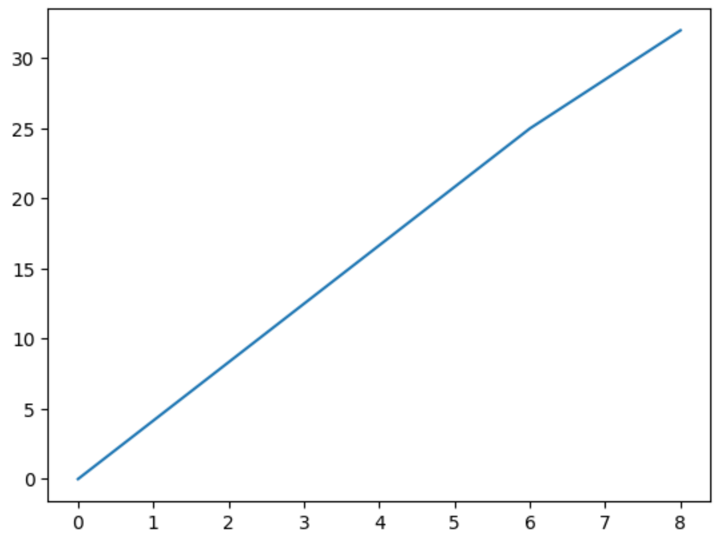
Matplotlib

To make necessary statistical inferences, it becomes necessary to visualize your data and Matplotlib is one such solution for the Python users. It is a very powerful plotting library useful for those working with Python and NumPy. The most used module of Matplotib is Pyplot which provides an interface like MATLAB but instead, it uses Python and it is open source.

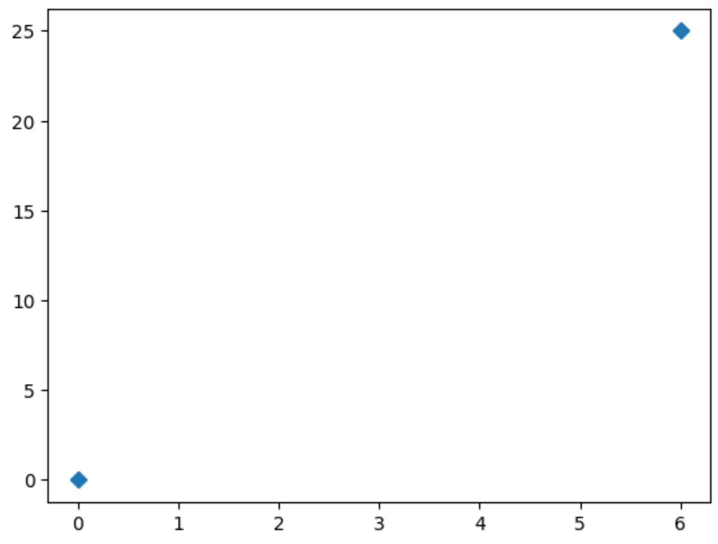
```
import matplotlib.pyplot as plt
import numpy as np
```

A **line plot**, also known as a line chart or a line graph, is a graphical representation used to display data points along a number line or x-axis, connected by straight lines. Line plots are commonly used in various fields to illustrate trends, patterns, and relationships within a dataset. They are particularly useful for visualizing continuous data, such as time series data or data that varies across a continuous range.

```
# Drawing a simple line
xpoints = np.array([0, 6, 8])
ypoints = np.array([0, 25, 32])

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



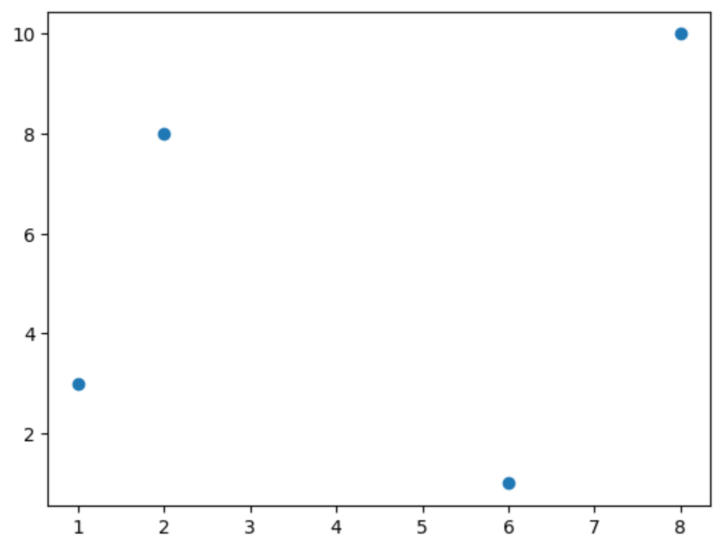
```
# Plotting Without Line
xpoints = np.array([0, 6])
ypoints = np.array([0, 25])

