
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))



plt.show()
```

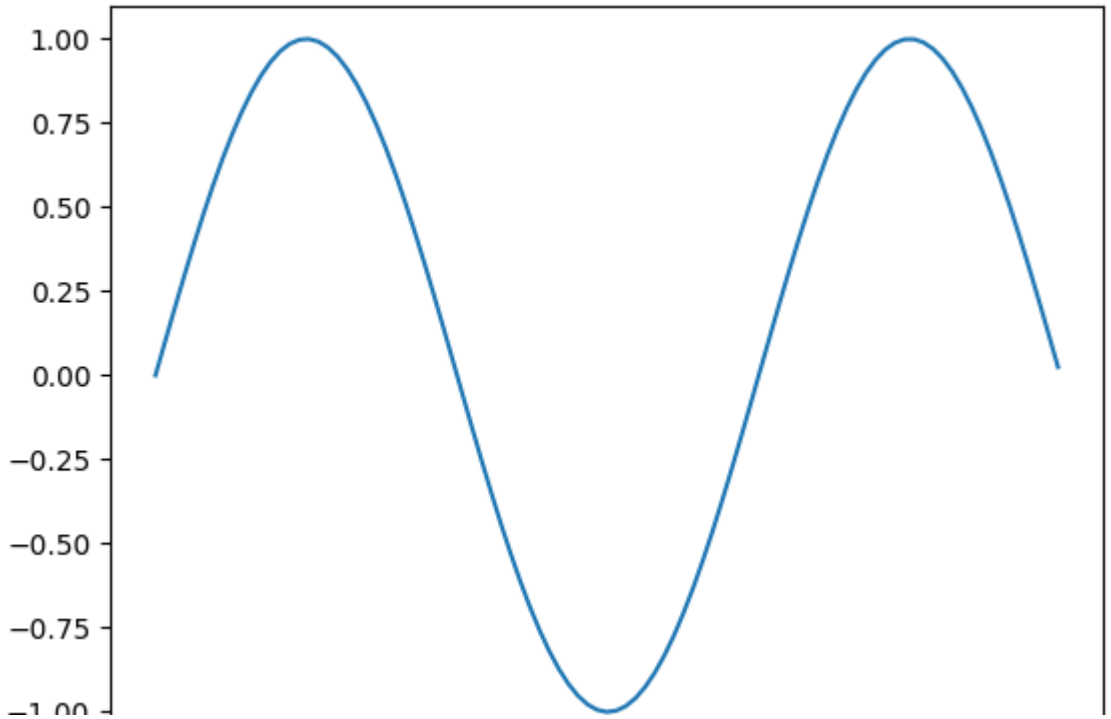


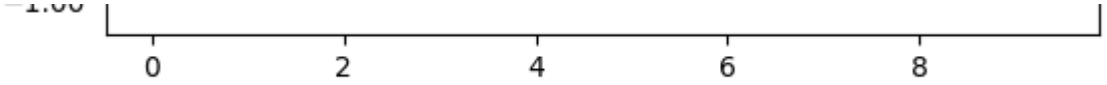

```
import numpy as np
from scipy.spatial import Delaunay
import matplotlib.pyplot as plt


points= np.array([
    [2 4] [3 4] [3 0] [2 2] [4 1]
```

```
    [2,4],[3,4],[3,0],[2,2],[4,1]
])
simplices = Delaunay(points).simplices
plt.triplot(points[:,0],points[:,1],simplices)
plt.scatter(points[:,0],points[:,1],color='r')
plt.show()
```

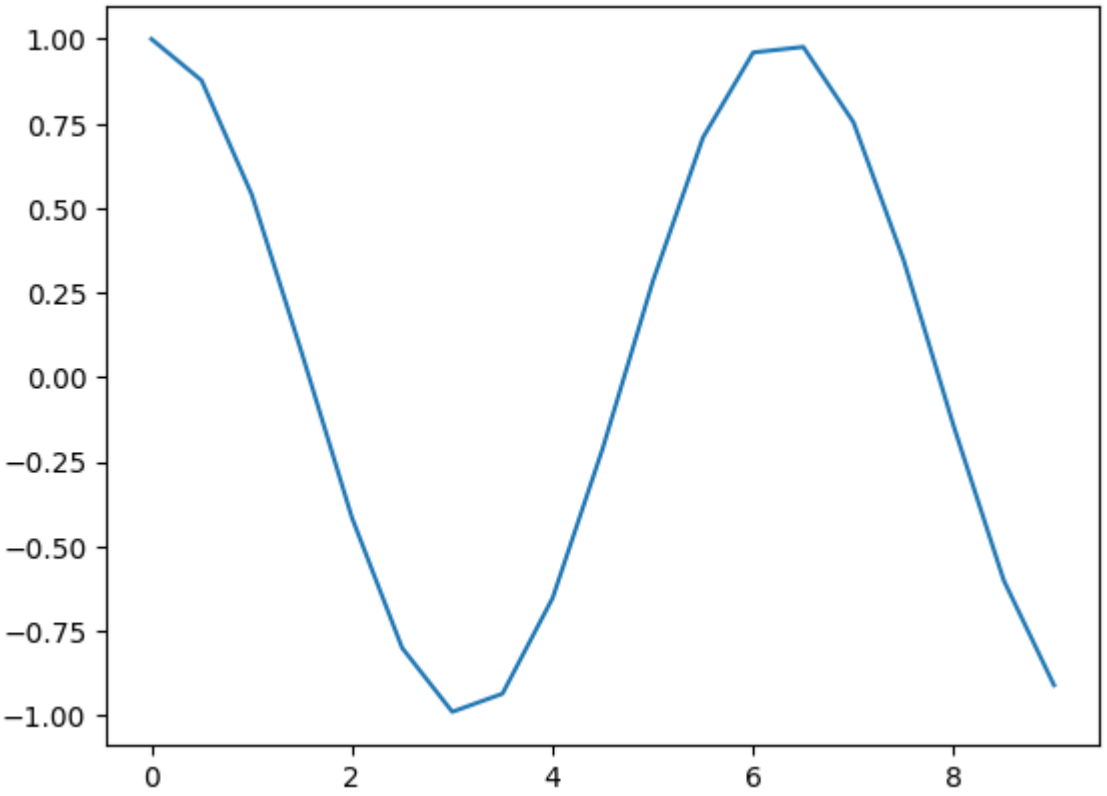



```
x=np.arange(0,3* np.pi, 0.1)
y=np.sin(x)
plt.plot(x,y)
plt.show;
```

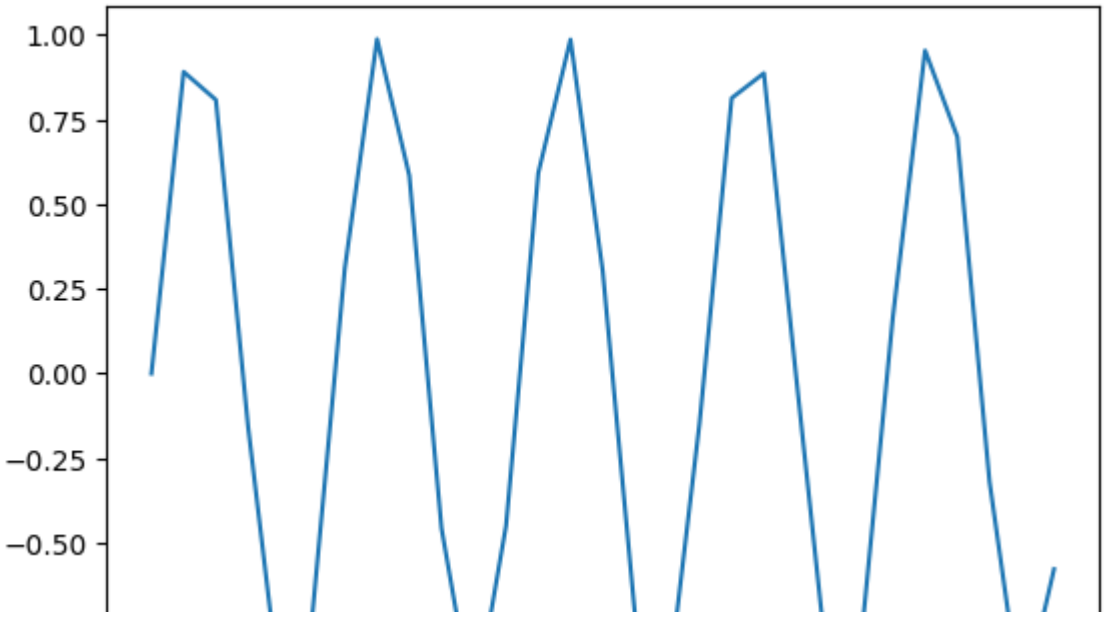


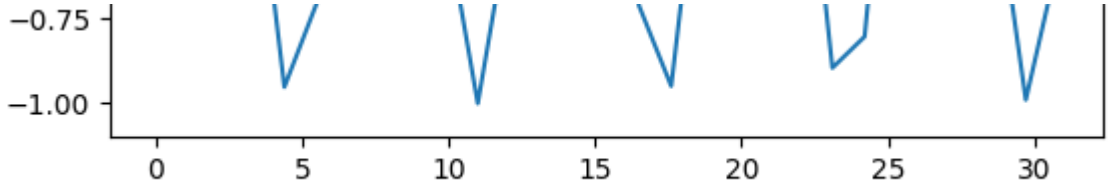



