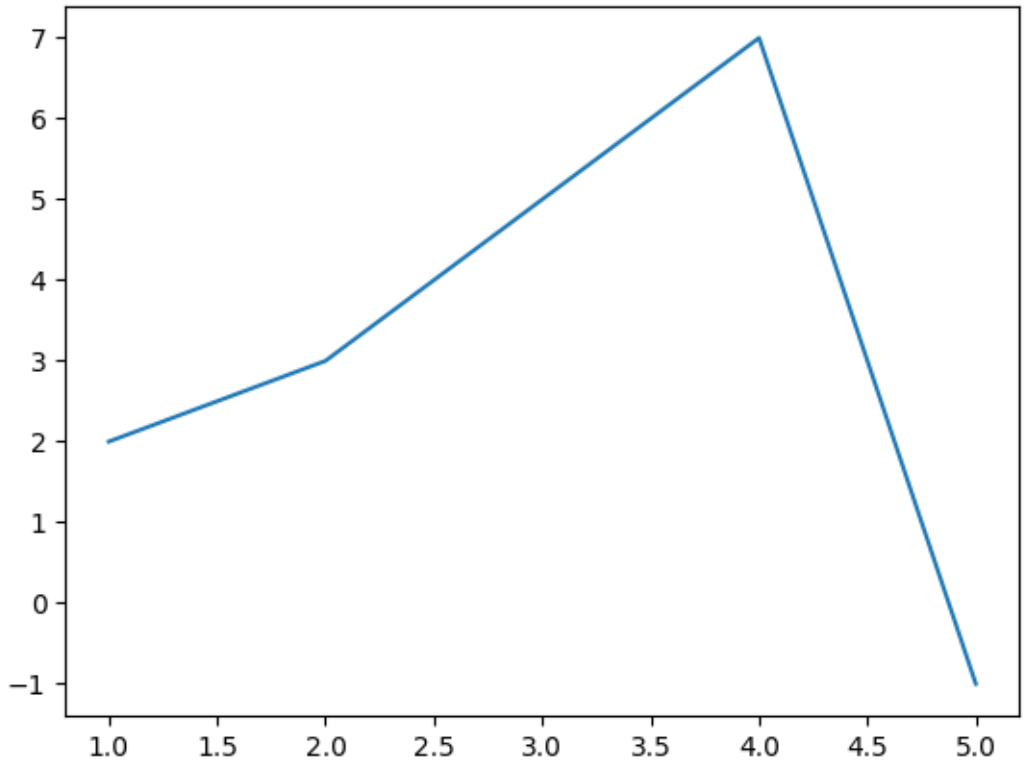
Matplotlib Fundamentals

Category	*Details*
Definition	A Python 2D plotting library used for creating static, interactive, and animated visualizations.
Main Module	matplotlib.pyplot
Developed In	Python
Common Use	Data visualization in data science, machine learning, and research

