Frequency Distribution and Graphs

In statistical study, the researcher gathers data from different sources and finds a conclusion based on the data. The representation of data is an art, where data have good visualization on specific type of figure. The most useful method for data visualization is by statistical graphs and charts.
The first step in data representation is arranging the raw data the most convenient method is frequency distribution.

For example:
A researcher wants to study the ages of 50 billionaires in the world. The data provided is: 45, 46, 64, 57, 85, 92, 51, 71, 54, 48, 27, 66, 76, 55, 69, 54, 44, 54, 75, 46, 61, 68, 78, 61, 83, 88, 45, 89, 67, 56, 81, 58, 55, 62, 38, 55, 56, 64, 81, 38, 49, 68, 91, 56, 68, 46, 47, 83, 71, 62

The data does not make any sense unless we arrange it.

So, the researcher organizes data into frequency distribution.

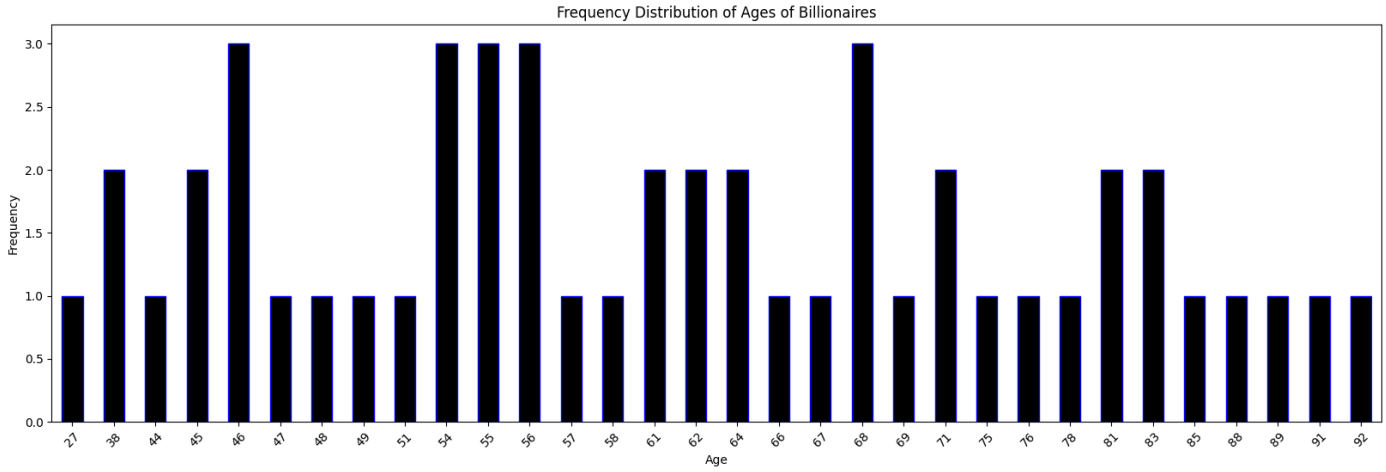
```
# import libraries for graphs and charts
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
ages = [45, 46, 64, 57, 85, 92, 51, 71, 54, 48, 27, 66, 76, 55, 69, 54, 44, 54, 75, 46, 61, 68, 78, 61, 83, 88, 45, 89, 67, 56, 81, 58,
df = pd.DataFrame(data = ages, columns=['Age'])
frequency_table = df['Age'].value_counts().sort_index()
print(frequency_table)

```

```
Age
27    1
38    2
44    1
45    2
46    3
47    1
48    1
49    1
51    1
54    3
55    3
56    3
57    1
58    1
61    2
62    2
64    2
66    1
67    1
68    3
69    1
71    2
75    1
76    1
78    1
81    2
83    2
85    1
88    1
89    1
91    1
92    1
Name: count, dtype: int64
```

```
# Plotting the frequency distribution
frequency_table.plot(kind='bar', figsize=(20, 6), color='black', edgecolor='blue')
plt.title('Frequency Distribution of Ages of Billionaires')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.xticks(rotation=45)
plt.show()
```



The above graph looks more readable, however, the number of data on x-axis is too many. So, it might be more effective if we arrange the data in class distribution.