```
x=np.arange(0,3* np.pi, 0.5)
y=np.cos(x)
plt.plot(x,y)
plt.show()
```

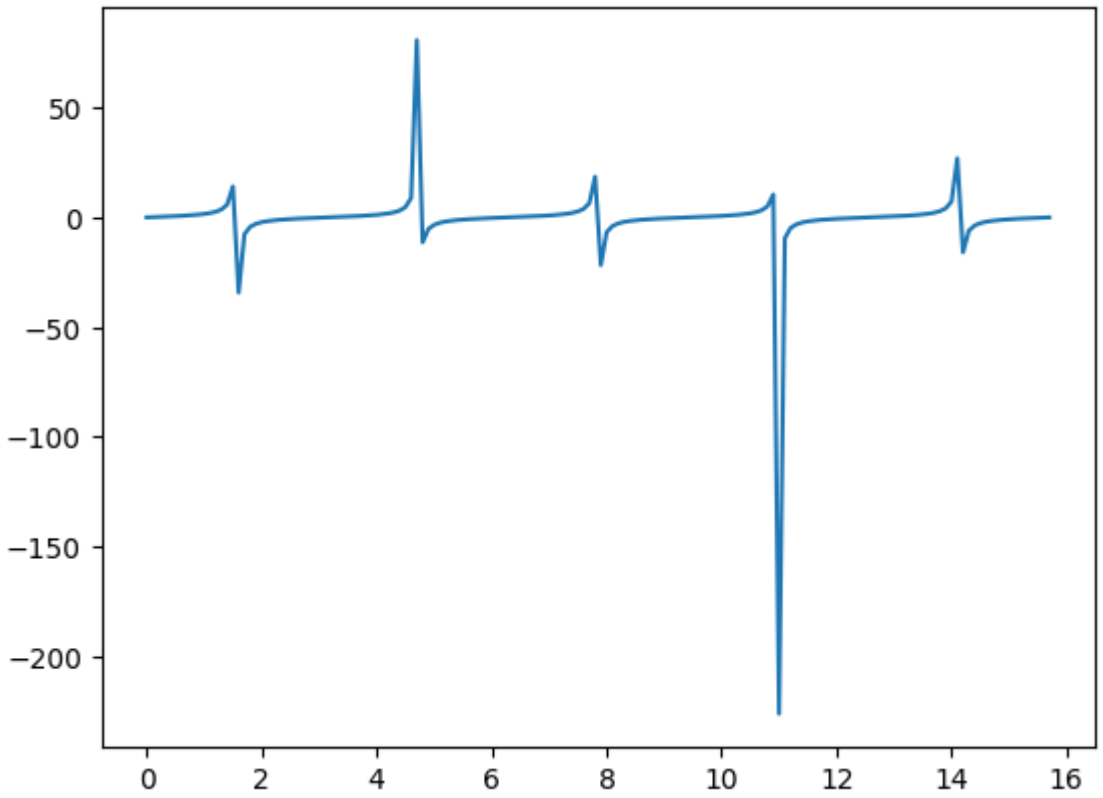



```
x=np.arange(0,10*np.pi,1.1);
y=np.sin(x);
plt.plot(x,y);
plt.show;
```

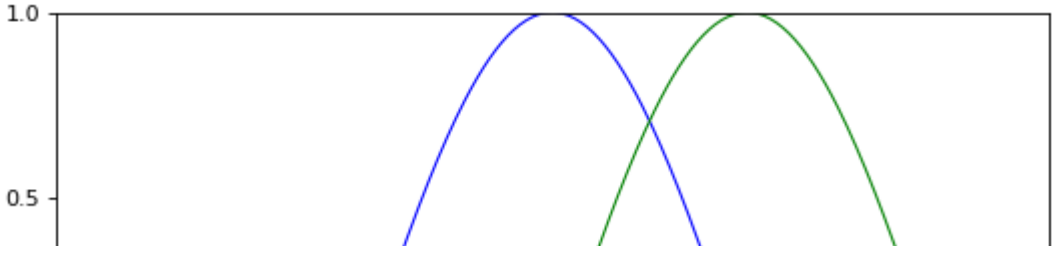


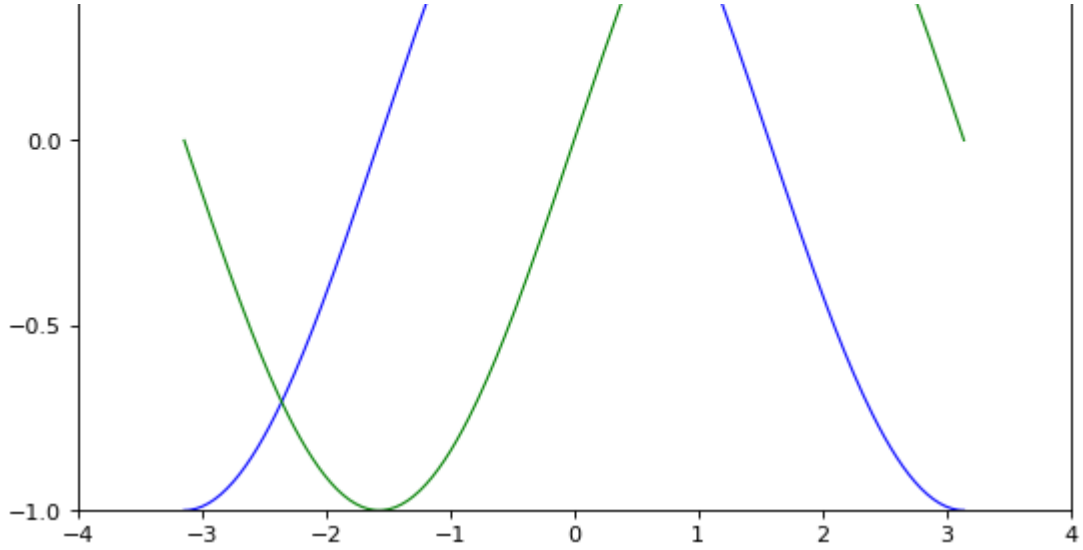