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



Different Values for marker

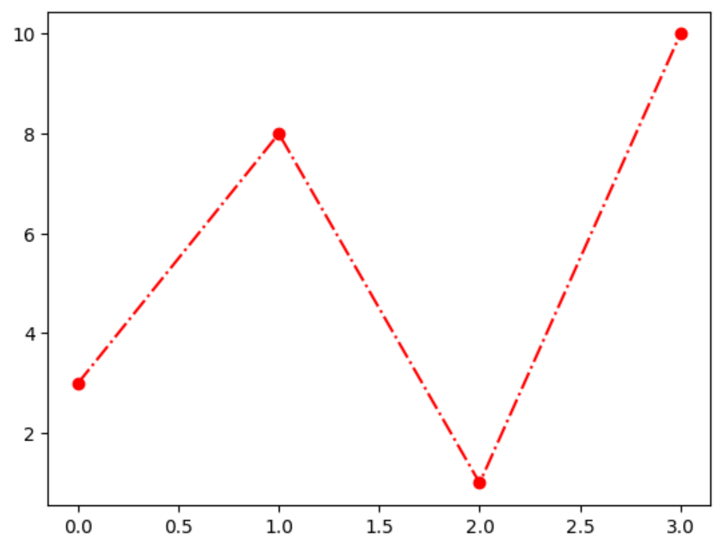
'o' Circle '*' Star '.' Point ',' Pixel 'x' X 'X' X (filled) '+' Plus 'P' Plus (filled) 's' Square 'D' Diamond 'd' Diamond (thin) 'p' Pentagon 'H' Hexagon 'h' Hexagon 'v' Triangle Down '^' Triangle Up '<' Triangle Left '>' Triangle Right '1' Tri Down '2' Tri Up '3' Tri Left '4' Tri Right '|' Vline '_' Hline

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

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



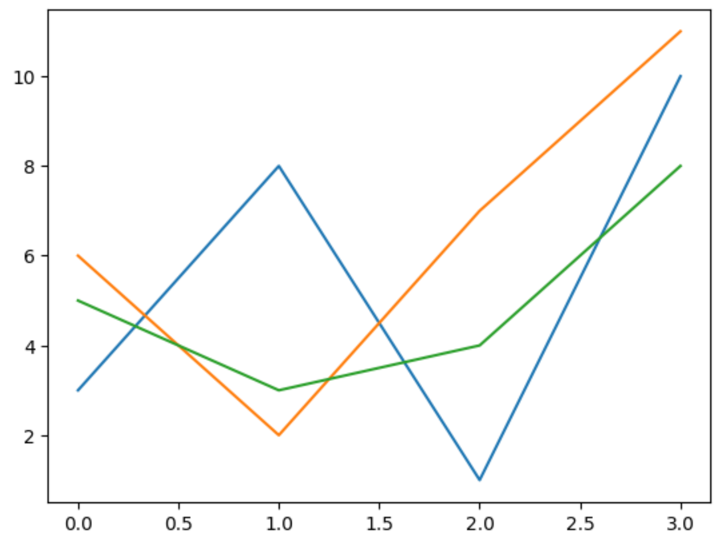
```
# Without explicit points for x-axis
ypoints = np.array([3, 8, 1, 10, 5, 7])

plt.plot(ypoints, marker="o")
plt.show()
```

```
# Format Strings fmt
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, 'o-.r')
plt.show()
```

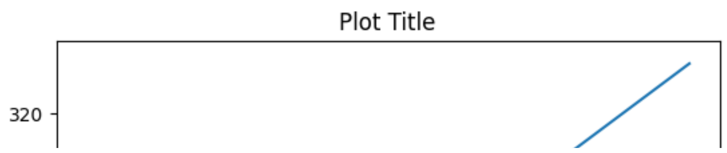


```
# Marker Style
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o', ms = 10, mec = 'g', mfc='y')
plt.show()
```