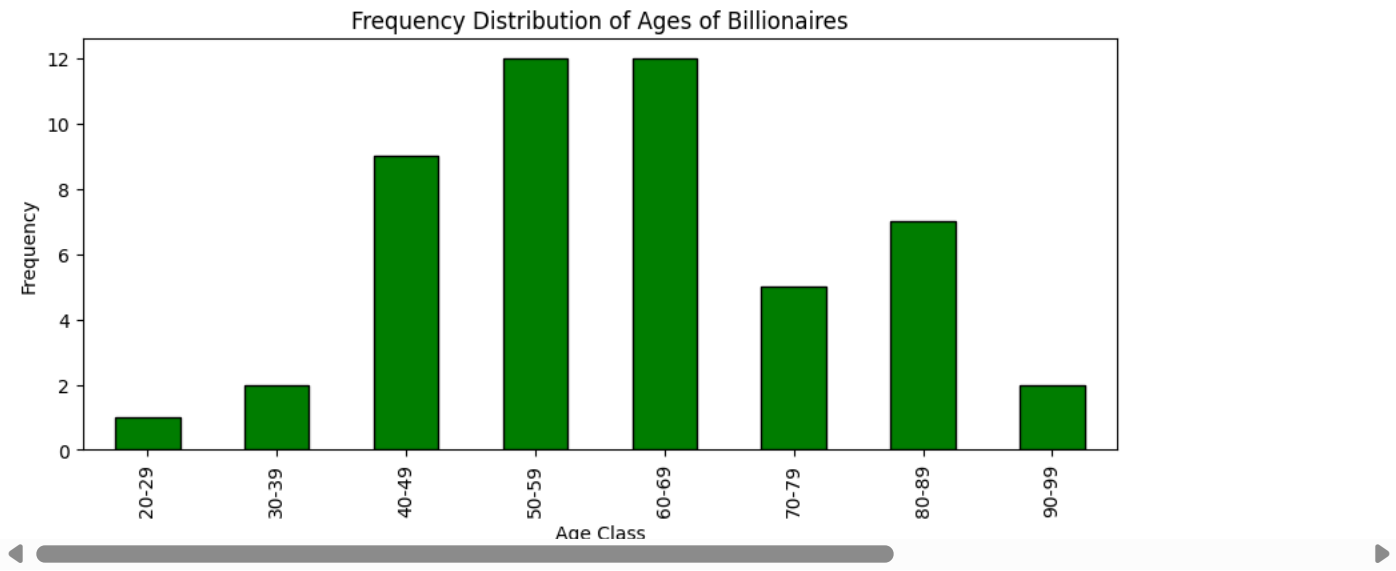
```
# Define age bins
bins = [20, 30, 40, 50, 60, 70, 80, 90, 100]
labels = ['20-29', '30-39', '40-49', '50-59', '60-69', '70-79', '80-89', '90-99']

# Cut the data into bins
df['Age Class'] = pd.cut(df['Age'], bins=bins, labels=labels, right=True)

# Create frequency distribution table
frequency_table = df['Age Class'].value_counts().sort_index()
print(frequency_table)
```

```
Age Class
20-29     1
30-39     2
40-49     9
50-59    12
60-69    12
70-79     5
80-89     7
90-99     2
Name: count, dtype: int64
```

```
# Plotting the frequency distribution
frequency_table.plot(kind='bar', figsize=(10, 4), color='green', edgecolor='black')
plt.title('Frequency Distribution of Ages of Billionaires')
plt.xlabel('Age Class')
plt.ylabel('Frequency')
plt.xticks(rotation=90)
plt.show()
```



Question 1

The data show NFL team payrolls (in millions of dollars) for a specific year. Construct a frequency distribution for the payroll.
99, 105, 106, 102, 102, 93, 109, 106, 77, 91, 103, 118, 97, 100, 107, 103, 94, 109, 100, 98, 84, 92, 98, 110, 94, 104, 98, 123, 102, 99, 100, 107

Construct a frequency distribution table using classes.