```
x=np.arange(0,5* np.pi, 0.1)
y=np.tan(x)
plt.plot(x,y)
plt.show()
```

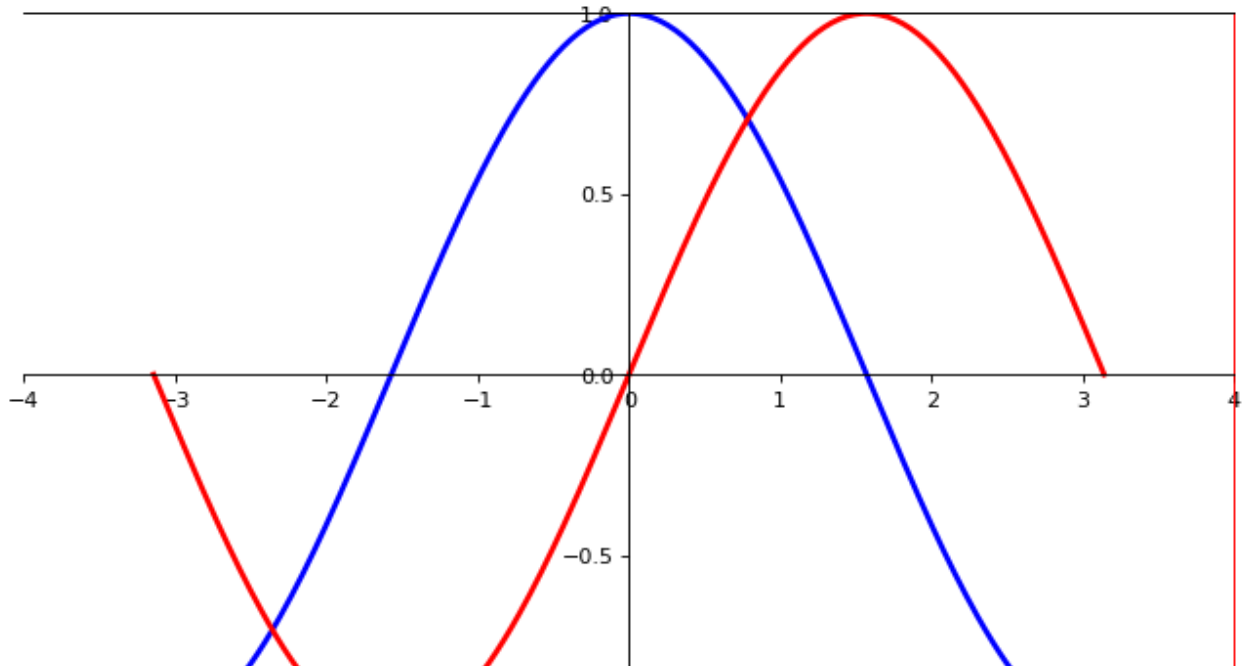



```
plt.figure(figsize=(8,6), dpi=80)
plt.subplot(1,1,1)
X= np.linspace(-np.pi, np.pi,256)
C,S= np.cos(X),np.sin(X)
plt.plot(X,C, color='blue', linewidth=1.0, linestyle='-')
plt.plot(X,S,color='green', linewidth=1.0, linestyle='-')
plt.xlim(-4.0,4.0)
plt.xticks(np.linspace(-4,4,9))
plt.ylim(-1.0,1.0)
plt.yticks(np.linspace(-1,1,5))
plt.show()
```





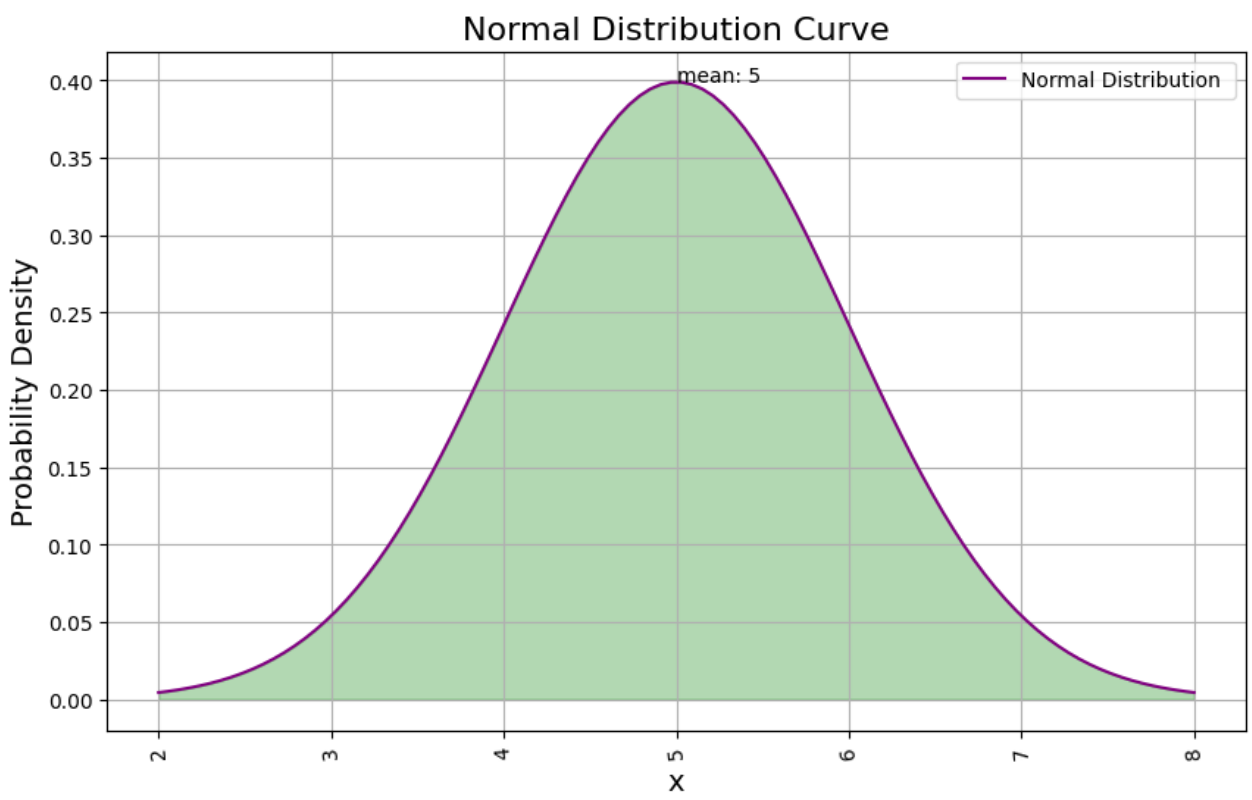

```
plt.figure(figsize=(10, 6), dpi=80)
plt.subplot(1, 1, 1)
X = np.linspace(-np.pi, np.pi, 256)
C, S = np.cos(X), np.sin(X)
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-")
plt.plot(X, S, color="RED", linewidth=2.5, linestyle="-")
plt.xlim(-4.0, 4.0)
plt.xticks(np.linspace(-4, 4, 9))
plt.ylim(-1.0, 1.0)
plt.yticks(np.linspace(-1, 1, 5))