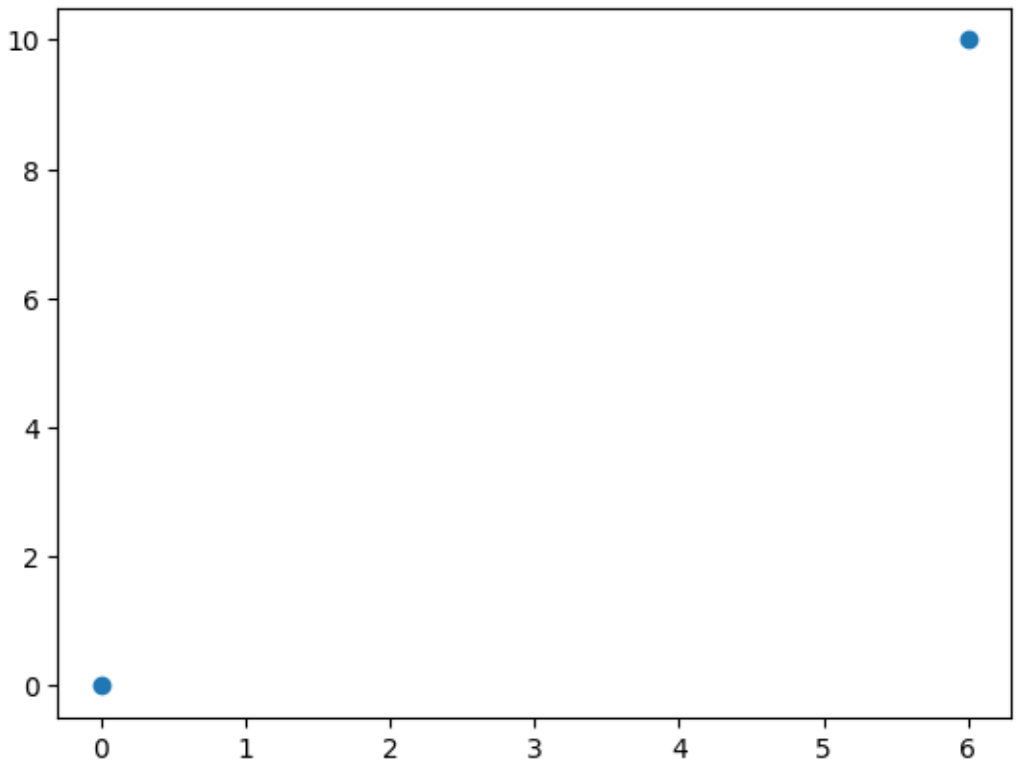
Plot Type	*Description*
Line Plot	Shows trends over time or continuous data using lines.
Bar Plot	Displays categorical data with rectangular bars.
Histogram	Shows distribution of a dataset by grouping into bins.
Scatter Plot	Plots points for two variables to show relationships or clusters.
Pie Chart	Represents data as slices of a pie to show proportions.
Box Plot	Shows data distribution with median, quartiles, and

[1]:
```
# Matplotib.pyplot is a plotting library for the Python used for 2D graphics␣
 ↪and visualizations.
# Numpy is a library for numerical computations in Python, often used for␣
 ↪handling arrays and matrices.
# This script imports these libraries, which are commonly used for data␣
 ↪visualization and numerical operations.

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

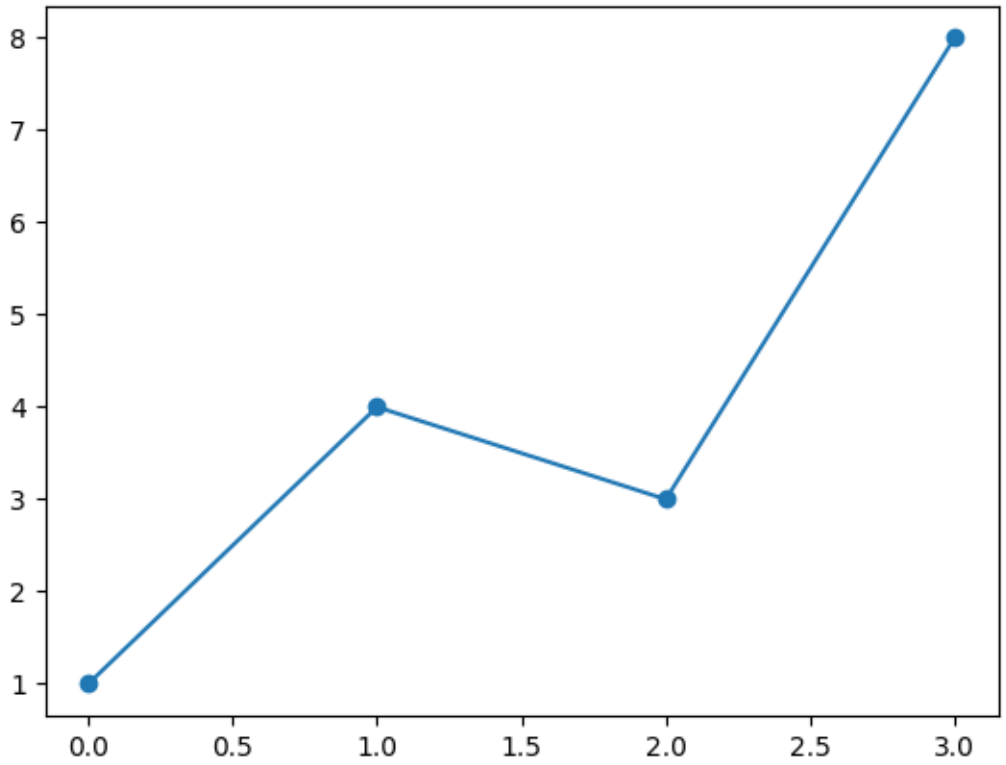
[2]:
```
# Display the plot

