Course: Programming for AI (AI2001)   Semester: Fall 2023
Instructor: Sameer Faisal   T.A: N/A

Note:
- Maintain discipline during the lab.
- Listen and follow the instructions as they are given.
- Just raise hand if you have any problem.
- Completing all tasks of each lab is compulsory.
- Get your lab checked at the end of the session.

Data Visualization

Data visualization is all about understanding data by placing it in a visual context (e.g. graphs, charts, etc.) so that patterns, trends and correlations can be exposed that might not otherwise be detected.

Data visualization is visual representation of data for exploration, discovery, and insight of data. Interactive component provides more insight as compared to a static image.
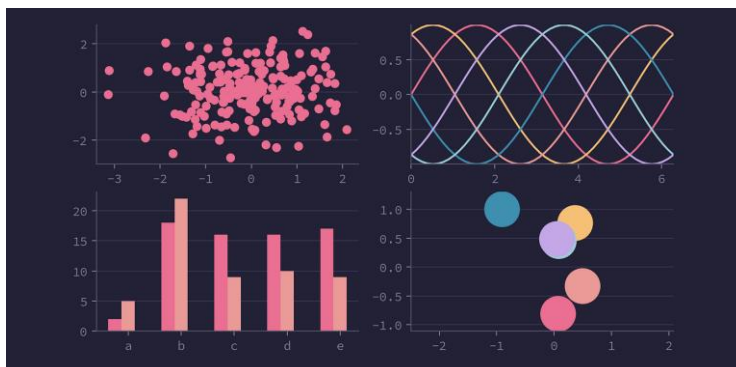
For example:
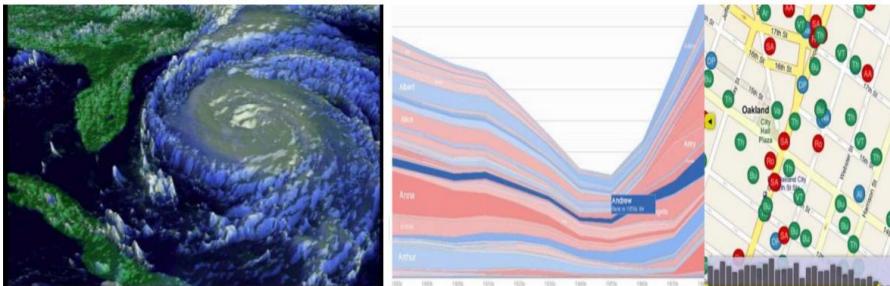
Matplotlib Library for Data Visualization

Matplotlib is a Python library used to create charts and graphs.

For example:

Using Matplotlib Library for Data Visualization

To use matplotlib in python, firstly we need to import that into our environment like this:
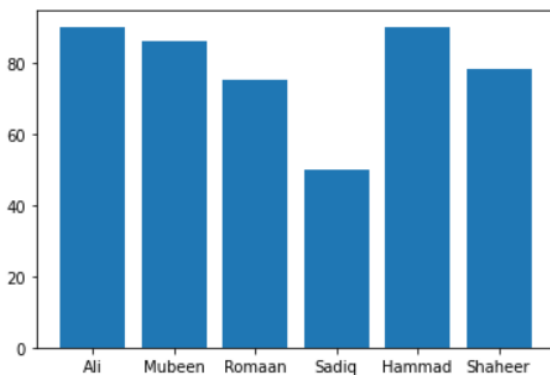
from matplotlib import pyplot as plt

OR

import matplotlib.pyplot as plt

This will help us in creating many types of charts and graphs that we will study in this lab.

➢ Bargraphs

• Creating a bargraph:

```python
import matplotlib.pyplot as plt
names=["Ali","Mubeen","Romaan","Sadiq","Hammad","Shaheer"] #storing values that will be shown on x-axis
percentages=[90,86,75,50,90,78] ##storing values that will be shown on y-axis
plt.bar(names,percentages) #x-axis, y-axis
plt.show()
```
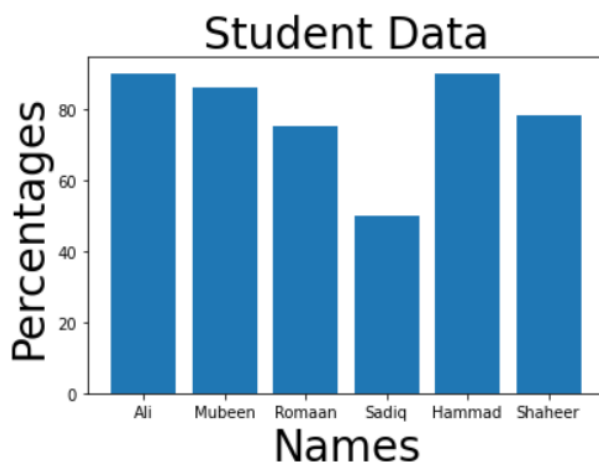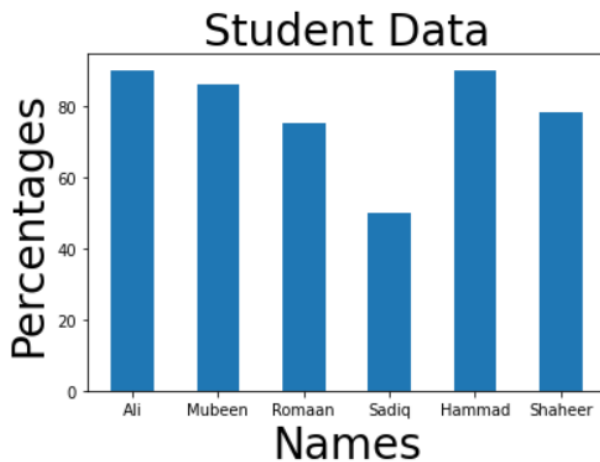

• Adding titles and axis labels to the bargraph

```python
import matplotlib.pyplot as plt
names=["Ali","Mubeen","Romaan","Sadiq","Hammad","Shaheer"] #storing values that will be shown on x-axis
percentages=[90,86,75,50,90,78] ##storing values that will be shown on y-axis
plt.xlabel("Names",fontsize=28) #fontsize is optional
plt.ylabel("Percentages",fontsize=28) #fontsize is optional
plt.title("Student Data", fontsize=28) #fontsize is optional
plt.bar(names,percentages) #x-axis, y-axis
plt.show()
```