Histogram

In histogram, graph is constructed using vertical continuous bars for class data only. The height of the bars represents the frequency of the classes.

For example,

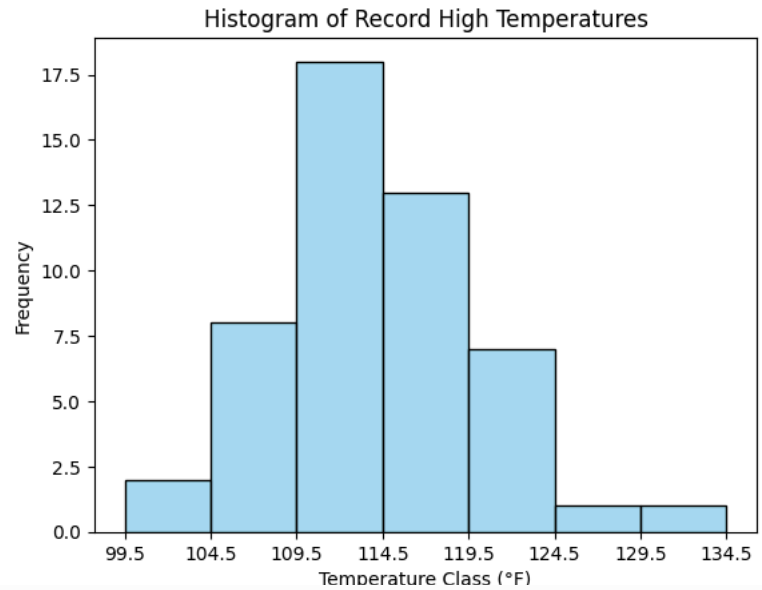
Construct a histogram to represent the data shown for the record high temperatures for each of the 50 states.

Class boundaries	Frequency
99.5 - 104.5	2
104.5 - 109.5	8
109.5 - 114.5	18
114.5 - 119.5	13
119.5 - 124.5	7
124.5 - 129.5	1
129.5 - 134.5	1

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

# Class boundaries and frequencies
class_boundaries = [99.5, 104.5, 109.5, 114.5, 119.5, 124.5, 129.5, 134.5]
frequency = [2, 8, 18, 13, 7, 1, 1]

# Create the histogram
sns.histplot(x=class_boundaries[:-1], weights=frequency, bins=class_boundaries, color='skyblue')
plt.title('Histogram of Record High Temperatures')
plt.xlabel('Temperature Class (°F)')
plt.ylabel('Frequency')
plt.xticks(class_boundaries)
plt.show()
```



Question 2

For 108 randomly selected college applicants, the following frequency distribution for entrance exam scores was obtained. Construct a histogram.

Class limits	Frequency
90 - 98	6
99 - 107	22
108 - 116	43
117 - 125	28
126 - 134	9

Frequency Polygon

In frequency polygon, a line represents the data which connects the point for the frequencies at the midpoint of the classes.

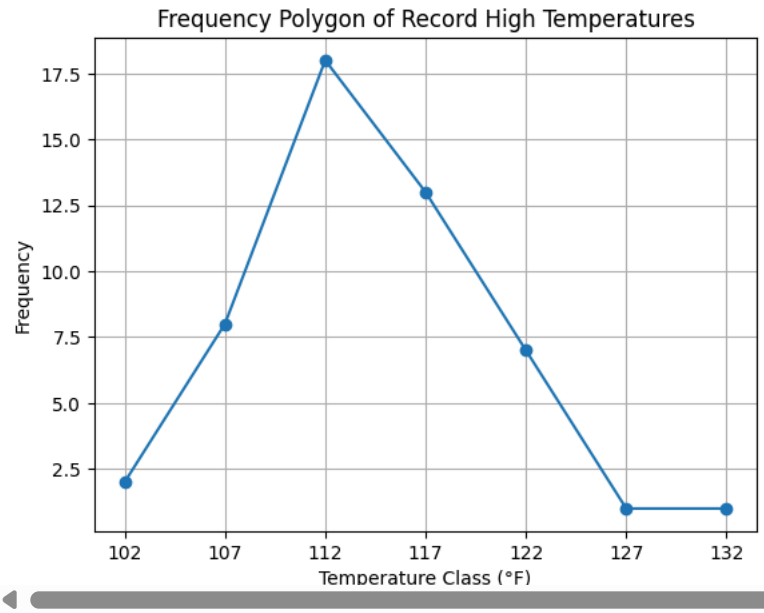
For example:
Constructing frequency polygon of the above data.