x=[1,2,3,4,5]
y=[2,3,5,7,-1]
plt.plot(x, y)
plt.show()
```



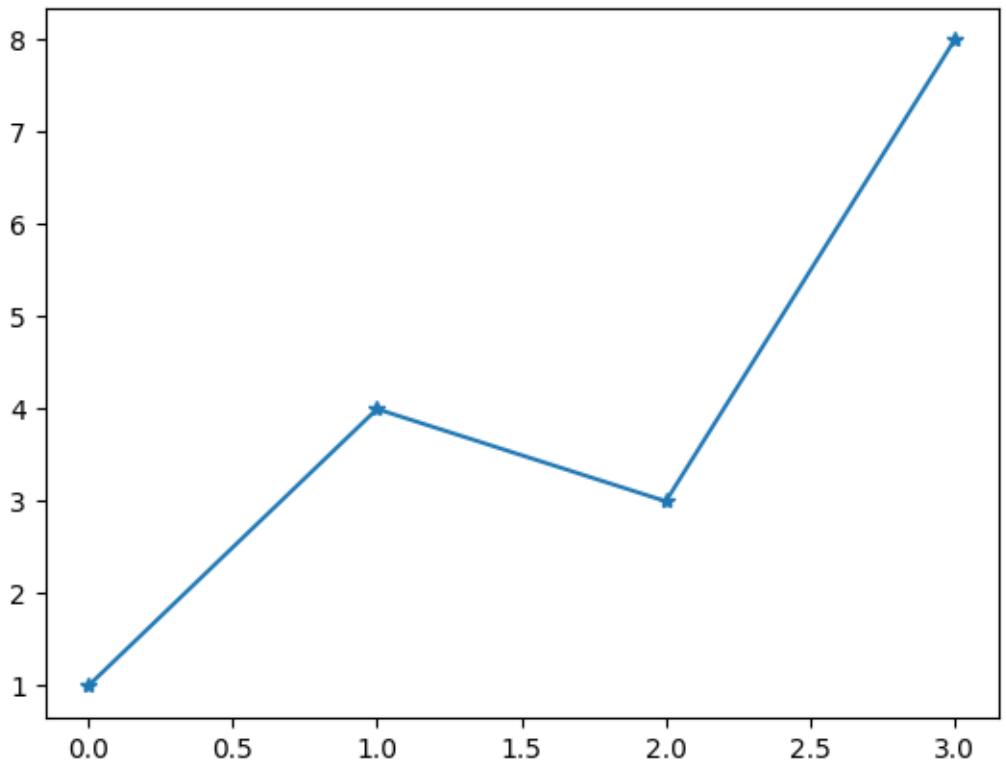
[3]:
```
# This example shows how to plot points as circles in a scatter plot.

# Define x-coordinates for the point
# Define y-coordinates for the point
# Plot the points as circles
# Display the points on the plot