- **Adjusting the width**

```python
import matplotlib.pyplot as plt
names=["Ali","Mubeen","Romaan","Sadiq","Hammad","Shaheer"] #storing values that will be shown on x-axis
percentages=[90,86,75,50,90,78] ##storing values that will be shown on y-axis
plt.xlabel("Names",fontsize=28) #fontsize is optional
plt.ylabel("Percentages",fontsize=28) #fontsize is optional
plt.title("Student Data", fontsize=28) #fontsize is optional
plt.bar(names,percentages,width=0.5) #x-axis, y-axis
plt.show()
```


Values for width adjustment are normally given in a range of 0 – 1. However, a value greater than 1 is also acceptable, but, it will make all the bars overlap each other.

- **Color Adjustment**

  You can use any of the 140 supported color names
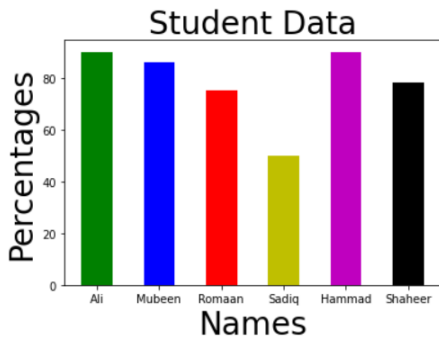 (https://www.w3schools.com/colors/colors_names.asp).

- ❖ **Single color:**

```python
import matplotlib.pyplot as plt
names=["Ali","Mubeen","Romaan","Sadiq","Hammad","Shaheer"] #storing values that will be shown on x-axis
percentages=[90,86,75,50,90,78] ##storing values that will be shown on y-axis
plt.xlabel("Names",fontsize=28) #fontsize is optional
plt.ylabel("Percentages",fontsize=28) #fontsize is optional
plt.title("Student Data", fontsize=28) #fontsize is optional
plt.bar(names,percentages,width=0.5, color='violet') #x-axis, y-axis
plt.show()
```
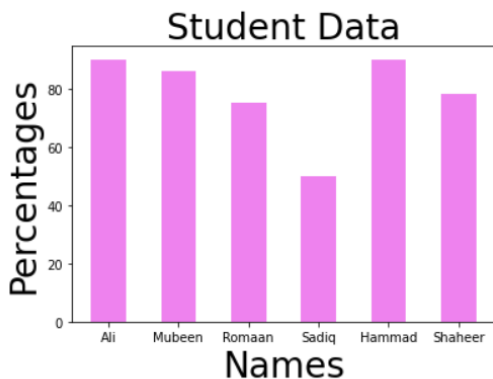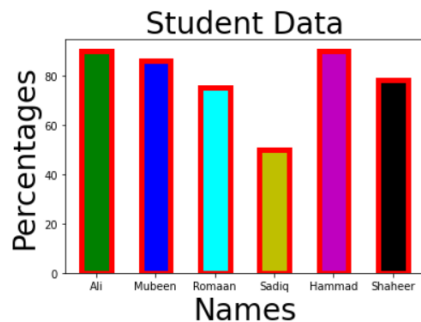
❖ Multiple colors:

```python
import matplotlib.pyplot as plt
names=["Ali","Mubeen","Romaan","Sadiq","Hammad","Shaheer"] #storing values that will be shown on x-axis
percentages=[90,86,75,50,90,78] ##storing values that will be shown on y-axis
plt.xlabel("Names",fontsize=28) #fontsize is optional
plt.ylabel("Percentages",fontsize=28) #fontsize is optional
plt.title("Student Data", fontsize=28) #fontsize is optional
colors=['g','b','r','y','m','black']
plt.bar(names,percentages,width=0.5, color=colors) #x-axis, y-axis
plt.show()
```


❖ Edge Coloring (Making Outlines):

```python
import matplotlib.pyplot as plt
names=["Ali","Mubeen","Romaan","Sadiq","Hammad","Shaheer"] #storing values that will be shown on x-axis
percentages=[90,86,75,50,90,78] ##storing values that will be shown on y-axis
plt.xlabel("Names",fontsize=28) #fontsize is optional
plt.ylabel("Percentages",fontsize=28) #fontsize is optional
plt.title("Student Data", fontsize=28) #fontsize is optional
colors=['g','b','aqua','y','m','black']
plt.bar(names,percentages,width=0.5, color=colors,edgecolor='red',linewidth=5) #x-axis, y-axis
plt.show()
```