```
# Class boundaries and frequencies
class_boundaries = [99.5, 104.5, 109.5, 114.5, 119.5, 124.5, 129.5, 134.5]
frequency = [2, 8, 18, 13, 7, 1, 1]
midpoints = [] # empty list for midpoints

# calculating midpoints
for i in range(len(class_boundaries)-1):
    midpoint = (class_boundaries[i]+class_boundaries[i+1])/2
    midpoints.append(midpoint) # append added the values to the 'midpoints' list

# ploting frequency polygon
plt.plot(midpoints, frequency, marker = 'o', linestyle='-')
plt.title('Frequency Polygon of Record High Temperatures')
plt.xlabel('Temperature Class (°F)')
plt.ylabel('Frequency')
plt.xticks(midpoints) # x-axis data labels
plt.grid()
plt.show()

```



Question 3

The data show the number of railroad crossing accidents for the 50 states of the United States for a specific year. Construct a frequency polygon.

Class limits	Frequency
1 - 43	24
44 - 86	17
87 - 129	3
130 - 172	4
173 - 215	1
216 - 258	0
259 - 301	0
302 - 244	1

Ogive

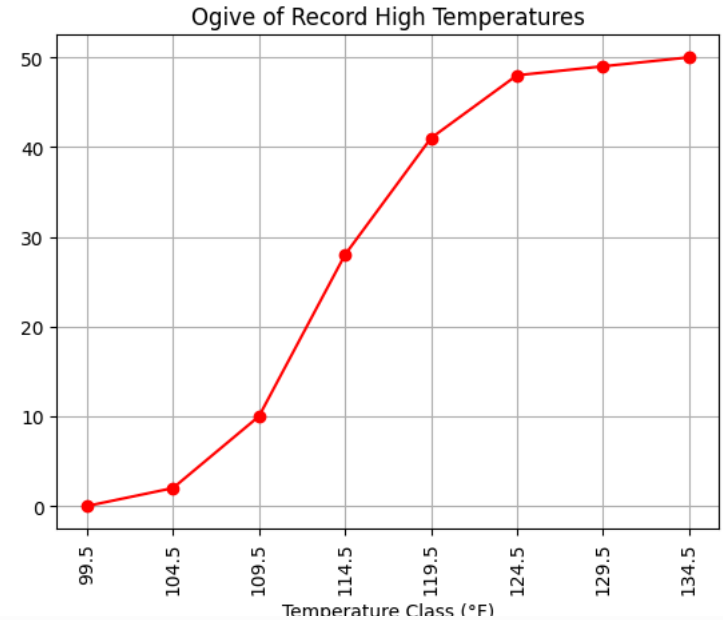
Ogive or cumulative frequency graph represents the frequencies for the classes in a frequency distribution

For example:

Constructing ogive of the above data.