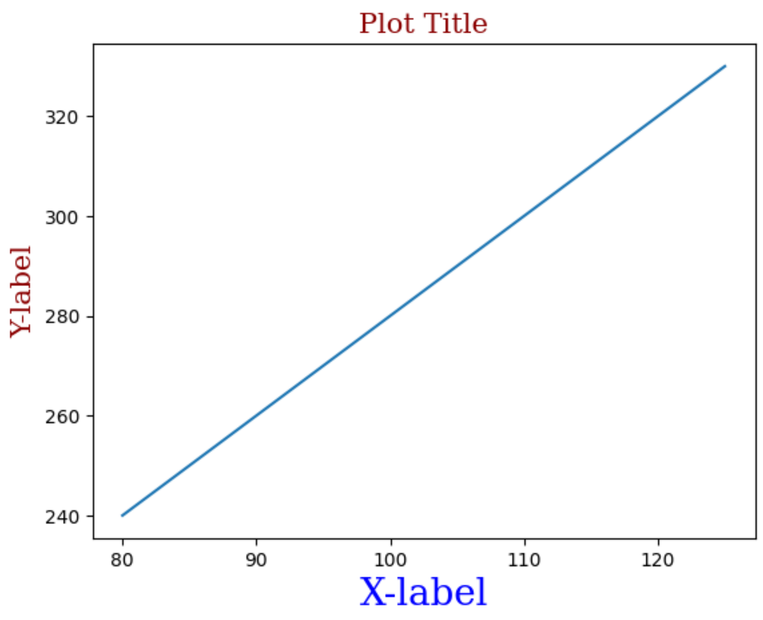
```
# Line Style
ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, linestyle = 'dashed', color = 'red', linewidth='3')
plt.show()


# Multiple Lines
y1 = np.array([3, 8, 1, 10])
y2 = np.array([6, 2, 7, 11])
y3 = np.array([5, 3, 4, 8])

plt.plot(y1)
plt.plot(y2)
plt.plot(y3)

plt.show()
```



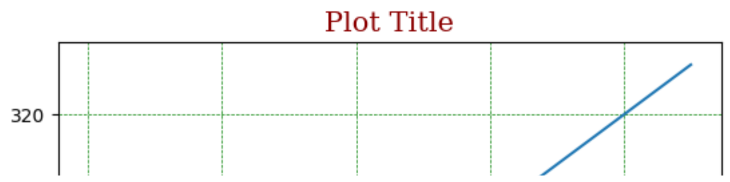
```
# Multiple Lines with X-axis
x1 = np.array([0, 1, 2, 3])
y1 = np.array([3, 8, 1, 10])
x2 = np.array([0, 1, 2, 3])
y2 = np.array([6, 2, 7, 11])

plt.plot(x1, y1, x2, y2)
plt.show()


# Add title and labels to the x- and y-axis:

x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])


plt.xlabel("X-label")
plt.ylabel("Y-label")
plt.title("Plot Title")

plt.plot(x, y)
plt.show()
```



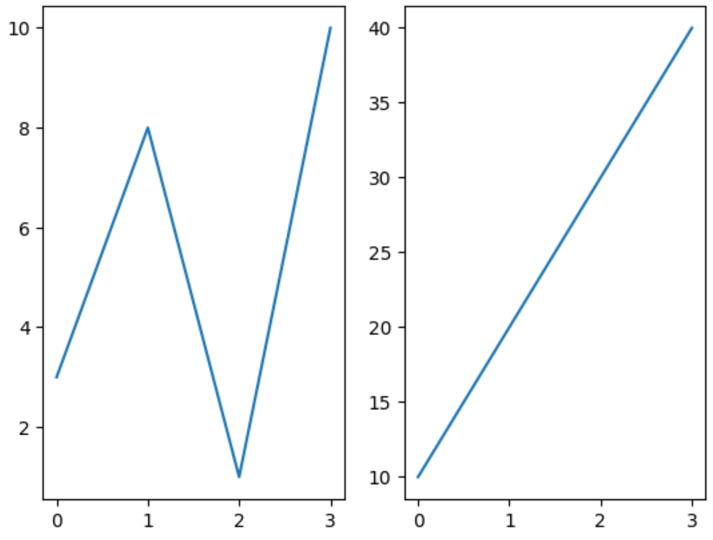
```
# Formatted title and labels
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

font1 = {'family':'serif','color':'blue','size':20}
font2 = {'family':'serif','color':'darkred','size':15}

plt.xlabel("X-label", fontdict = font1)
plt.ylabel("Y-label", fontdict = font2)
plt.title("Plot Title", fontdict = font2)


plt.plot(x, y)
plt.show()
```