ax = plt.gca() # gca stands for get current axis;
ax.spines['right'].set_color('red')
ax.spines['top'].set_color('black')
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))
plt.show()
```



Practical - 4

Aim: Create a python program that generates and display a normal distribution curve.

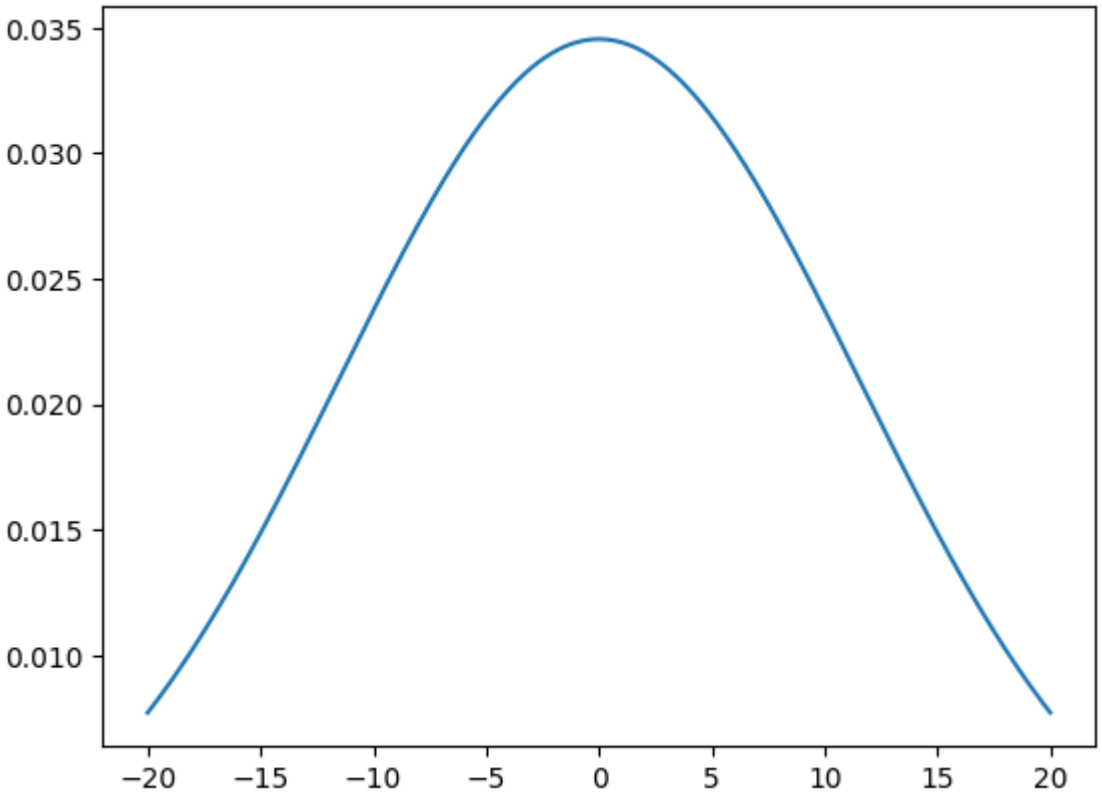
```
mean = 5
sigma = 1
x = np.linspace(mean - 3*sigma, mean  + 3*sigma, 100)
y = norm.pdf(x, mean , sigma)
plt.figure(figsize=(10, 6))
plt.plot(x, y, color='purple', label='Normal Distribution Curve')
plt.title('Normal Distribution Curve', fontsize=16)
plt.xlabel('x', fontsize=14)
plt.ylabel('Probability Density', fontsize=14)
plt.grid(True)
plt.legend(['Normal Distribution '])

plt.fill_between(x, y, alpha=0.3,color='green')
plt.text(mean , max(y), f'mean: {mean}')
#plt.axvline(x=0, color='r', linestyle='--')
#plt.axhline(y=0.5, color='r', linestyle='--')
plt.xticks(rotation=95)
plt.show()
```



```
x = random.normal(size=(2,3))
print(x)
```

```
[[ 0.60454964  0.45127951  0.64501106]
 [-0.33336061  0.42675764  0.45312146]]
```

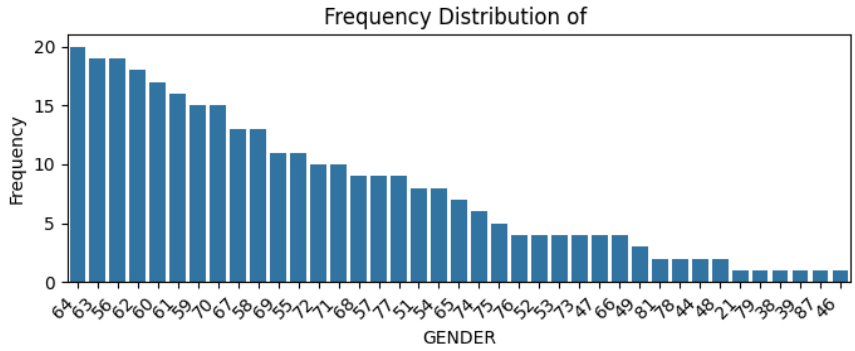
```
from numpy import random
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
import statistics
import seaborn as sns

x_axis = np.arange(-20, 20, 0.01)
mean = statistics.mean(x_axis)
sd = statistics.stdev(x_axis)
plt.plot(x_axis, norm.pdf(x_axis, mean, sd))
plt.show()
```



Practical 5: Write a Python program that calculates and displays a frequency distribution of a given dataset.

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



df = pd.read_csv('lung_cancer.csv')
frequency_distribution = df['GENDER'].value_counts()
print(frequency_distribution)

plt.figure(figsize=(7, 3))
sns.countplot(x='AGE', data=df, order=df['AGE'].value_counts().index)
plt.xticks(rotation=45, ha='right')
plt.title('Frequency Distribution of ')
plt.xlabel('GENDER')
plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```

```
GENDER
M    162
F    147
Name: count, dtype: int64
```

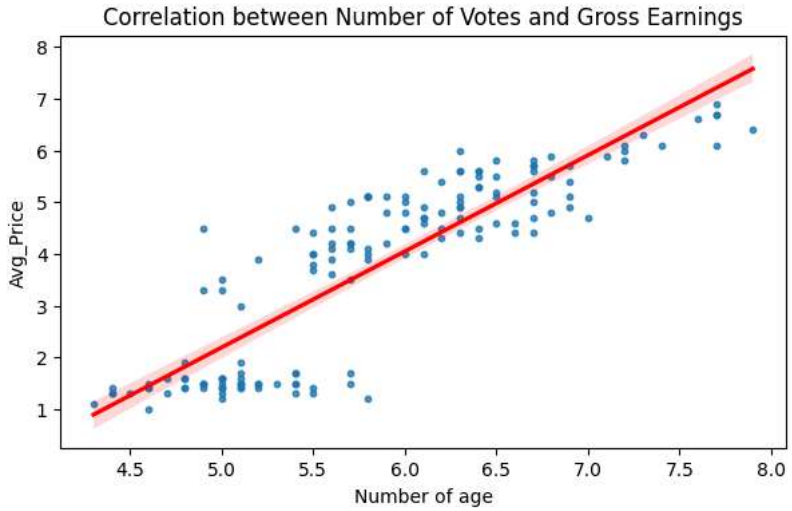


Start coding or generate with AI.

Practical 6: Write a Python program that computes the correlation between two sets of data and visualizes the relationship using a scatter plot.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('Iris.csv')
print(df.head())

# Calculate the Pearson correlation coefficient between 'num_voted_users' and 'gross'
correlation = df['SepalLengthCm'].corr(df['PetalLengthCm'])

print(f'Pearson correlation coefficient: {correlation}')

plt.figure(figsize=(7, 4))
sns.regplot(x='SepalLengthCm', y='PetalLengthCm', data=df, scatter_kws={'s': 10}, line_kws={'color': 'red'})

plt.title('Correlation between Number of Votes and Gross Earnings')
plt.xlabel('Number of age')
plt.ylabel('Avg_Price')

plt.show()
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
Pearson correlation coefficient: 0.8717541573048718
```



Practical 7: Develop a Python program to calculate the correlation coefficient between two sets of data.

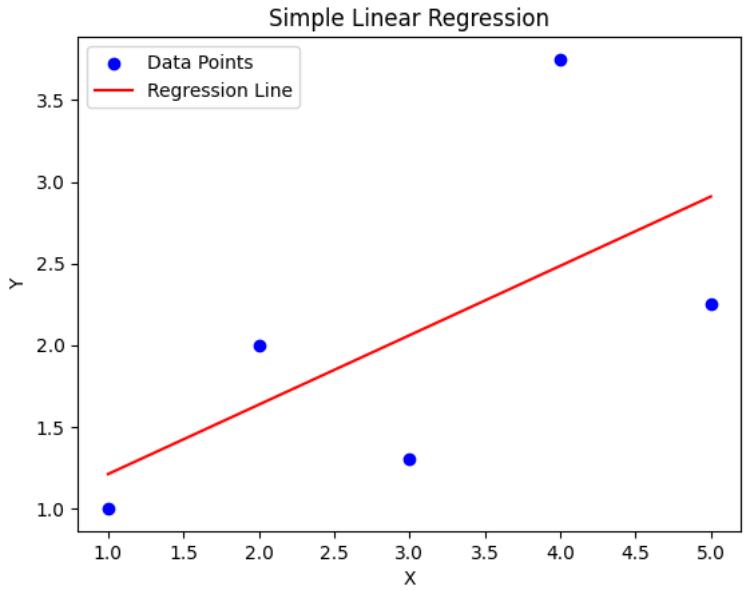
```
import numpy as np

def calculate_correlation(x, y):
    if len(x) != len(y):
        raise ValueError("Both datasets must have the same length")

    # Calculate means
    mean_x = np.mean(x)
    mean_y = np.mean(y)

    # Calculate covariance
    covariance = np.sum((x - mean_x) * (y - mean_y))

    # Calculate standard deviations
    std_x = np.sqrt(np.sum((x - mean_x) ** 2))  # Corrected: squared differences of x
    std_y = np.sqrt(np.sum((y - mean_y) ** 2))  # Corrected: squared differences of y

    # Calculate correlation coefficient
    correlation = covariance / (std_x * std_y)

    return correlation

# New example datasets
x_data = [10, 15, 20, 25, 30]
y_data = [1, 2, 3, 4, 5]

# Calculate and print correlation
correlation_coefficient = calculate_correlation(np.array(x_data), np.array(y_data))
print(f"Correlation Coefficient: {correlation_coefficient:.2f}")
```

```
Correlation Coefficient: 1.00
```

Practical 8: Develop a Python program that performs Simple Linear Regression to model the relationship between two variables.