```
# Class boundaries and frequencies
class_boundaries = [99.5, 104.5, 109.5, 114.5, 119.5, 124.5, 129.5, 134.5]
frequency = [2, 8, 18, 13, 7, 1, 1]
cumulative_frequency= np.cumsum(frequency) # calculating cumulative frequency
cumulative_frequency=[0] + list(cumulative_frequency) # adding initial 0 and
changing to list data type

# ploting ogive
plt.plot(class_boundaries,cumulative_frequency, marker = 'o', linestyle='-',
color = 'red')
plt.title('Ogive of Record High Temperatures')
plt.xlabel('Temperature Class (°F)')
plt.xticks(class_boundaries, rotation =  90) # x-axis data labels
plt.grid()
plt.show()
```



Question 4

The frequency distribution represents the cost (in cents) for the utilities of states that supply much of their own power. Construct an ogive for the data.

Class limits	Frequency
6 - 8	12
9 - 11	16
12 - 14	3
15 - 17	1
18 - 20	0
21 - 23	0
24 - 26	1

```
1 ##Write Your Code Here ##
```

Bar Graph

A bar graph represents the data by using vertical or horizontal bars whose heights or lengths represents the frequencies of the data.

Bar graph can be used to compare data of two or more groups.

For example:

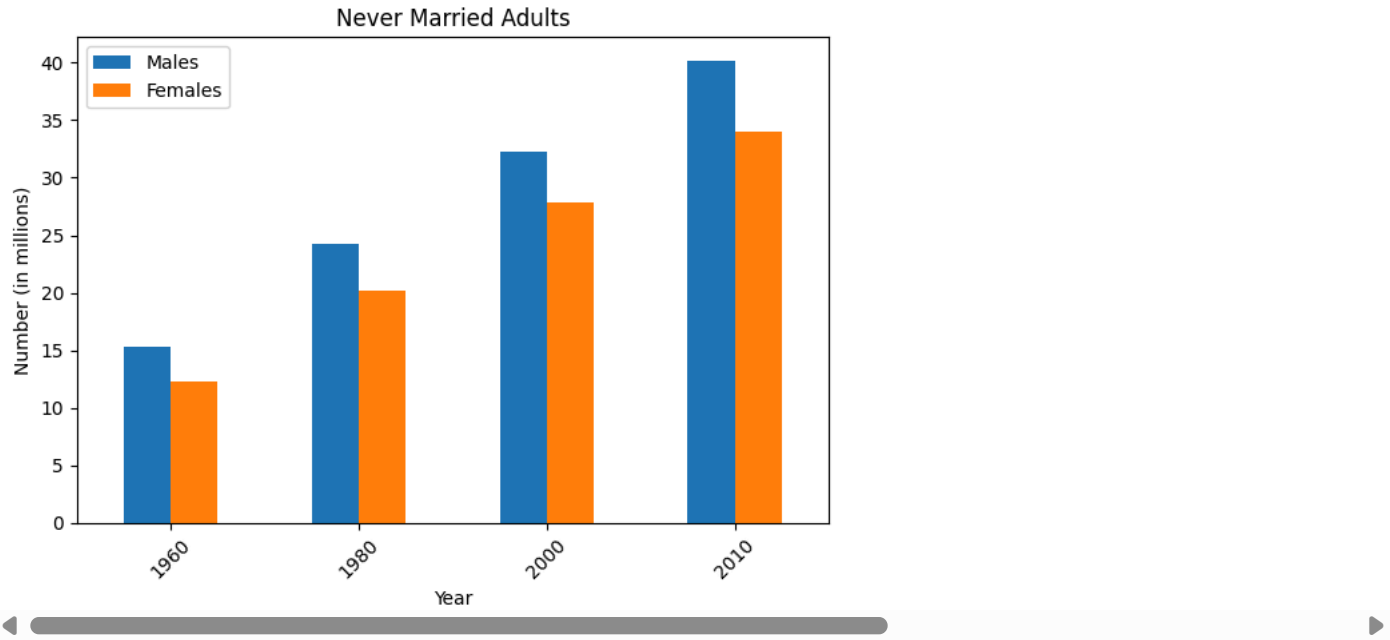
The following data for the number (in millions) of never married adults in the United States.

Year	Males	Females
1960	15.3	12.3
1980	24.2	20.2
2000	32.3	27.8
2010	40.2	34.0

```
1 # data in dictionary
2 bar_data = {'Year': [1960, 1980, 2000, 2010],
3             'Males': [15.3, 24.2, 32.3, 40.2],
4             'Females': [12.3, 20.2, 27.8, 34.0]
5 }
6
7 # converting it into table
8 bar_table = pd.DataFrame(bar_data)
9 bar_table.set_index('Year', inplace=True) # setting year as index
10 print(bar_table)
11
12 # Plotting bar graph
```