xpoint = np.array([0, 6])
ypoint = np.array([0, 10])
plt.plot(xpoint, ypoint, 'o')
plt.show()
```



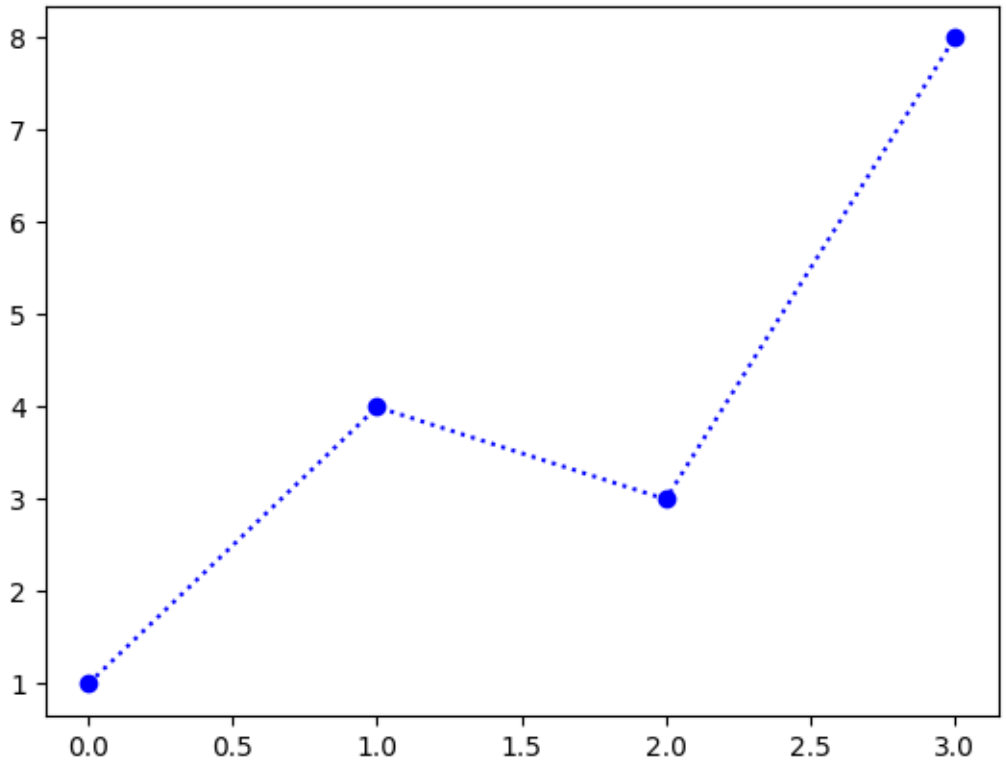
[4]:
```
# This example shows how to plot a line with markers.

# markers is used to plot lines with markers
# scatter plot is used to plot points with markers
# Display the plot with markers

xpoint = np.array([1,4,3,8])
plt.plot(xpoint, marker='o')
plt.show()
```



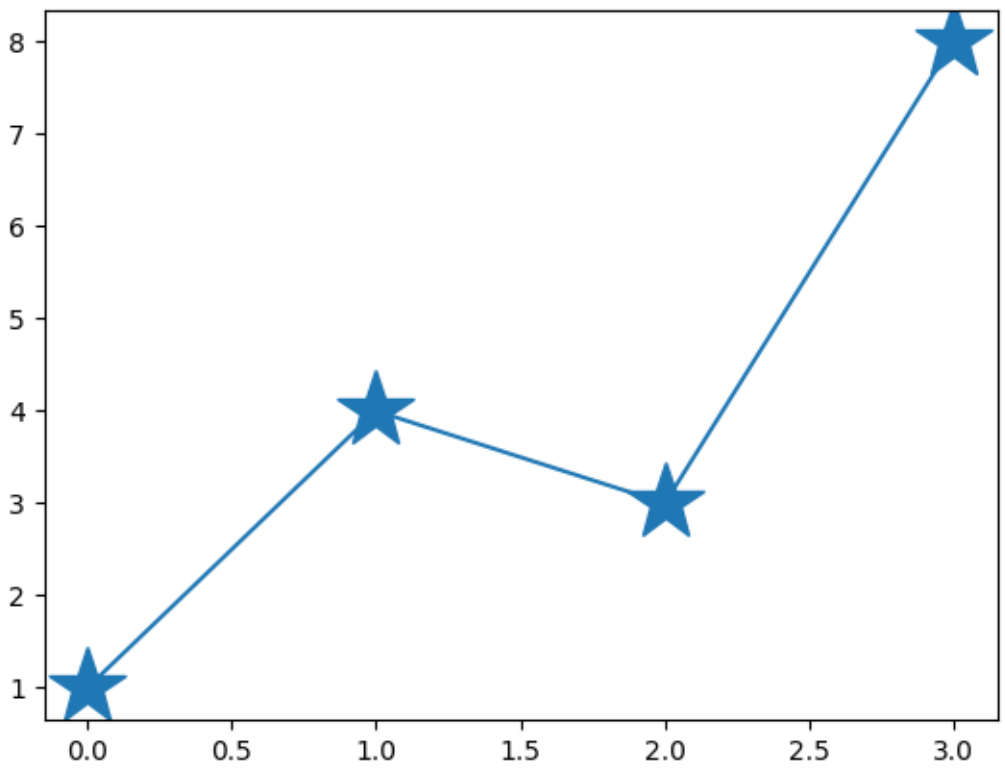
[5]:
```
# This example shows how to set the marker style in a plot.