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

# Function to calculate the slope (m) and intercept (b)
def simple_linear_regression(x, y):
    # Calculate the means of x and y
    mean_x = np.mean(x)
    mean_y = np.mean(y)

    # Calculate the slope (m)
    m = np.sum((x - mean_x) * (y - mean_y)) / np.sum((x - mean_x) ** 2)

    # Calculate the intercept (b)
    b = mean_y - m * mean_x

    return m, b

# Function to predict y values using the regression line
def predict(x, m, b):
    return m * x + b

# Example datasets (x - independent variable, y - dependent variable)
x_data = np.array([1, 2, 3, 4, 5])
y_data = np.array([1, 2, 1.3, 3.75, 2.25])

# Perform Simple Linear Regression
m, b = simple_linear_regression(x_data, y_data)

# Print the regression line equation
print(f"Regression line: y = {m:.2f}x + {b:.2f}")

# Predict the y values using the regression line
y_pred = predict(x_data, m, b)

# Plot the original data points and the regression line
plt.scatter(x_data, y_data, color='blue', label='Data Points')
plt.plot(x_data, y_pred, color='red', label='Regression Line')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Simple Linear Regression')
plt.legend()
plt.show()
```

```
Regression line: y = 0.42x + 0.79
```



9. Write a Python program to convert data (e.g., JSON, CSV, XML, etc.) from one format to another.

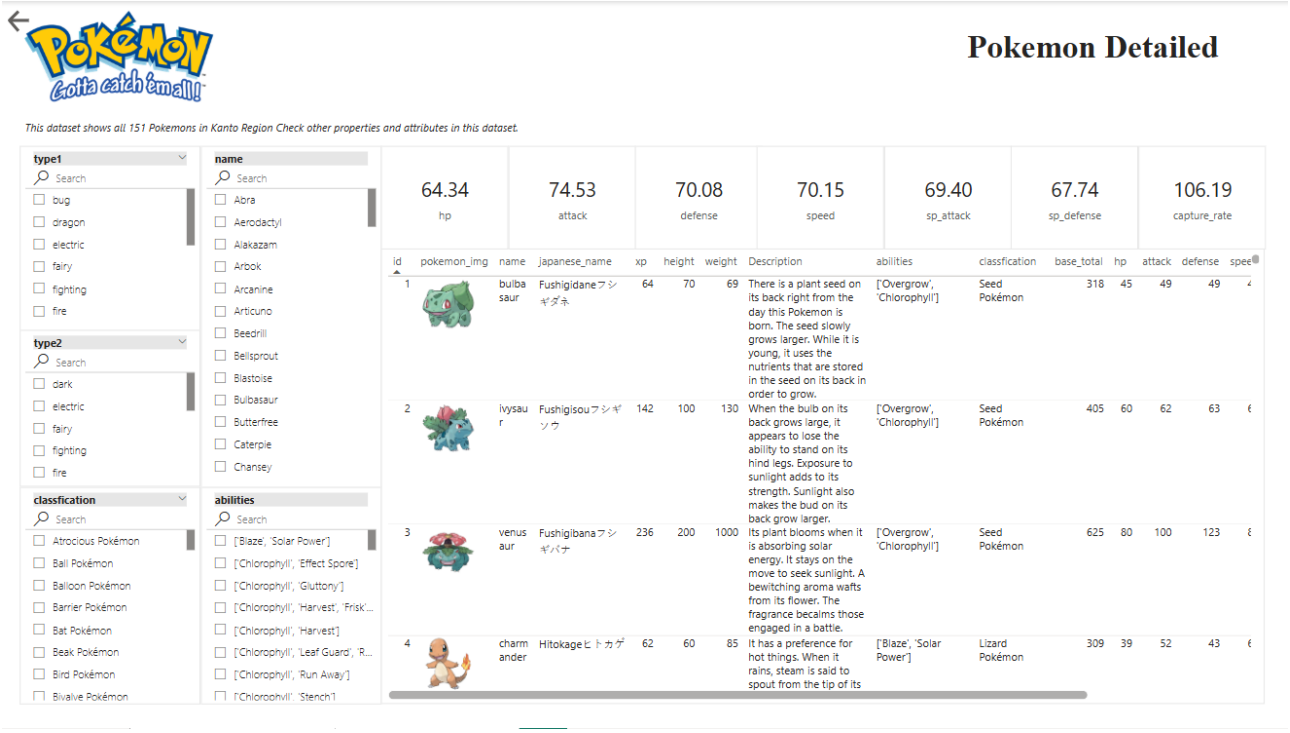
```
import json
import csv
import xml.etree.ElementTree as ET

# Convert JSON to CSV
def json_to_csv(json_data, csv_filename):
    # Convert JSON data to Python list of dictionaries
    data = json.loads(json_data)

    # Get the keys from the first item in the list (to use as header)
    keys = data[0].keys()

    # Write to CSV file
    with open(csv_filename, mode='w', newline='') as file:
        writer = csv.DictWriter(file, fieldnames=keys)
        writer.writeheader()
        writer.writerows(data)
    print(f"Data successfully written to {csv_filename}")

# Convert CSV to JSON
def csv_to_json(csv_filename):
    # Read CSV file and convert to list of dictionaries
    with open(csv_filename, mode='r') as file:
        reader = csv.DictReader(file)
        data = list(reader)

    # Convert to JSON
    json_data = json.dumps(data, indent=4)
    return json_data

# Convert JSON to XML
def json_to_xml(json_data):
    data = json.loads(json_data)

    # Create the root element of XML
    root = ET.Element("root")

    # Add child elements to root
    for entry in data:
        entry_elem = ET.SubElement(root, "entry")
        for key, value in entry.items():
            child_elem = ET.SubElement(entry_elem, key)
            child_elem.text = str(value)

    # Convert tree to string
    xml_data = ET.tostring(root, encoding='unicode', method='xml')
    return xml_data

# Convert XML to JSON
def xml_to_json(xml_data):
    root = ET.fromstring(xml_data)

    data = []
    for entry_elem in root.findall('entry'):
        entry = {}
        for child_elem in entry_elem:
            entry[child_elem.tag] = child_elem.text
        data.append(entry)