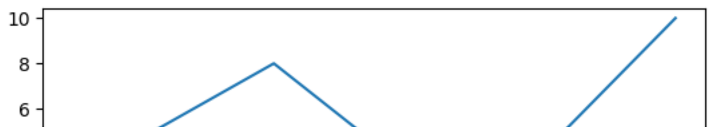
```
# Adding grid line
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

font1 = {'family':'serif','color':'blue','size':20}
font2 = {'family':'serif','color':'darkred','size':15}

plt.xlabel("X-label", fontdict = font1)
plt.ylabel("Y-label", fontdict = font2)
plt.title("Plot Title", fontdict = font2)


plt.plot(x, y)
plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)
plt.show()
```



```
# Subplots
#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(1, 2, 1)
plt.plot(x,y)

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(1, 2, 2)
plt.plot(x,y)

plt.show()
```



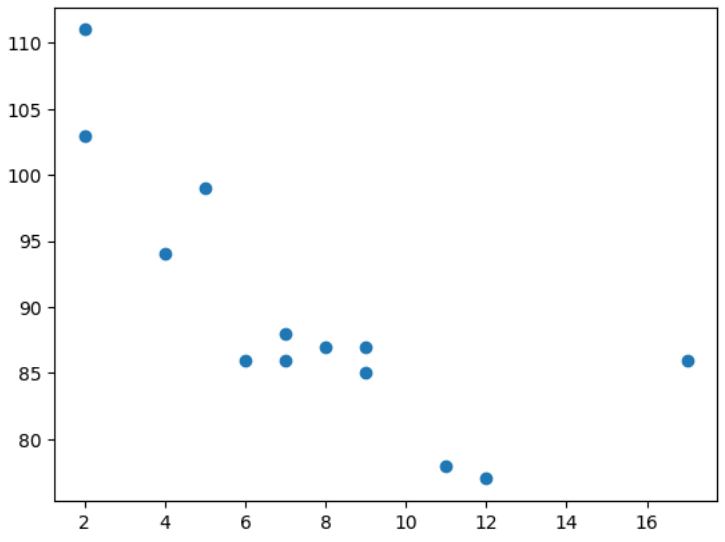
```
# Subplots
#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(2, 1, 1)
plt.plot(x,y)

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(2, 1, 2)
plt.plot(x,y)

plt.show()
```



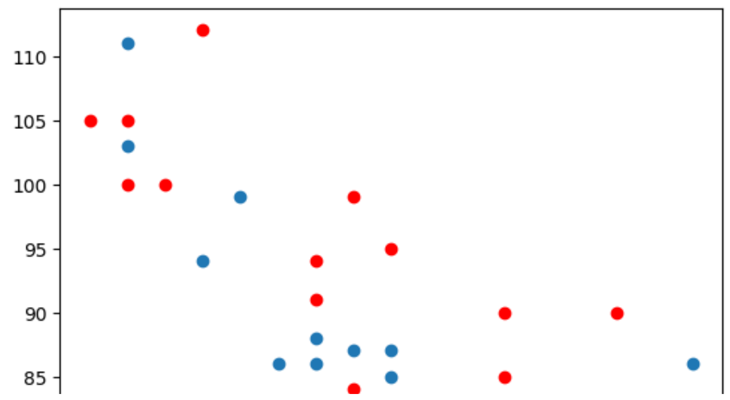
```
# Subplots with title
#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(1, 2, 1)
plt.plot(x,y)
plt.title("SALES")

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(1, 2, 2)
plt.plot(x,y)
plt.title("INCOME")

plt.suptitle("MY SHOP")
plt.show()
```

A **scatter plot** is a type of data visualization used to display individual data points as dots on a two-dimensional plane. It is particularly useful for showing the relationship between two continuous variables. Each dot on the scatter plot represents a single data point, with its position determined by the values of the two variables being compared.