❖ Adding Legends:

```python
import matplotlib.pyplot as plt
names=["Ali","Mubeen","Romaan","Sadiq","Hammad","Shaheer"] #storing values that will be shown on x-axis
percentages=[90,86,75,50,90,78] ##storing values that will be shown on y-axis
plt.xlabel("Names",fontsize=28) #fontsize is optional
plt.ylabel("Percentages",fontsize=28) #fontsize is optional
plt.title("Student Data", fontsize=28) #fontsize is optional
colors=['g','b','aqua','y','m','black']
plt.bar(names,percentages,width=0.5, color=colors,edgecolor='red',linewidth=1,label="Marks Level") #x-axis, y-axis
plt.legend()
plt.show()
```
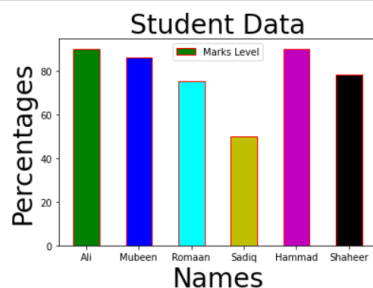
• Multiple Bargraphs

```
#Multiple bar graphs

x = ["python", "c", "c++", "java"]
y = [85,70,60,82]
z = [20, 30, 40, 50]
plt.xlabel("languages", fontsize = 10)
plt.ylabel("No")
plt.title("fast nuces")
c = ["y","b","m","g"]
plt.bar(x,y, width = 0.4, color="r", edgecolor="r",linewidth=1, label = "popularity")
plt.bar(x,z, width = 0.4, color="b", edgecolor="r",linewidth=1, label = "popularity1"
plt.legend()
plt.show()
```
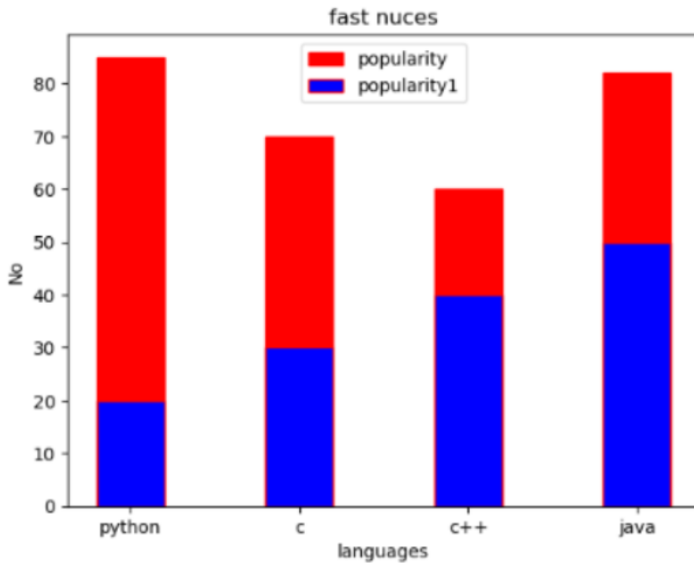

❖ Side – by – Side Bargraphs:

```
import matplotlib.pyplot as plt
import numpy as np
# Sample data
categories = ['Category A', 'Category B', 'Category C', 'Category D']
data_set1 = [25, 40, 30, 35]
data_set2 = [30, 35, 20, 45]
# Set the width of each bar
bar_width = 0.35
# Generate an array of equally spaced values for the x-axis
x = np.arange(len(categories))
# Create the side-by-side bars
plt.bar(x - bar_width/2, data_set1, bar_width, label='Data Set 1', color='blue')
plt.bar(x + bar_width/2, data_set2, bar_width, label='Data Set 2', color='green')
# Add labels and title
plt.xlabel('Categories')
plt.ylabel('Values')
plt.title('Side-by-Side Bar Graph')
plt.xticks(x, categories)
plt.legend()
# Display the graph
plt.show()
```
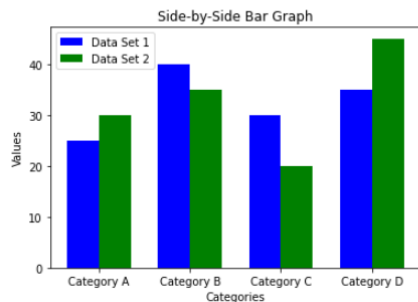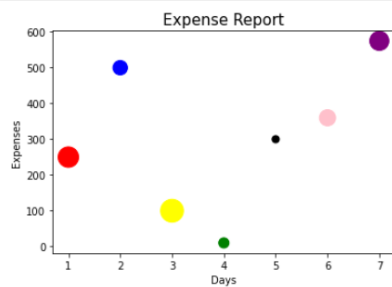
➢ Scatterplots
• Creating a scatterplot:

```python
import matplotlib.pyplot as plt
days_of_week=[1,2,3,4,5,6,7]
expenses=[250,500,100,10,300,360,575]
plt.scatter(days_of_week,expenses)
plt.xlabel('Days')
plt.ylabel('Expenses')
plt.title("Expense Report",fontsize=15)
plt.show()
```
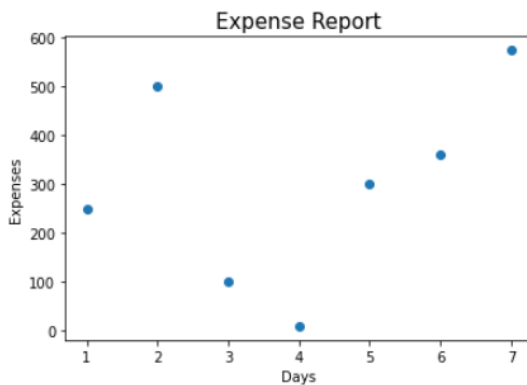

• Colors & sizes:

```python
import matplotlib.pyplot as plt
days_of_week=[1,2,3,4,5,6,7]
expenses=[250,500,100,10,300,360,575]
sizes=[400,200,500,100,50,250,350] #storing sizes for each value to be plotted
colors=['red','blue','yellow','green','black','pink','purple'] #storing colors for each value to be plotted
plt.scatter(days_of_week,expenses, s=sizes,c=colors) #'s' is the deafult attribute for size and 'c' for colors
plt.xlabel('Days')
plt.ylabel('Expenses')
plt.title("Expense Report",fontsize=15)
plt.show()
```

- **Changing Markers:**

```python
import matplotlib.pyplot as plt
days_of_week=[1,2,3,4,5,6,7]
expenses=[250,500,100,10,300,360,575]
sizes=[400,200,500,100,50,250,350] #storing sizes for each value to be plotted
colors=['red','blue','yellow','green','black','pink','purple'] #storing colors for each value to be plotted
plt.scatter(days_of_week,expenses,s=sizes,c=colors,marker="*", edgecolor='g') #marker attribute changes display of scatterplot
plt.xlabel('Days')
plt.ylabel('Expenses')
plt.title("Expense Report",fontsize=15)
plt.show()
```
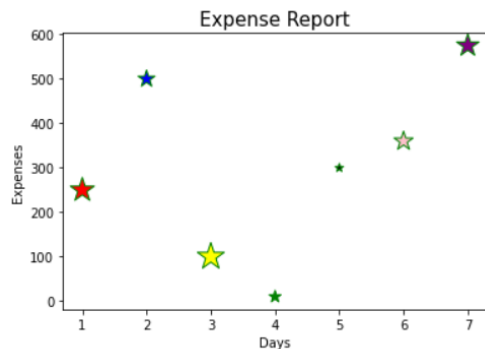

➢ Histograms

A histogram is a graph showing frequency distributions. It is a graph showing the number of observations within each given interval.