```
bar_table.plot(kind = 'bar')
plt.title('Never Married Adults')
plt.xlabel('Year')
plt.ylabel('Number (in millions)')
plt.xticks(rotation=45) # rotating x-label 45 degree
plt.tight_layout()
plt.show()
```

```
       Males  Females
Year
1960   15.3     12.3
1980   24.2     20.2
2000   32.3     27.8
2010   40.2     34.0
```



Question 5

The worldwide sales (in billions of dollars) for several fast-food franchises for a specific year are shown. Construct a bar graph for the data.

Fast Food	Sales
Wendy's	8.7
KFC	14.2
Pizza Hut	9.3
Burger King	12.7
Subway	10.0

```
##Write Your Code Here ##
```

Pareto Charts

A pareto chart is used to represent a frequency distribution for a categorical variable, and the frequencies are displayed by the heights of vertical bars, which are arranged in order from highest to lowest.

For example:

The data shown here consists of the number of police for specfic categories that a local municipality received in s specific year. Draw a Pareto chart for the data.

Category	Number
Juvenile complaint	92
Loud noise/music/party	27
Drug offenses	79
Driving under the influence	38
Disabled vehicle	65

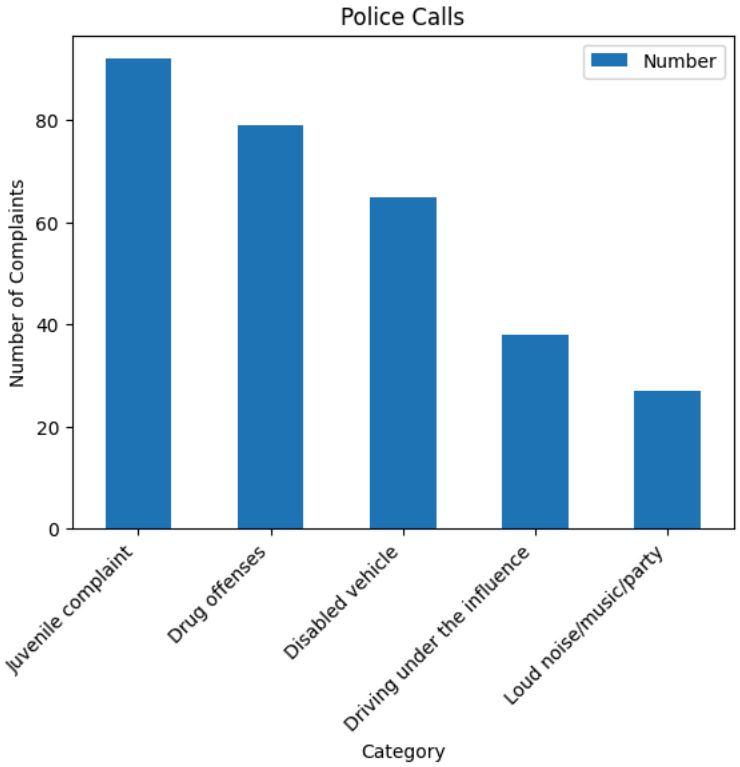
```
# Given data
pareto_data = {'Category': ['Juvenile complaint', 'Loud noise/music/party', 'Drug offenses', 'Driving under the influence', 'Disabled ve
```

```
                'Number': [92, 27, 79, 38, 65]
}

# converting into table
pareto_table = pd.DataFrame(pareto_data)
pareto_table.set_index('Category', inplace=True)
pareto_table.sort_values('Number', ascending=False, inplace=True) # descending order