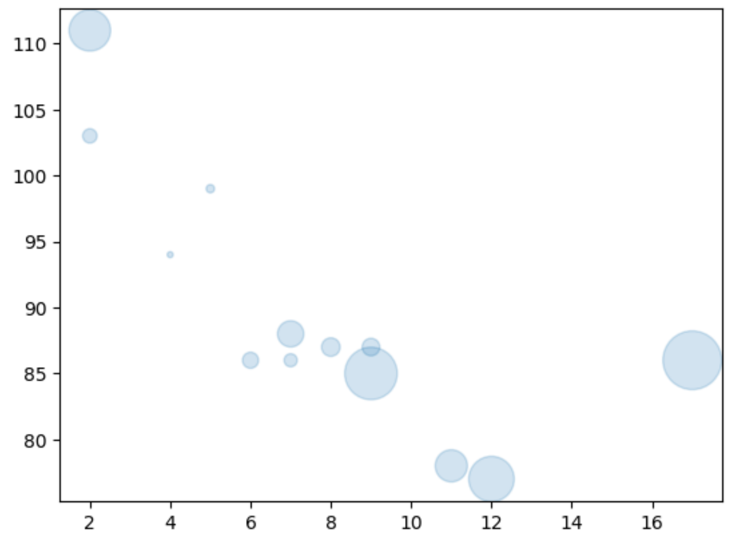
```
# Creating Scatter Plots
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

plt.scatter(x, y)
plt.show()
```



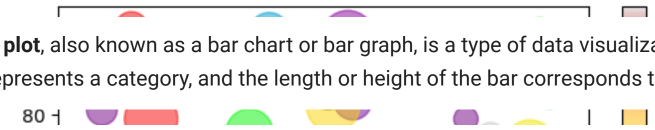
```
# Comparing two plots
#day one, the age and speed of 13 cars:
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x, y)

#day two, the age and speed of 15 cars:
x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
y = np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
plt.scatter(x, y, color='red')

plt.show()
```



```
# Adjusting the size of dots
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75])

plt.scatter(x, y, s=sizes, alpha=0.2)

plt.show()
```

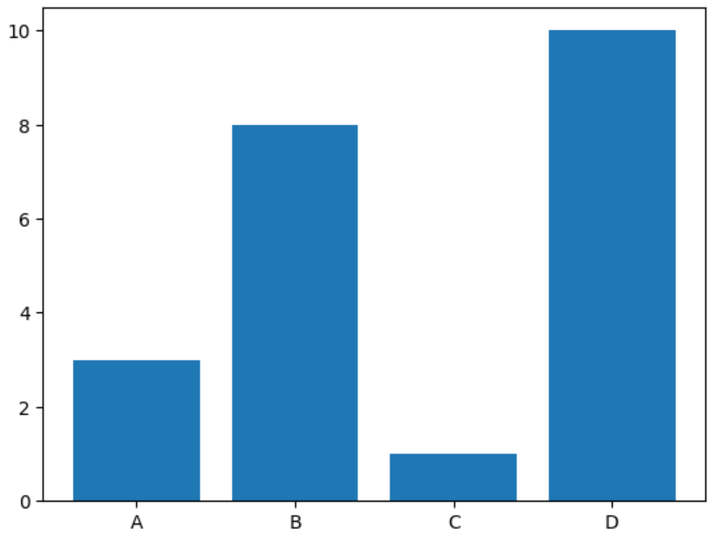


```
# Dot size and color
x = np.random.randint(100, size=(100))
y = np.random.randint(100, size=(100))
colors = np.random.randint(100, size=(100))
sizes = 10 * np.random.randint(100, size=(100))

plt.scatter(x, y, c=colors, s=sizes, alpha=0.5, cmap='nipy_spectral')

plt.colorbar()

plt.show()
```