Example: Say you ask for the height of 250 people; you might end up with a histogram like this:
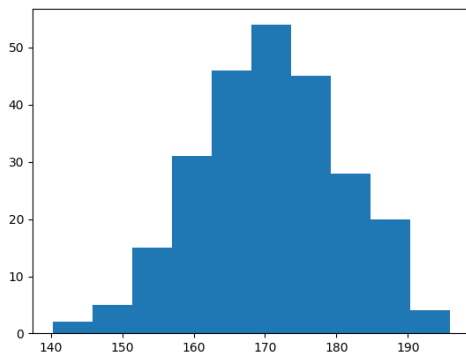
You can read from the histogram that there are approximately:

2 people from 140 to 145cm

5 people from 145 to 150cm

15 people from 151 to 156cm

31 people from 157 to 162cm

46 people from 163 to 168cm

53 people from 168 to 173cm

45 people from 173 to 178cm

28 people from 179 to 184cm

21 people from 185 to 190cm

4 people from 190 to 195cm

- # Creating a histogram:

```python
import matplotlib.pyplot as plt

values = [5, 5, 5, 4, 1, 3, 3, 2, 1, 4, 2, 2, 5, 5, 5, 5, 5, 5, 5, 5, 1, 4, 1, 3, 4, 1, 3, 4, 2, 5, 1, 4, 1, 2, 4, 4, 4, 4,
          5, 4, 5, 5, 3, 2, 3, 5, 2, 3, 1, 1, 5, 2, 1, 5, 3, 3, 3, 1, 3, 2, 3, 4, 4, 3, 5, 4, 4, 5, 3, 3, 4, 5, 1, 1, 3,
          4, 3, 5, 3, 5, 5, 2, 2, 1, 2, 4, 5, 3, 1, 5, 5, 5, 2, 5, 4, 5, 2, 3, 4, 4, 3]
# Print the array
print(values)
plt.hist(values,width=0.5)
plt.title('Occurence of numbers')
plt.xlabel('Value')
plt.ylabel('Frequency')
# Display the histogram
plt.show()
```

[5, 5, 5, 4, 1, 3, 3, 2, 1, 4, 2, 2, 5, 5, 5, 5, 5, 5, 5, 5, 1, 4, 1, 3, 4, 1, 3, 4, 2, 5, 1, 4, 1, 2, 4, 4, 4, 4, 5, 4, 5, 5, 3, 2, 3, 5, 2, 3, 1, 1, 5, 2, 1, 5, 3, 3, 3, 1, 3, 2, 3, 4, 4, 3, 5, 4, 4, 5, 3, 3, 4, 5, 1, 1, 3, 4, 3, 5, 3, 5, 5, 2, 2, 1, 2, 4, 5, 3, 1, 5, 5, 5, 2, 5, 4, 5, 2, 3, 4, 4, 3]
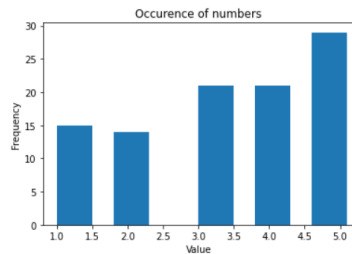
# Another Example:

```python
import matplotlib.pyplot as plt

std_marks = [50, 48, 41, 100, 38, 19, 46, 54, 61, 61, 20, 22, 52, 45, 13, 8, 19, 55, 66, 21, 27, 94, 36, 38, 88, 23, 78, 33,
             92, 51, 95, 66, 97, 74, 92, 76, 91, 17, 57, 68, 19, 31, 13, 79, 72, 35, 62, 79, 81, 29, 23, 11, 52, 98, 95, 45,
             98, 13, 84, 54, 13, 75, 63, 10, 96, 93, 53, 86, 22, 58, 59, 68, 91, 62, 84, 80, 45, 91, 19, 91, 15, 79, 55, 33,
             29, 24, 79, 87, 55, 17, 33, 82, 90, 43, 50, 13, 45, 32, 74, 33]
# Print the array
print(values)
plt.hist(values,width=5)
plt.title('Final Exam Marks')
plt.xlabel('Marks')
plt.ylabel('Number of students')
# Display the histogram
plt.show()
```

[50, 48, 41, 100, 38, 19, 46, 54, 61, 61, 20, 22, 52, 45, 13, 8, 19, 55, 66, 21, 27, 94, 36, 38, 88, 23, 78, 33, 92, 51, 95, 66, 97, 74, 92, 76, 91, 17, 57, 68, 19, 31, 13, 79, 72, 35, 62, 79, 81, 29, 23, 11, 52, 98, 95, 45, 98, 13, 84, 54, 13, 75, 63, 10, 96, 93, 53, 86, 22, 58, 59, 68, 91, 62, 84, 80, 45, 91, 19, 91, 15, 79, 55, 33, 29, 24, 79, 87, 55, 17, 33, 82, 90, 43, 50, 13, 45, 32, 74, 33]
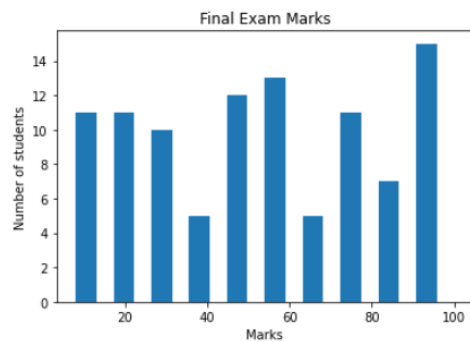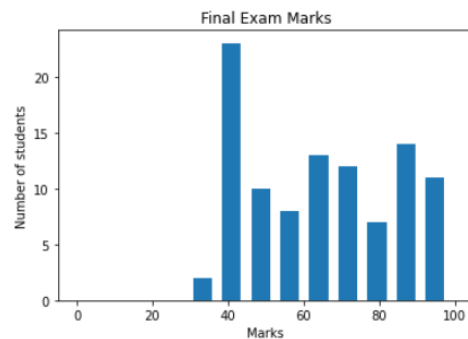
```python
import matplotlib.pyplot as plt

std_marks = [63, 41, 84, 97, 99, 46, 56, 91, 75, 43, 40, 92, 90, 75, 99, 43, 97, 51, 51, 44, 46, 62, 68, 72, 89, 65, 58, 97,
             76, 62, 50, 63, 46, 45, 92, 96, 94, 89, 37, 52, 45, 79, 46, 78, 90, 46, 94, 52, 73, 95, 42, 81, 59, 72, 41, 100,
             41, 73, 81, 87, 43, 76, 96, 46, 60, 53, 67, 54, 74, 91, 40, 65, 46, 67, 37, 81, 85, 60, 40, 66, 47, 63, 89, 87,
             53, 39, 72, 54, 74, 66, 59, 77, 40, 73, 63, 45, 52, 88, 85, 51]
# Print the array
print(values)
plt.hist(values,"auto",(0,100),width=5)
plt.title('Final Exam Marks')
plt.xlabel('Marks')
plt.ylabel('Number of students')
# Display the histogram
plt.show()
```