# point denotes the marker style, here '*' is used to denote star markers

xpoint = np.array([1,4,3,8])
plt.plot(xpoint, marker='*')
plt.show()
```



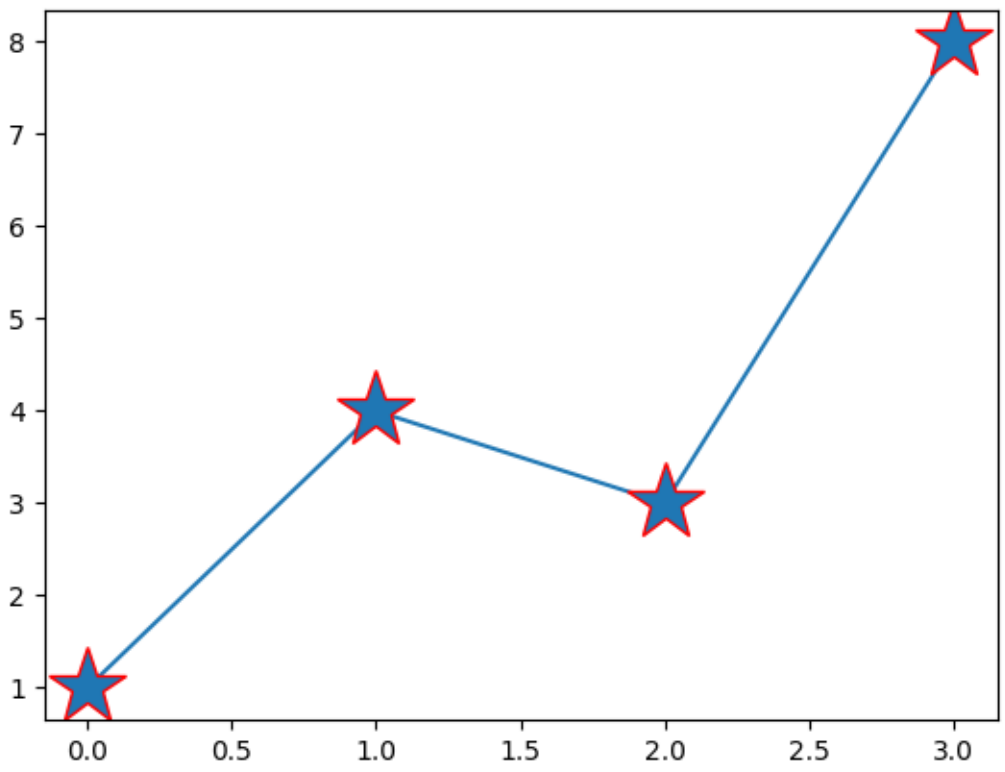
[6]:
```
# This example shows how to set the marker style and line style in a plot.

# 'o:b' denotes the marker style as circle and the line style as dotted blue␣
 ↪line

xpoint = np.array([1,4,3,8])
plt.plot(xpoint, 'o:b')
plt.show()
```



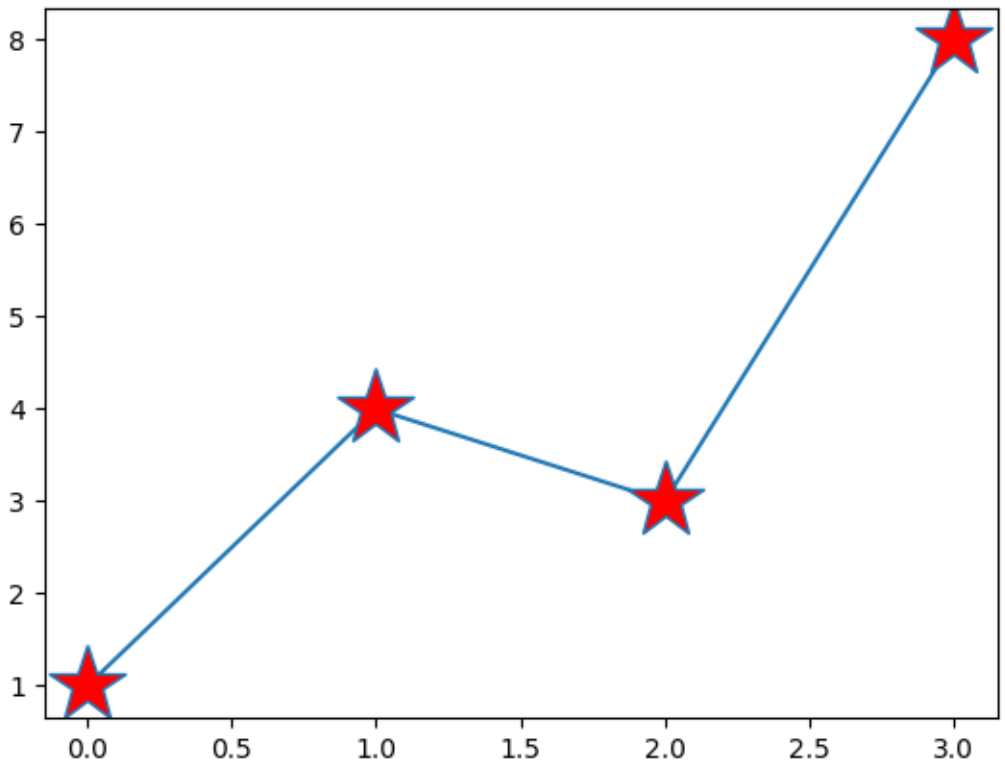
[7]:
```
# This example shows how to set the marker size in a plot.