# Bar chart for the Number of complaints
pareto_table.plot(kind='bar')
plt.title('Police Calls')
plt.xlabel('Category')
plt.ylabel('Number of Complaints')
plt.xticks(rotation=45, ha='right')
plt.show()

```



Question 6

Construct a Pareto chart for the following data on excercise.

Excercises	Calories burned per minute
Walking, 2 mph	2.8
Bicycling, 5.5 mph	3.2
Golfing	5.0
Tennis playing	7.1
Skiing, 3 mph	9.0
Running, 7 mph	14.5

```
##Write Your Code Here ##
```

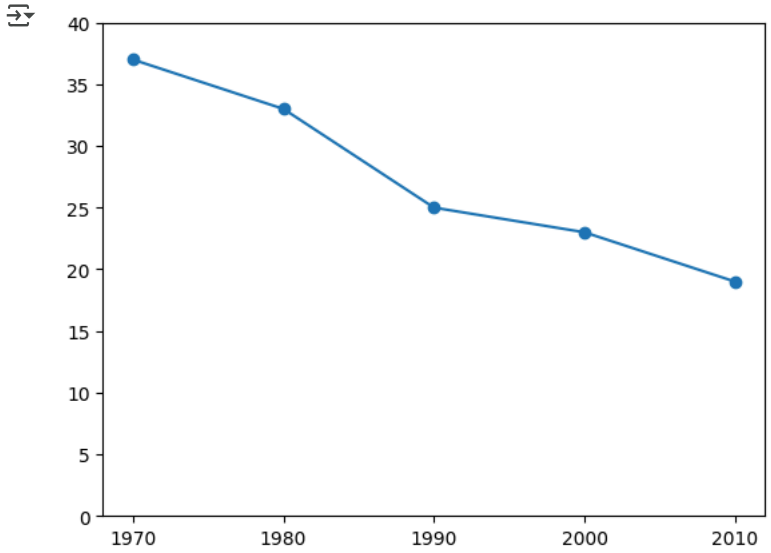
Time Series Graph

A time series graph represents data that occur over a specific period of time.

For example

Year	1970	1980	1990	2000	2010
Percent	37	33	25	23	19

```
# data
time_data = {'year': [1970, 1980, 1990, 2000, 2010],
             'percent': [37, 33, 25, 23, 19]
}
time_table = pd.DataFrame(time_data)
plt.plot(time_table['year'], time_table['percent'], marker = 'o', linestyle =
'-')
plt.ylim(0,40)
plt.xticks(time_table['year'])
plt.show()
```



Question 7

The amount spent (in billions of dollars) for ads online is shown. Draw a time series graph and comment on the trend.

Year	2010	2011	2012	2013	2014	2015
Amount	68.4	80.2	94.2	106.1	119.8	132.1

```
##Write Your Code Here ##
```

Pie Graph

A pie graph is a circle that is divided into sections or wedges according to the percentage of frequencies in each category of the distribution.

For example:

Snack	Pounds in millions (frequency)
Potato chips	11.2
Tortilla chips	8.2
Pretzels	4.3
Popcorn	3.8
Snack nuts	2.5

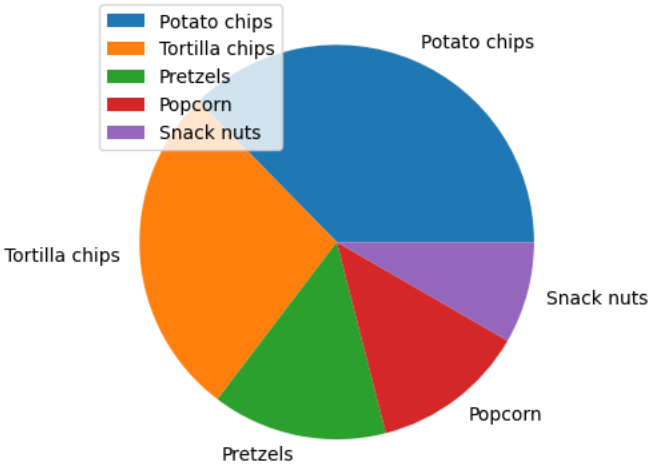
```
snack = ['Potato chips', 'Tortilla chips', 'Pretzels', 'Popcorn', 'Snack nuts']
pounds = [11.2, 8.2, 4.3, 3.8, 2.5]