[63, 41, 84, 97, 99, 46, 56, 91, 75, 43, 40, 92, 90, 75, 99, 43, 97, 51, 51, 44, 46, 62, 68, 72, 89, 65, 58, 97, 76, 62, 50, 6
3, 46, 45, 92, 96, 94, 89, 37, 52, 45, 79, 46, 78, 90, 46, 94, 52, 73, 95, 42, 81, 59, 72, 41, 100, 41, 73, 81, 87, 43, 76, 96,
46, 60, 53, 67, 54, 74, 91, 40, 65, 46, 67, 37, 81, 85, 60, 40, 66, 47, 63, 89, 87, 53, 39, 72, 54, 74, 66, 59, 77, 40, 73, 63,
45, 52, 88, 85, 51]


```python
import matplotlib.pyplot as plt

std_marks = [63, 41, 84, 97, 99, 46, 56, 91, 75, 43, 40, 92, 90, 75, 99, 43, 97, 51, 51, 44, 46, 62, 68, 72, 89, 65, 58, 97,
             76, 62, 50, 63, 46, 45, 92, 96, 94, 89, 37, 52, 45, 79, 46, 78, 90, 46, 94, 52, 73, 95, 42, 81, 59, 72, 41, 100,
             41, 73, 81, 87, 43, 76, 96, 46, 60, 53, 67, 54, 74, 91, 40, 65, 46, 67, 37, 81, 85, 60, 40, 66, 47, 63, 89, 87,
             53, 39, 72, 54, 74, 66, 59, 77, 40, 73, 63, 45, 52, 88, 85, 51]
# Print the array
print(values)
plt.hist(values, bins=10, edgecolor="yellow", color="green",cumulative=1) #bins reduce bars in histogram based on range
plt.title('Final Exam Marks')
plt.xlabel('Marks')
plt.ylabel('Number of students')
# Display the histogram
plt.show()
```

[63, 41, 84, 97, 99, 46, 56, 91, 75, 43, 40, 92, 90, 75, 99, 43, 97, 51, 51, 44, 46, 62, 68, 72, 89, 65, 58, 97, 76, 62, 50, 6
3, 46, 45, 92, 96, 94, 89, 37, 52, 45, 79, 46, 78, 90, 46, 94, 52, 73, 95, 42, 81, 59, 72, 41, 100, 41, 73, 81, 87, 43, 76, 96,
46, 60, 53, 67, 54, 74, 91, 40, 65, 46, 67, 37, 81, 85, 60, 40, 66, 47, 63, 89, 87, 53, 39, 72, 54, 74, 66, 59, 77, 40, 73, 63,
45, 52, 88, 85, 51]
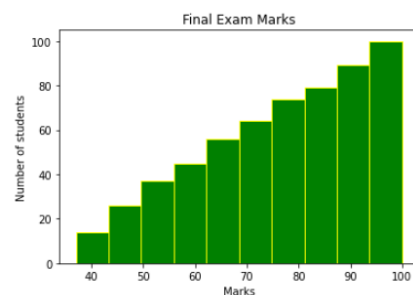
Here, the bins = 10, i.e the number of bins to be created is 10. Setting bins to an integer creates bins of equal size or width. As the bin size is changed so then the bin width would be changed.

In the plt.hist function from the Matplotlib library, the cumulative parameter controls whether the histogram should be cumulative or not. When cumulative is set to -1, it means that the histogram will be cumulative, but it will be displayed with the opposite orientation.

A cumulative histogram displays the cumulative distribution of a dataset, showing how the data accumulates over a range of values. The values on the x-axis represent the data points, and the y-axis represents the cumulative count or proportion. Default value is 'none'.

- cumulative=1: A regular cumulative histogram, starting from the lowest value and accumulating towards higher values.
- cumulative=-1: A cumulative histogram but displayed in reverse order, starting from the highest value and accumulating towards lower values.
- cumulative=None (or omitted): The default behavior, which is a standard histogram.
- **Setting type in a histogram:**

```python
import matplotlib.pyplot as plt

std_marks = [63, 41, 84, 97, 99, 46, 56, 91, 75, 43, 40, 92, 90, 75, 99, 43, 97, 51, 51, 44, 46, 62, 68, 72, 89, 65, 58, 97,
             76, 62, 50, 63, 46, 45, 92, 96, 94, 89, 37, 52, 45, 79, 46, 78, 90, 46, 94, 52, 73, 95, 42, 81, 59, 72, 41, 100,
             41, 73, 81, 87, 43, 76, 96, 46, 60, 53, 67, 54, 74, 91, 40, 65, 46, 67, 37, 81, 85, 60, 40, 66, 47, 63, 89, 87,
             53, 39, 72, 54, 74, 66, 59, 77, 40, 73, 63, 45, 52, 88, 85, 51]
# Print the array
print(values)
plt.hist(values, bins=10, edgecolor="blue", color="green",cumulative=-1, histtype="step")
plt.title('Final Exam Marks')
plt.xlabel('Marks')
plt.ylabel('Number of students')
# Display the histogram
plt.show()
```
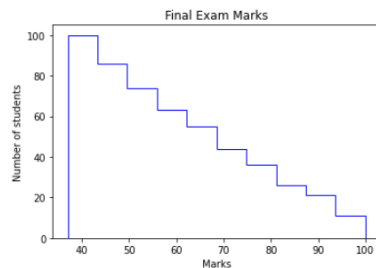
```
[63, 41, 84, 97, 99, 46, 56, 91, 75, 43, 40, 92, 90, 75, 99, 43, 97, 51, 51, 44, 46, 62, 68, 72, 89, 65, 58, 97, 76, 62, 50, 6
3, 46, 45, 92, 96, 94, 89, 37, 52, 45, 79, 46, 78, 90, 46, 94, 52, 73, 95, 42, 81, 59, 72, 41, 100, 41, 73, 81, 87, 43, 76, 96,
46, 60, 53, 67, 54, 74, 91, 40, 65, 46, 67, 37, 81, 85, 60, 40, 66, 47, 63, 89, 87, 53, 39, 72, 54, 74, 66, 59, 77, 40, 73, 63,
45, 52, 88, 85, 51]
```