    # Convert to JSON
    json_data = json.dumps(data, indent=4)
    return json_data


# Example usage:

# JSON data as a string
json_data = '''[
    {"name": "Alice", "age": 25, "city": "New York"},
    {"name": "Bob", "age": 30, "city": "Los Angeles"},
    {"name": "Charlie", "age": 35, "city": "Chicago"}
]'''
```

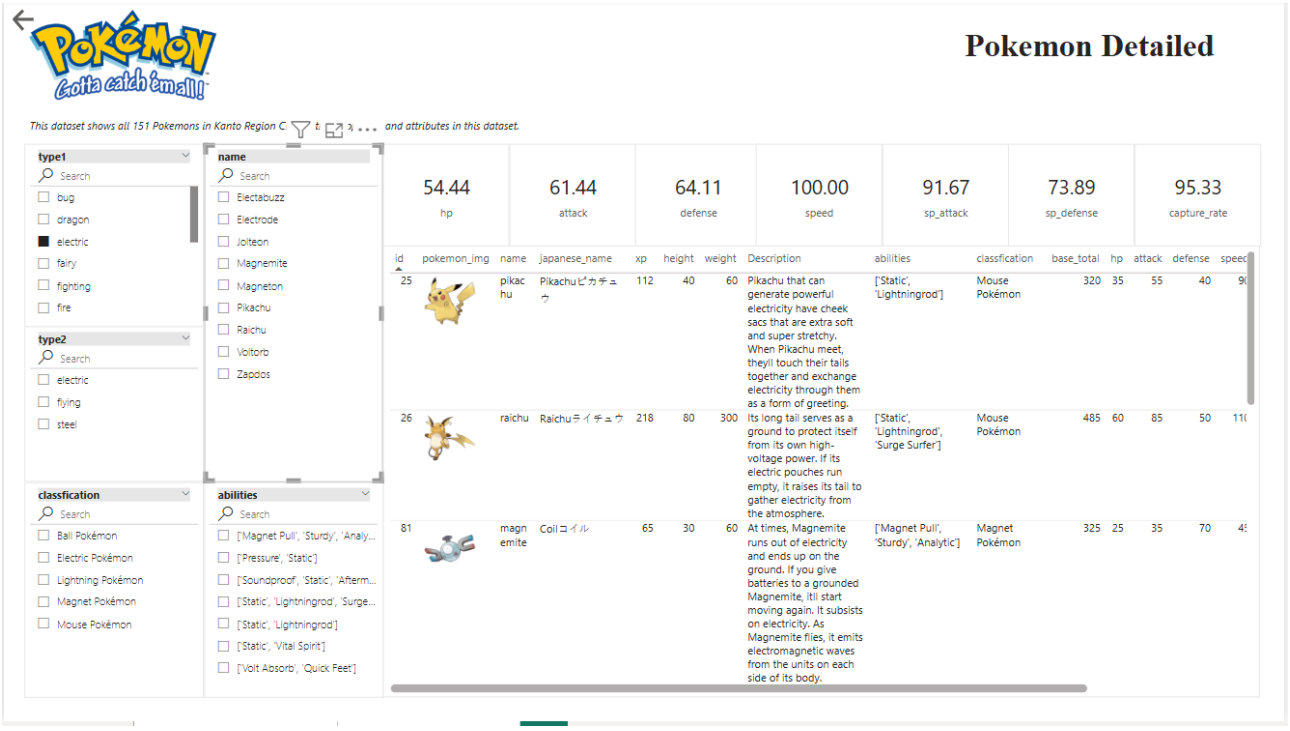
```
# 1. Convert JSON to CSV
json_to_csv(json_data, "output.csv")