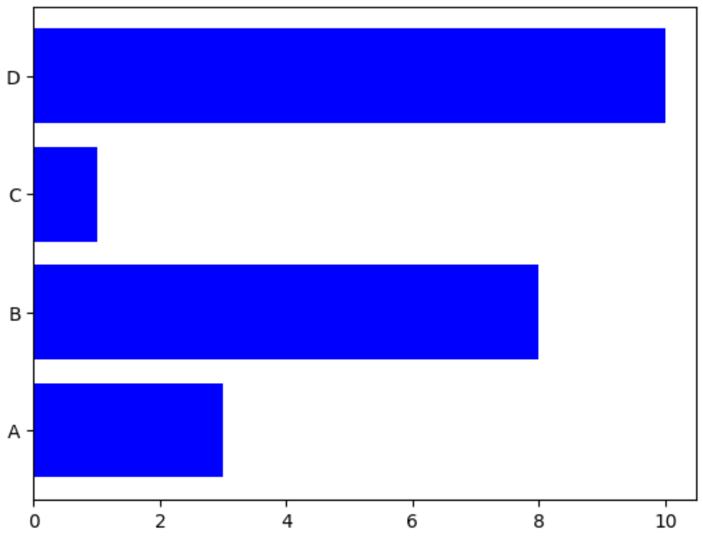


A **bar plot**, also known as a bar chart or bar graph, is a type of data visualization used to display categorical data using rectangular bars. Each bar represents a category, and the length or height of the bar corresponds to the value or frequency of the category it represents.

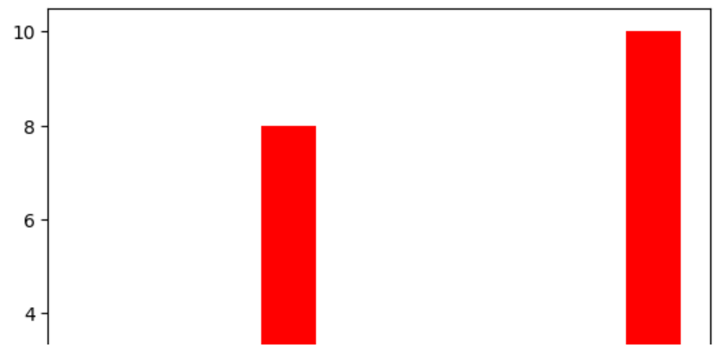
```
# Bar Plots (Vertical)
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x,y)
plt.show()
```



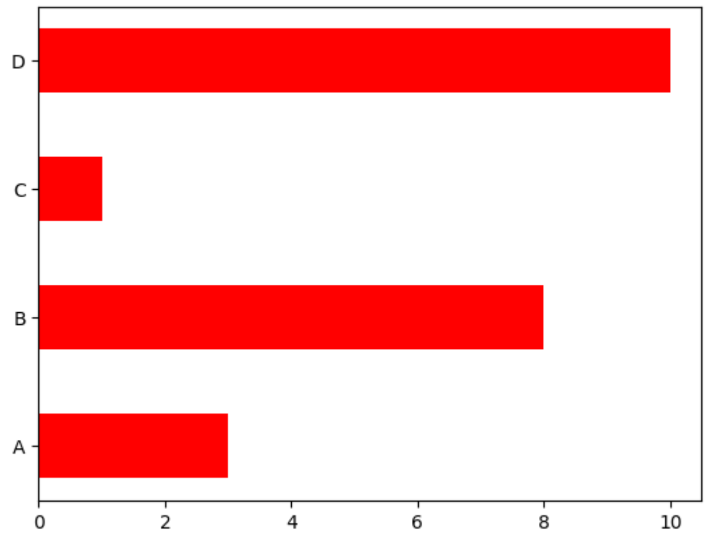
```
# Bar Plots (Horizontal)
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.barh(x, y, color='b')
plt.show()
```



```
# Changing the bar width
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x,y, color='red', width=0.3)
plt.show()
```

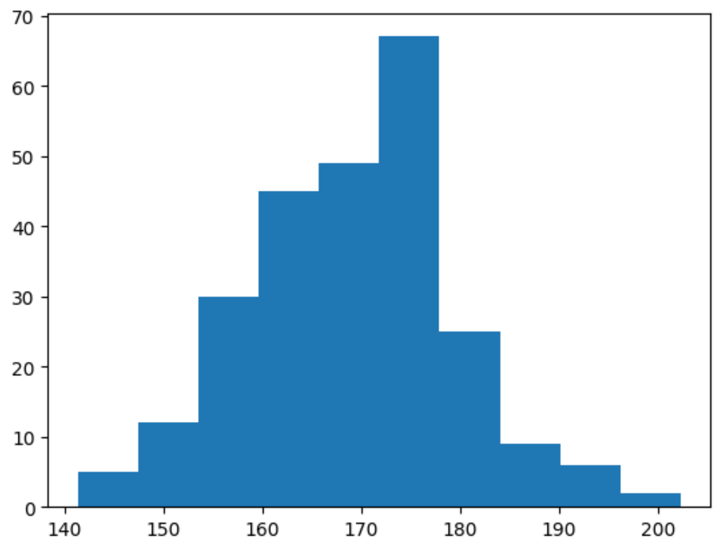


```
# Changing the bar height
x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.barh(x,y, color='red', height=0.5)
plt.show()
```