plt.pie(pounds, labels=snack)
plt.legend(snack, loc = 'upper left')
plt.show
```

```
matplotlib.pyplot.show
def show(*args, **kwargs) -> None
```

```
**Auto-show in jupyter notebooks**

The jupyter backends (activated via ``%matplotlib inline``,
``%matplotlib notebook``, or ``%matplotlib widget``), call ``show()`` at
the end of every cell by default. Thus, you usually don't have to call it
explicitly there.
```



Question 8

The data show the percentages of the types of energy consumed in the United States. Draw a pie graph for the data.

Energy	Percent
Natural gas	25
Coal	21
Petroleum	37
Nuclear	9
Renewable	8

```
1 ##Write Your Code Here ##
```

Dotplots

A dotplot is a statistical graph in which each data value is plotted as a point (dot) above the horizontal axis.

For example:

19, 9, 16, 15, 10, 28, 15, 16, 12, 15, 15, 12, 14, 8, 13, 19, 7, 8, 7, 8, 14, 11, 12, 7, 6, 11, 13, 4, 6, 12, 11, 9, 12, 6, 10, 9, 11, 8, 7, 13

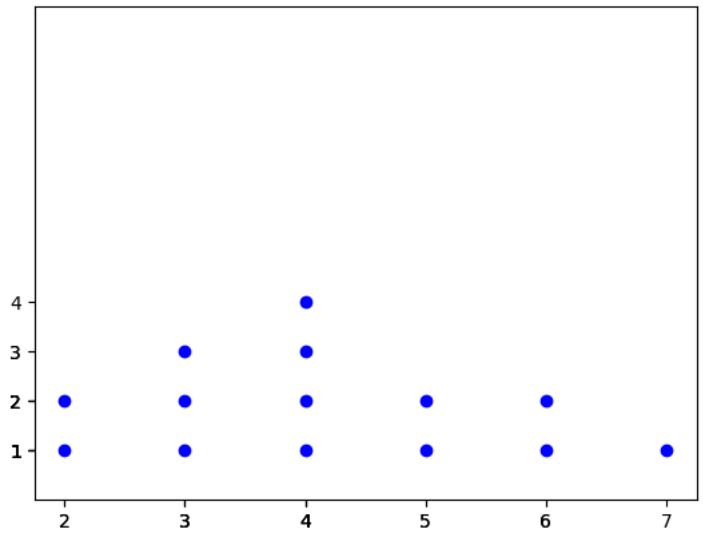
```
1 dot_data = [19, 9, 16, 15, 10, 28, 15, 16, 12, 15, 15, 12, 14, 8, 13, 19, 7, 8, 7, 8, 14, 11, 12, 7, 6, 11, 13, 4, 6, 12, 11, 9, 12, 6,
2
3 unique_values, counts = np.unique(dot_data, return_counts=True)
4
5 print(unique_values)
6 print(counts)
7
```

```
[ 4  6  7  8  9 10 11 12 13 14 15 16 19 28]
[1 3 4 4 3 2 4 5 3 2 4 2 2 1]
```

```
1 x_list = []
2 y_list = []
3
4 for i, x in enumerate(unique_values)
5     count = 1
6     while count != counts[i] + 1:
7         x_list.append(x)
8         y_list.append(count)
```

```
        count += 1

plt.plot(x_list, y_list, 'bo')
plt.yticks(y_list)
plt.ylim(0,10)
plt.xticks(x_list)
plt.show()
```




```

data = [2, 2, 3, 3, 3, 4, 4, 4, 4, 5, 5, 6, 6, 7]
unique_values, counts = np.unique(data, return_counts=True)

plt.figure(figsize=(8, 6))

for value, count in zip(unique_values, counts):
    plt.scatter([value] * count, range(count), marker='o')

plt.xlabel('Value')
plt.ylabel('Frequency')
plt.ylim(-1,10)
plt.yticks([])
plt.tight_layout()
plt.title('Dot Plot with Frequency')
plt.show()
```