# 2. Convert CSV back to JSON
csv_json = csv_to_json("output.csv")
print("CSV to JSON:\n", csv_json)

# 3. Convert JSON to XML
xml_data = json_to_xml(json_data)
print("JSON to XML:\n", xml_data)

# 4. Convert XML back to JSON
xml_json = xml_to_json(xml_data)
print("XML to JSON:\n", xml_json)
```

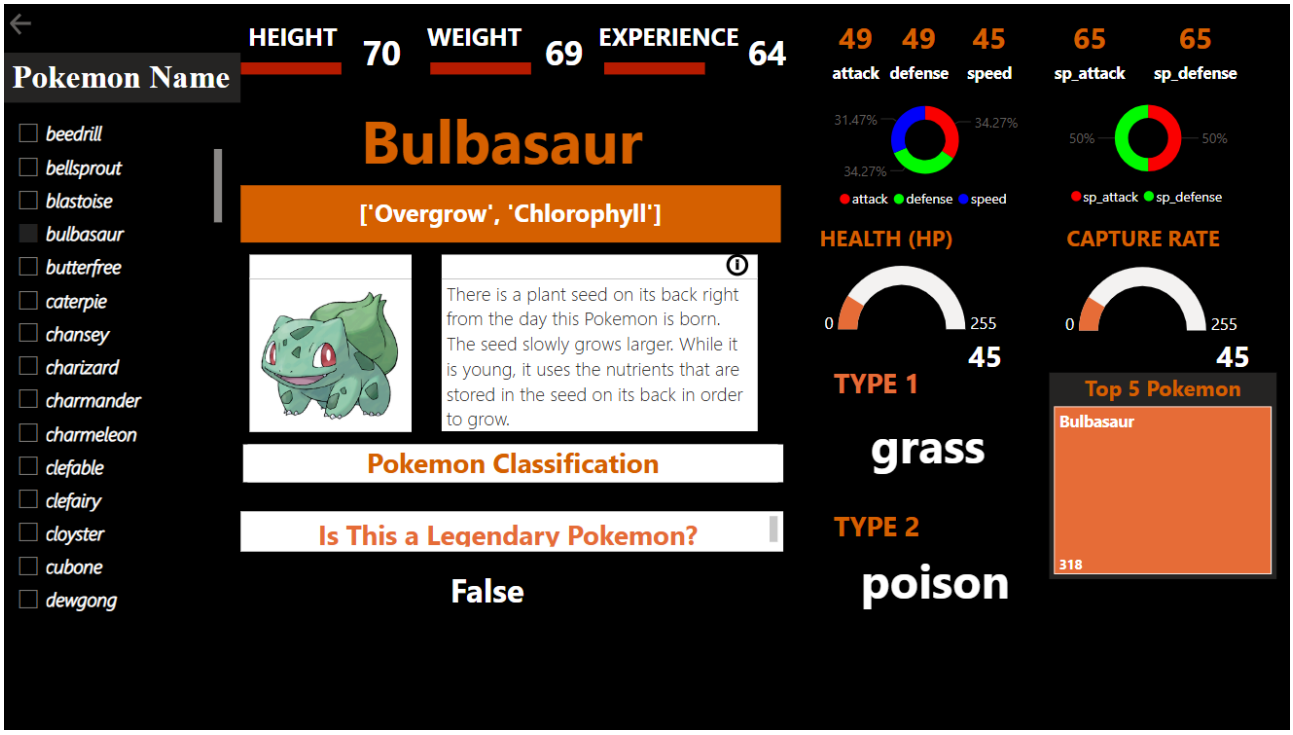
```
Data successfully written to output.csv
CSV to JSON:
 [
    {
        "name": "Alice",
        "age": "25",
        "city": "New York"
    },
    {
        "name": "Bob",
        "age": "30",
        "city": "Los Angeles"
    },
    {
        "name": "Charlie",
        "age": "35",
        "city": "Chicago"
    }
]
JSON to XML:
 <root><entry><name>Alice</name><age>25</age><city>New York</city></entry><entry><name>Bob</name><age>30</age><city>Los Angeles</city></
XML to JSON:
 [
    {
        "name": "Alice",
        "age": "25",
        "city": "New York"
    },
    {
        "name": "Bob",
        "age": "30",
        "city": "Los Angeles"
    },
    {
        "name": "Charlie",
        "age": "35",
        "city": "Chicago"
    }
]
```

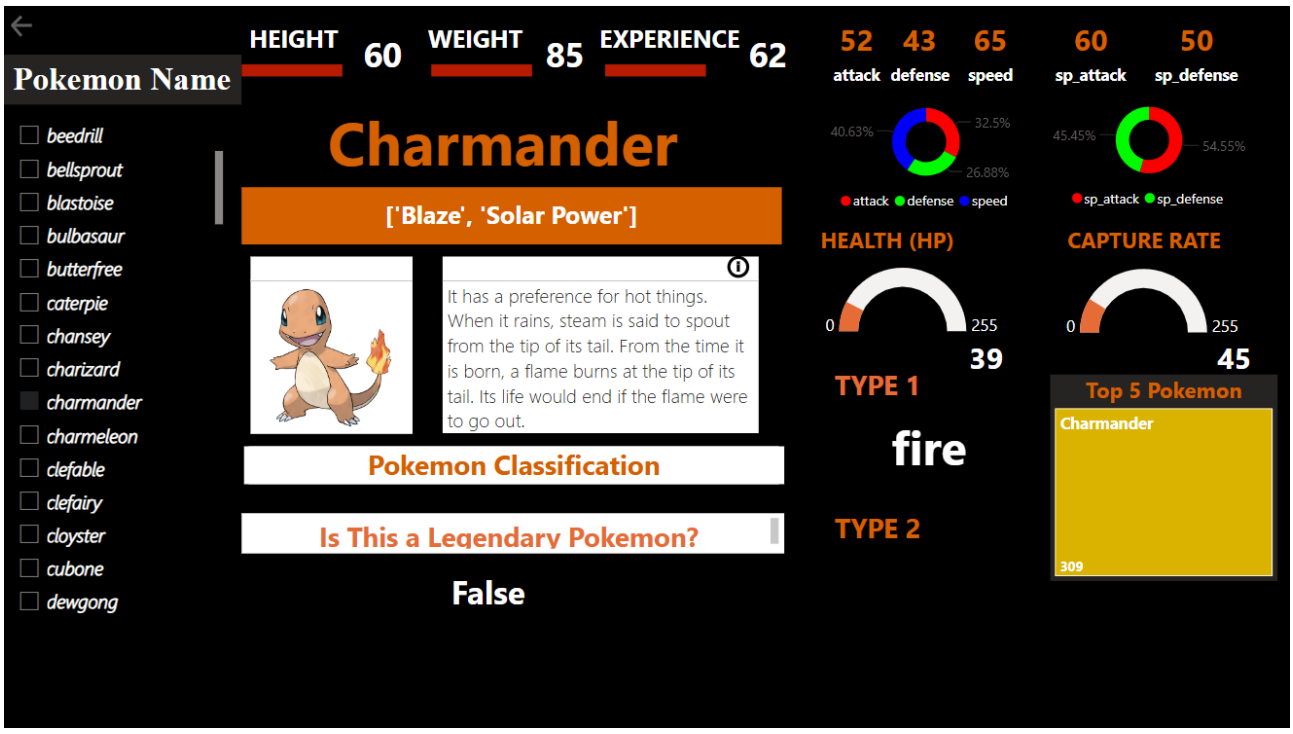
Practical No. 10

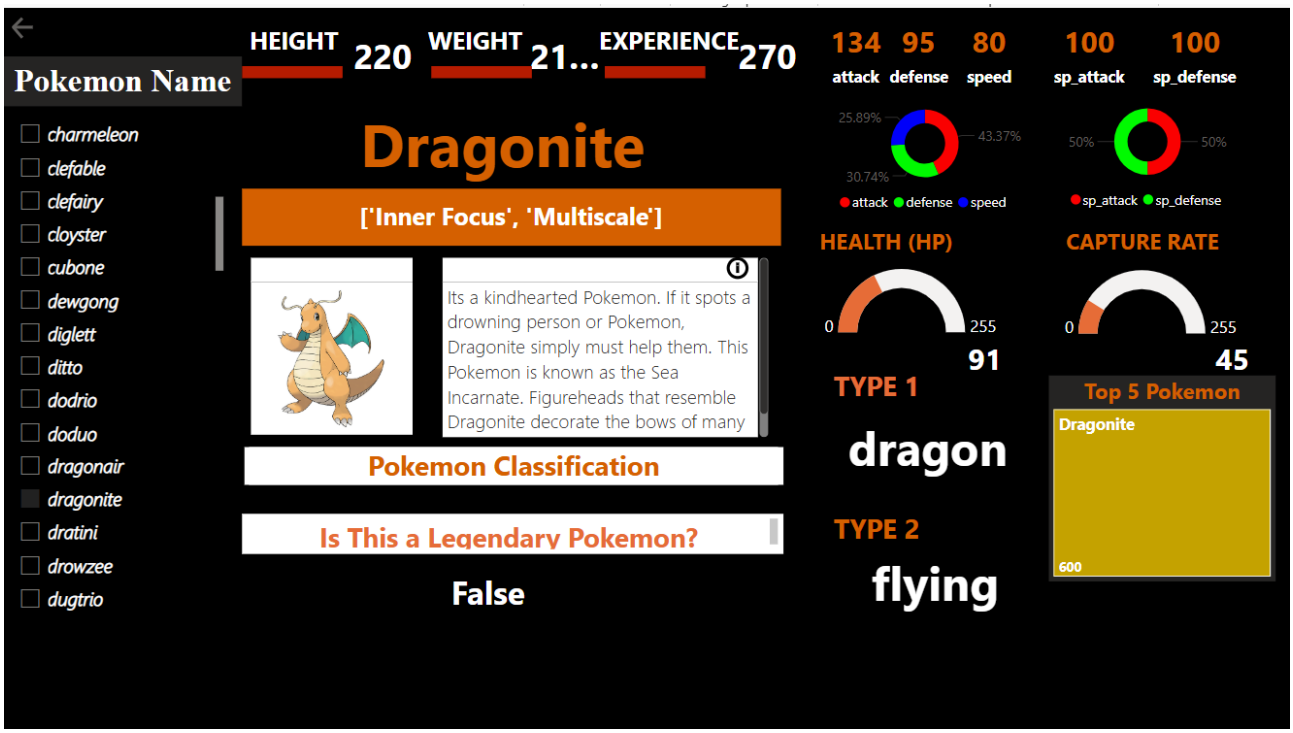
Create a Dashboard using Power BI

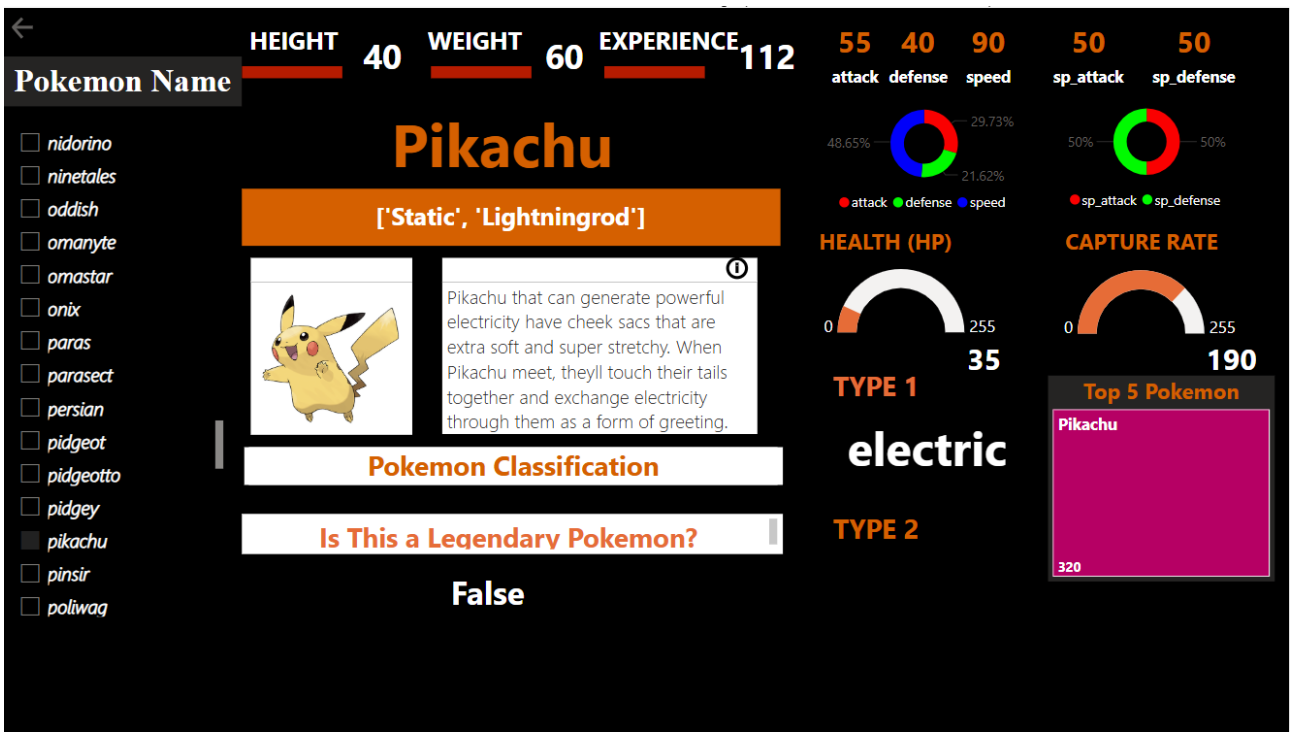


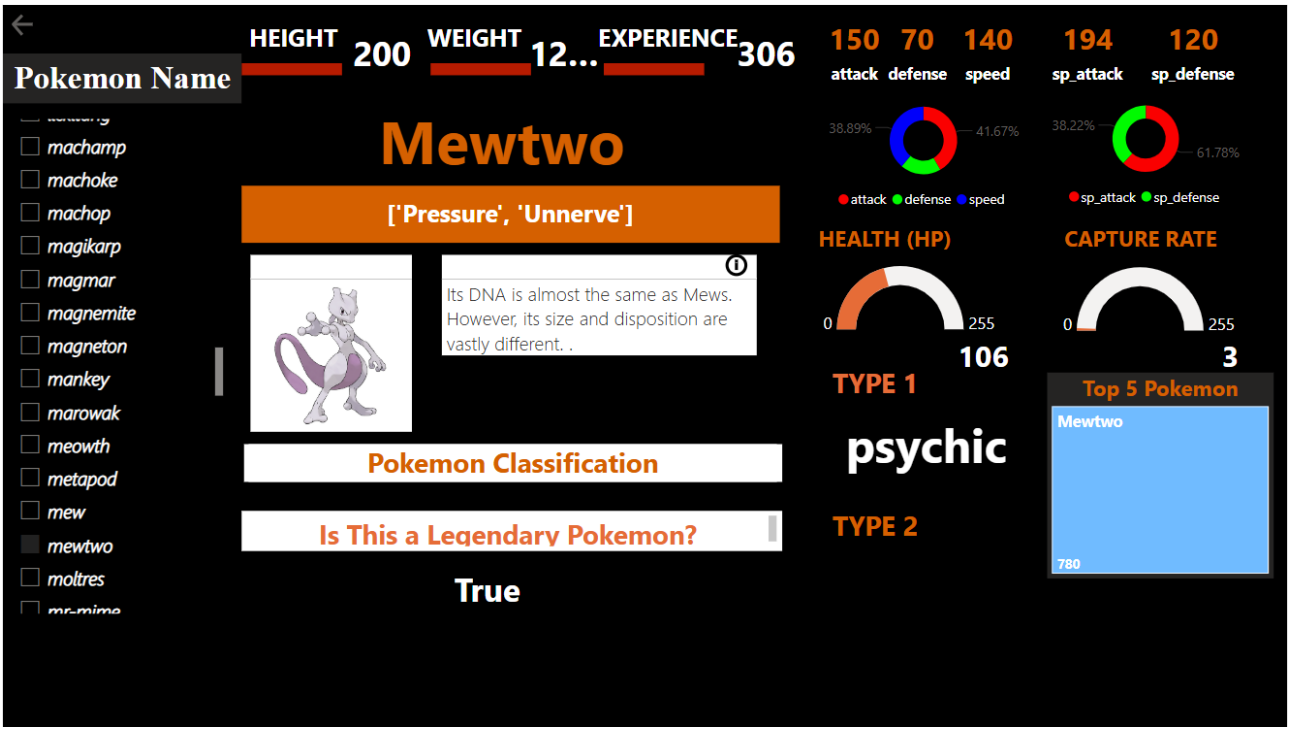



Pokemon Name
beedrill
bellsprout
blastoise
bulbasaur
butterfree
caterpie
chansey
charizard
charmander
charmeleon
clefable
clefairy
cloyster
cubone
dewgong
HEIGHT 70
WEIGHT 69
EXPERIENCE 64
Bulbasaur
['Overgrow', 'Chlorophyll']
There is a plant seed on its back right from the day this Pokemon is born. The seed slowly grows larger. While it is young, it uses the nutrients that are stored in the seed on its back in order to grow.
Pokemon Classification
Is This a Legendary Pokemon?
False
49 attack
49 defense
45 speed
31.47%
34.27%
34.27%
attack defense speed
65 sp_attack
65 sp_defense
50%
50%
sp_attack sp_defense
HEALTH (HP)
0
255
45
CAPTURE RATE
0
255
45
TYPE 1
grass
TYPE 2
poison
Top 5 Pokemon
Bulbasaur
318


Pokemon Name
beedrill
bellsprout
blastoise
bulbasaur
butterfree
caterpie
chansey
charizard
charmander
charmeleon
clefable
clefairy
cloyster
cubone
dewgong
HEIGHT 60
WEIGHT 85
EXPERIENCE 62
Charmander
['Blaze', 'Solar Power']
It has a preference for hot things. When it rains, steam is said to spout from the tip of its tail. From the time it is born, a flame burns at the tip of its tail. Its life would end if the flame were to go out.