histtype : This parameter is an optional parameter and it is used to draw type of histogram. {'bar', 'barstacked', 'step', 'stepfilled'}

- ## Changing orientation in a histogram:

```python
import matplotlib.pyplot as plt

std_marks = [63, 41, 84, 97, 99, 46, 56, 91, 75, 43, 40, 92, 90, 75, 99, 43, 97, 51, 51, 44, 46, 62, 68, 72, 89, 65, 58, 97,
             76, 62, 50, 63, 46, 45, 92, 96, 94, 89, 37, 52, 45, 79, 46, 78, 90, 46, 94, 52, 73, 95, 42, 81, 59, 72, 41, 100,
             41, 73, 81, 87, 43, 76, 96, 46, 60, 53, 67, 54, 74, 91, 40, 65, 46, 67, 37, 81, 85, 60, 40, 66, 47, 63, 89, 87,
             53, 39, 72, 54, 74, 66, 59, 77, 40, 73, 63, 45, 52, 88, 85, 51]
# Print the array
print(values)
plt.hist(values, bins=10, edgecolor="blue", color="green",cumulative=-1,orientation="horizontal" )
plt.title('Final Exam Marks')
plt.xlabel('Marks')
plt.ylabel('Number of students')
# Display the histogram
plt.show()
```

```
[63, 41, 84, 97, 99, 46, 56, 91, 75, 43, 40, 92, 90, 75, 99, 43, 97, 51, 51, 44, 46, 62, 68, 72, 89, 65, 58, 97, 76, 62, 50, 6
3, 46, 45, 92, 96, 94, 89, 37, 52, 45, 79, 46, 78, 90, 46, 94, 52, 73, 95, 42, 81, 59, 72, 41, 100, 41, 73, 81, 87, 43, 76, 96,
46, 60, 53, 67, 54, 74, 91, 40, 65, 46, 67, 37, 81, 85, 60, 40, 66, 47, 63, 89, 87, 53, 39, 72, 54, 74, 66, 59, 77, 40, 73, 63,
45, 52, 88, 85, 51]
```
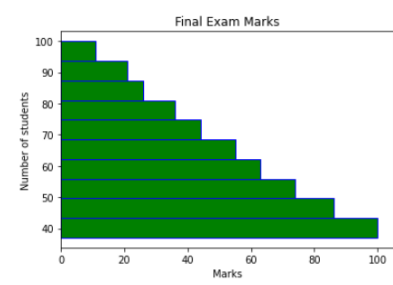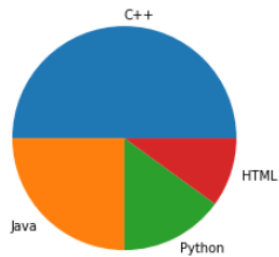

➢ **Pie Charts**

- ## Creating a pie chart:

```python
import matplotlib.pyplot as plt

popularityPercent = [50, 25, 15, 10]
labels=['C++','Java','Python','HTML']
plt.pie(popularityPercent,labels=labels)
plt.show()
```
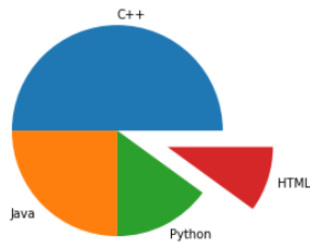
- **Exploding in a pie chart:**

Maybe you want one of the wedges to stand out? The explode parameter allows you to do that.
The explode parameter, if specified, and not None, must be an array with one value for each wedge.

Each value represents how far from the center each wedge is displayed:
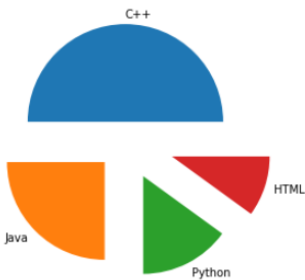
```
import matplotlib.pyplot as plt

popularityPercent = [50, 25, 15, 10]
labels=['C++','Java','Python','HTML']
ex=[0,0,0,0.5]
plt.pie(popularityPercent,labels=labels,explode=ex)
plt.show()
```
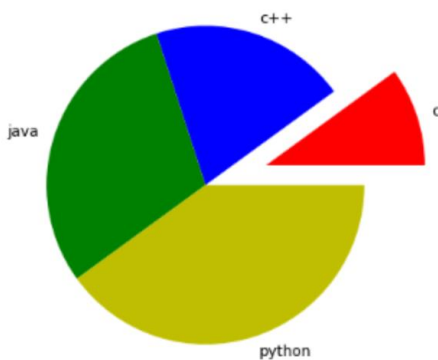


```
import matplotlib.pyplot as plt

popularityPercent = [50, 25, 15, 10]
labels=['C++','Java','Python','HTML']
ex=[0.2,0.3,0.4,0.5]
plt.pie(popularityPercent,labels=labels,explode=ex)
plt.show()
```


- **Color specification in a pie chart:**

```
x = [10,20,30,40]
y = ['c','C++','java','python']
ex = [0.4,0,0,0]
color = ["r","b","g","y"]
plt.pie(x, labels=y, explode = ex, colors = color)
plt.show()
```

- **Assigning percentages in a pie chart:**

In Matplotlib's pie chart, the autopct parameter is used to label the individual slices of the pie with their numeric values and optionally format them as percentages. This parameter allows you to display the exact values or percentages of each slice on the chart.