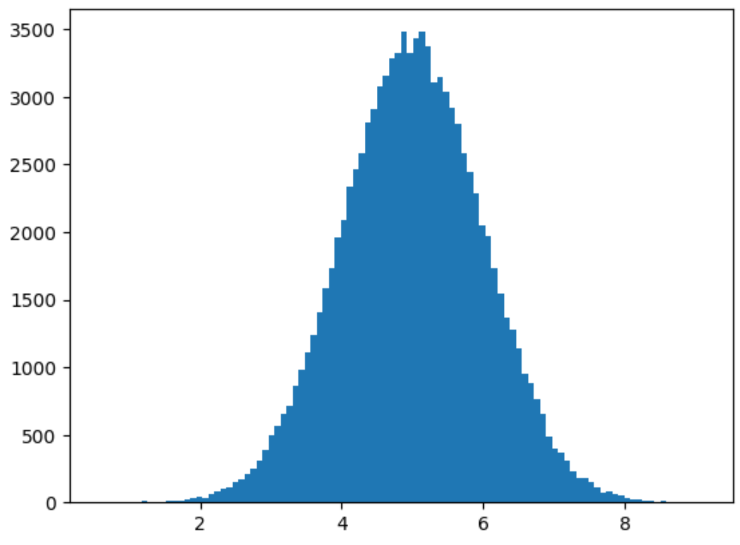
A **histogram** is a graphical representation used to visualize the distribution of continuous data. It displays the frequencies or counts of data points that fall within specified intervals, or "bins," along a continuous range. Histograms are particularly useful for understanding the shape, central tendency, and spread of a dataset.

```
# Histogram
x = np.random.normal(170, 10, 250)
#print(x)
plt.hist(x)
plt.show()
```



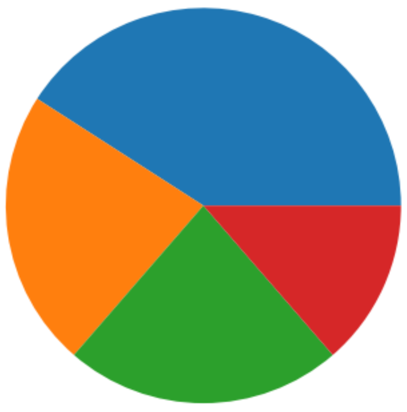
```
# Normal Distribution
x = np.random.normal(5.0, 1.0, 100000)
```

```
plt.hist(x, 100)
plt.show()
```

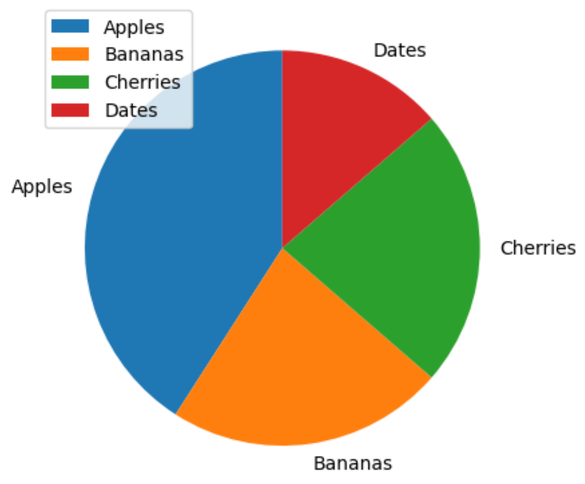


A **pie chart** is a circular data visualization that is divided into slices, each representing a proportion or percentage of a whole. Pie charts are commonly used for displaying the distribution of categorical data and the relative sizes of different parts in relation to the whole.

```
# Pie chart
y = np.array([45, 25, 25, 15])

plt.pie(y)
plt.show()
```



```
# Pie with labels
y = np.array([45, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

plt.pie(y, labels = mylabels, startangle = 90)
plt.legend()
plt.show()
```



✓ 0s completed at 10:09 AM