Pokemon Classification
Is This a Legendary Pokemon?
False
52 attack
43 defense
65 speed
40.63%
32.5%
26.88%
attack defense speed
60 sp_attack
50 sp_defense
45.45%
54.55%
sp_attack sp_defense
HEALTH (HP)
0
255
39
CAPTURE RATE
0
255
45
TYPE 1
fire
TYPE 2
Top 5 Pokemon
Charmander
309


Pokemon Name
charmeleon
clefable
clefairy
cloyster
cubone
dewgong
diglett
ditto
dodrio
doduo
dragonair
dragonite
dratini
drowzee
dugtrio
HEIGHT 220
WEIGHT 21...
EXPERIENCE 270
Dragonite
['Inner Focus', 'Multiscale']
Its a kindhearted Pokemon. If it spots a drowning person or Pokemon, Dragonite simply must help them. This Pokemon is known as the Sea Incarnate. Figureheads that resemble Dragonite decorate the bows of many
Pokemon Classification
Is This a Legendary Pokemon?
False
134 95 80
attack defense speed
25.89%
43.37%
30.74%
attack defense speed
100 100
sp_attack sp_defense
50% 50%
sp_attack sp_defense
HEALTH (HP)
0 255
91
CAPTURE RATE
0 255
45
TYPE 1
dragon
TYPE 2
flying
Top 5 Pokemon
Dragonite
600


Pokemon Name
nidorino
ninetales
oddish
omanyte
omastar
onix
paras
parasect
persian
pidgeot
pidgeotto
pidgey
pikachu
pinsir
poliwag
HEIGHT 40
WEIGHT 60
EXPERIENCE 112
Pikachu
['Static', 'Lightningrod']
Pikachu that can generate powerful electricity have cheek sacs that are extra soft and super stretchy. When Pikachu meet, theyll touch their tails together and exchange electricity through them as a form of greeting.
Pokemon Classification
Is This a Legendary Pokemon?
False
55 40 90
attack defense speed
48.65%
29.73%
21.62%
attack defense speed
50 50
sp_attack sp_defense
50% 50%
sp_attack sp_defense
HEALTH (HP)
0 255
35
CAPTURE RATE
0 255
190
TYPE 1
electric
TYPE 2
Top 5 Pokemon
Pikachu
320


Pokemon Name
machamp
machoke
machop
magikarp
magmar
magnemite
magneton
mankey
marowak
meowth
metapod
mew
mewtwo
moltres
HEIGHT 200
WEIGHT 12...
EXPERIENCE 306
Mewtwo
['Pressure', 'Unnerve']
Its DNA is almost the same as Mews. However, its size and disposition are vastly different. .
Pokemon Classification
Is This a Legendary Pokemon?
True
150 attack
70 defense
140 speed
38.89%
41.67%
attack defense speed
194 sp_attack
120 sp_defense
38.22%
61.78%
sp_attack sp_defense
HEALTH (HP)
0
255
106
CAPTURE RATE
0
255
3
TYPE 1
psychic
TYPE 2
Top 5 Pokemon
Mewtwo
780

Practical No. 11

Create a Dashboard using Tableau