- If you set autopct=None or omit it, the chart won't display any labels.
- If you set autopct='%1.1f%%', it will display the percentages of each slice with one decimal place.
- If you set autopct='%1.2f%%', it will display the percentages of each slice with two decimal places.
- If you set autopct='%d', it will display the raw numeric value of each slice without decimal places.

```python
import matplotlib.pyplot as plt

popularityPercent = [50, 25, 15, 10]
labels=['C++','Java','Python','HTML']
plt.pie(popularityPercent,labels=labels,autopct="%1.3f%%")
plt.show()
```
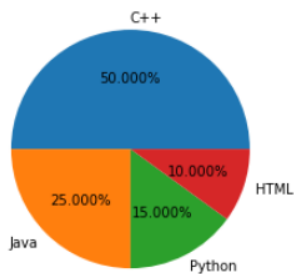
# ➢ Visualizing an actual dataset

```python
import matplotlib.pyplot as plt
import pandas as pd
df=pd.read_csv(r'students_data.csv')

#Showing histogram for grade column
plt.hist(df['Grade'],"auto",(0,100),width=3,color='y')
plt.xlabel("Grades")
plt.ylabel("Frequency")
plt.title("Grade Chart")
plt.show()

#Showing a histogram for age column
plt.hist(df['Age'],width=1,color='g')
plt.xlabel("Ages")
plt.ylabel("No of students")
plt.title("Age Chart")
plt.show()
```
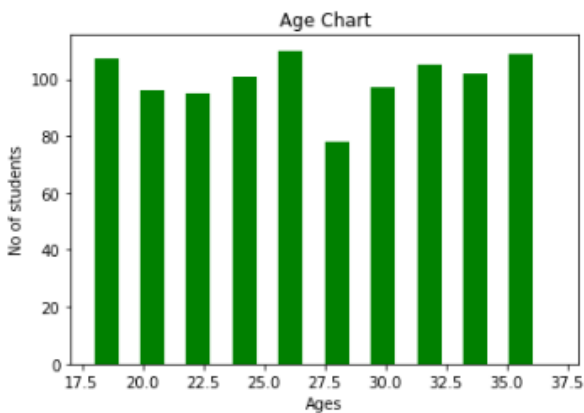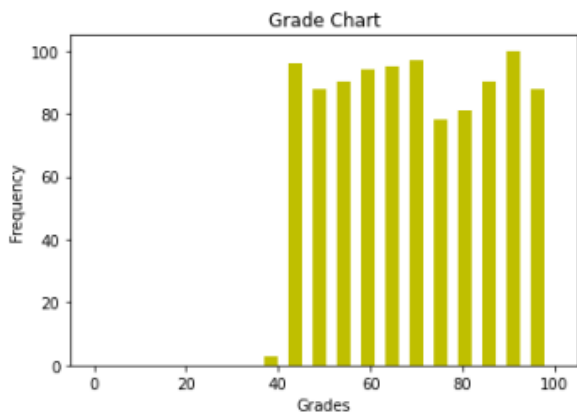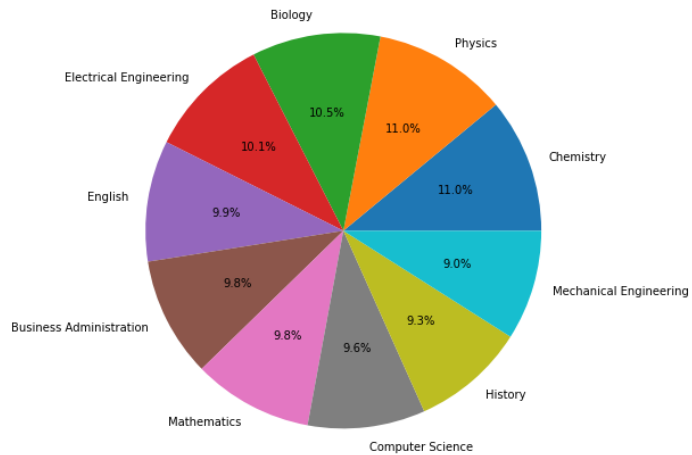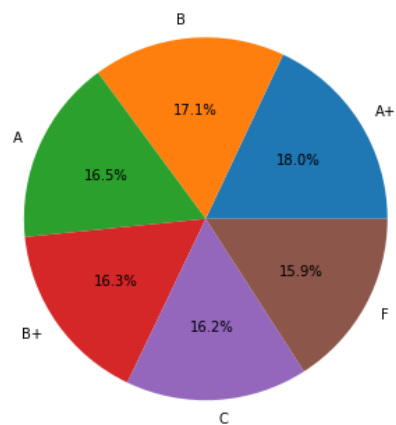
```python
#Plotting a pie chart for number of students in each department

import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv(r'students_data.csv')
plt.figure(figsize=(8, 8))
department_counts = df['Department'].value_counts()
plt.pie(department_counts, labels=department_counts.index, autopct='%1.1f%%')
plt.show()
```



```python
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv(r'students_data.csv')
def assign_grade(percentage):
    if percentage >= 90:
        return "A+"
    elif percentage >= 80:
        return "A"
    elif percentage >= 70:
        return "B+"
    elif percentage >= 60:
        return "B"
    elif percentage >= 50:
        return "C"
    else:
        return "F"
df['Grades'] = df['Grade'].apply(assign_grade)
plt.figure(figsize=(6, 6))
grade_counts = df['Grades'].value_counts()
plt.pie(grade_counts, labels=grade_counts.index, autopct='%1.1f%%')
plt.show()
```

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

df=pd.read_csv(r'students_data.csv')
# Group the data by the 'Grade' column and count the number of students in each grade
grade_counts = df['Grade'].value_counts()
# Sort the grades in ascending order
sorted_grades = sorted(grade_counts.index)
# Get the corresponding student counts for the sorted grades
student_counts = [grade_counts[grade] for grade in sorted_grades]
# Create a bar graph
plt.figure(figsize=(10, 6))
plt.bar(sorted_grades, student_counts,width=0.05)
plt.xlabel('Grade')
plt.ylabel('Number of Students')
plt.title('Number of Students by Grade')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.yticks(rotation=45)
plt.show()
```
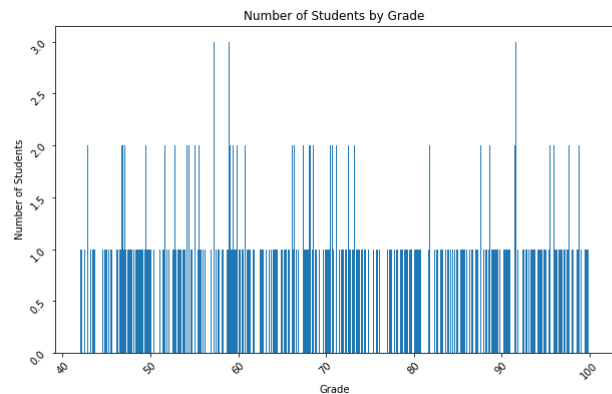

Number of Students by Grade

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

df=pd.read_csv(r'students_data.csv')
# Group the data by the 'Grade' column and count the number of students in each grade
grade_counts = df['Grade'].value_counts()
# Sort the grades in ascending order
sorted_grades = sorted(grade_counts.index)
# Get the corresponding student counts for the sorted grades
student_counts = [grade_counts[grade] for grade in sorted_grades]
# Create a scatterplot
plt.figure(figsize=(10, 6))
plt.scatter(sorted_grades, student_counts, marker='*', color='b', label='Students')
plt.xlabel('Grade')
plt.ylabel('Number of Students')
plt.title('Scatterplot of Students by Grade')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.legend()
plt.grid(True)
plt.show()
```


Scatterplot of Students by Grade
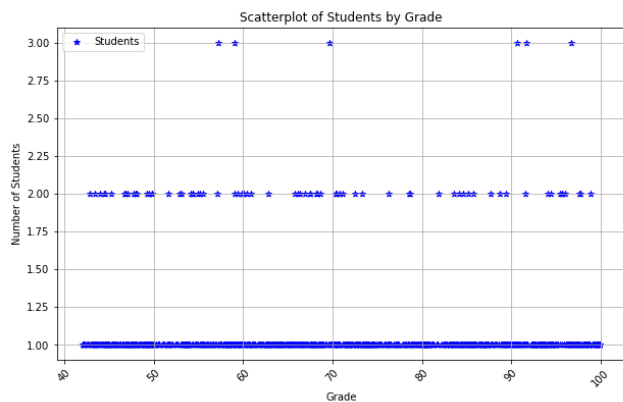
➢ Stackplot

A stack plot, also known as a stacked area chart, is a type of data visualization that is used to display multiple variables over a continuous interval, typically representing changes in those variables over time.

Stack plots are effective for visualizing data with multiple categories or components, allowing you to see the relative importance of each component and how it changes over time or along a continuous axis. They are commonly used in areas such as finance to represent the contributions of various sectors to a market index, in ecology to show the composition of different species in an ecosystem, and in many other fields where tracking the evolution of multiple components is important.

- **Examples:**

```python
import matplotlib.pyplot as plt

days=[1,2,3,4,5,6,7]
expenses=[100,200,50,25,500,800,20]

plt.stackplot(days,expenses)
plt.show()
```
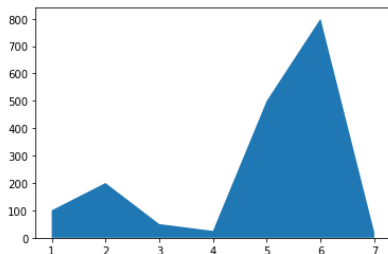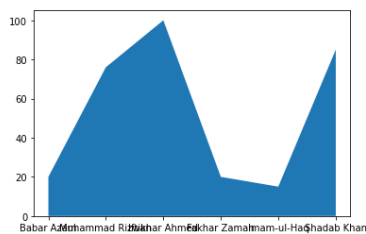


```python
import matplotlib.pyplot as plt

Players=["Babar Azam","Muhammad Rizwan","Iftikhar Ahmed","Fakhar Zaman","Imam-ul-Haq","Shadab Khan"]
Runs=[20,76,100,20,15,85]

plt.stackplot(Players,Runs)
plt.show()
```
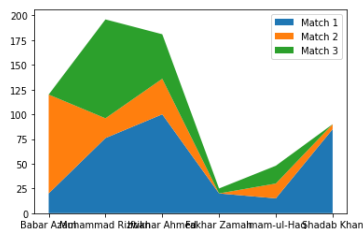


```python
import matplotlib.pyplot as plt

Players=["Babar Azam","Muhammad Rizwan","Iftikhar Ahmed","Fakhar Zaman","Imam-ul-Haq","Shadab Khan"]
Runs_in_match1=[20,76,100,20,15,85]
Runs_in_match2=[100,20,36,0,15,5]
Runs_in_match3=[0,100,45,5,18,0]
l=["Match 1","Match 2","Match 3"]
plt.stackplot(Players,Runs_in_match1,Runs_in_match2,Runs_in_match3,labels=l)
plt.legend()
plt.show()
```

Lab Tasks:

1. Write a Python program to display a bar chart and a pie chart for student height data taken from your class. Use different color for each bar and take at least 20 students.
2. You have a list of cities and their respective populations. Create a horizontal bar graph to visualize the populations of these cities.
3. You have a list of ice cream flavors and the number of scoops sold for each flavor. Create a pie chart to show the distribution of ice cream sales.
4. Create a pie chart to show the distribution of student ages by dividing them into groups (e.g., 18-25, 26-30, 31-35, 36 and above). Use the provided dataset for data.
5. You're interested in the gender distribution among the students in your dataset. Create a pie chart to visualize the percentage of male and female students.
6. Write a Python program to draw a scatter plot comparing two subject marks of Mathematics and Science. Use marks of 10 students and also add legends for better understanding of the data.
7. Write a Python program to draw a scatter plot to find sea level rise in past 100 years. Use the provided dataset.