# 'ms' denotes the marker size, here it is set to 30

xpoint = np.array([1,4,3,8])
plt.plot(xpoint, marker='*', ms=30)
plt.show()
```



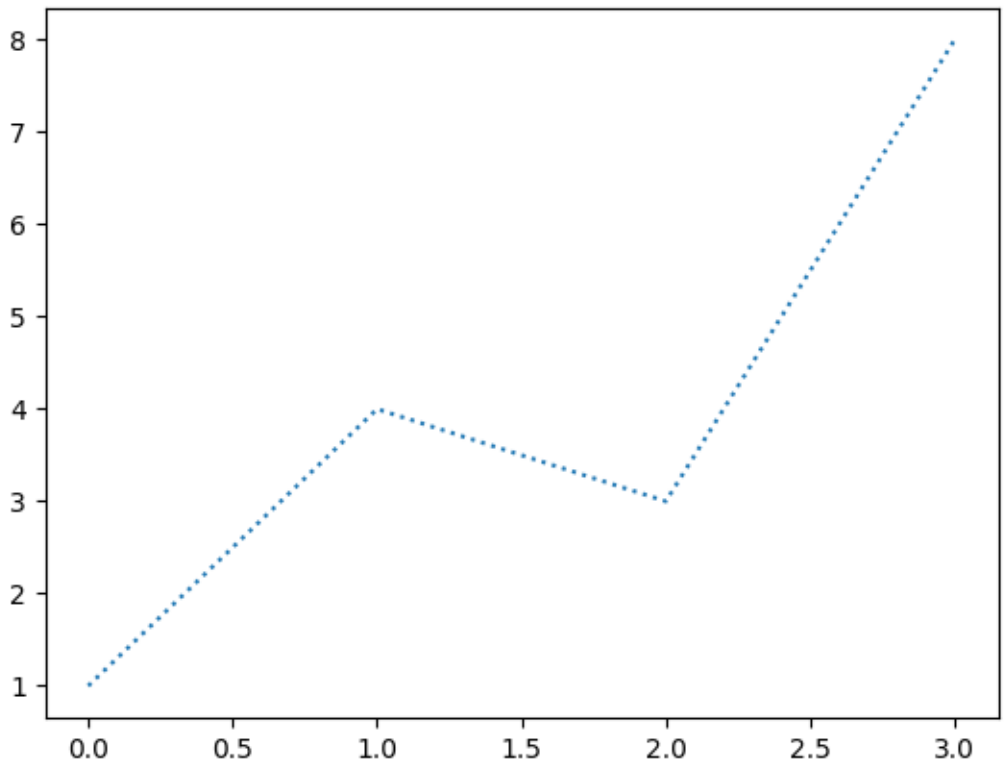
[8]:
```
# This example shows how to set the marker edge color in a plot.

# 'mec' denotes the marker edge color, here it is set to red

xpoint = np.array([1,4,3,8])
plt.plot(xpoint, marker='*', ms=30, mec='r')
plt.show()
```



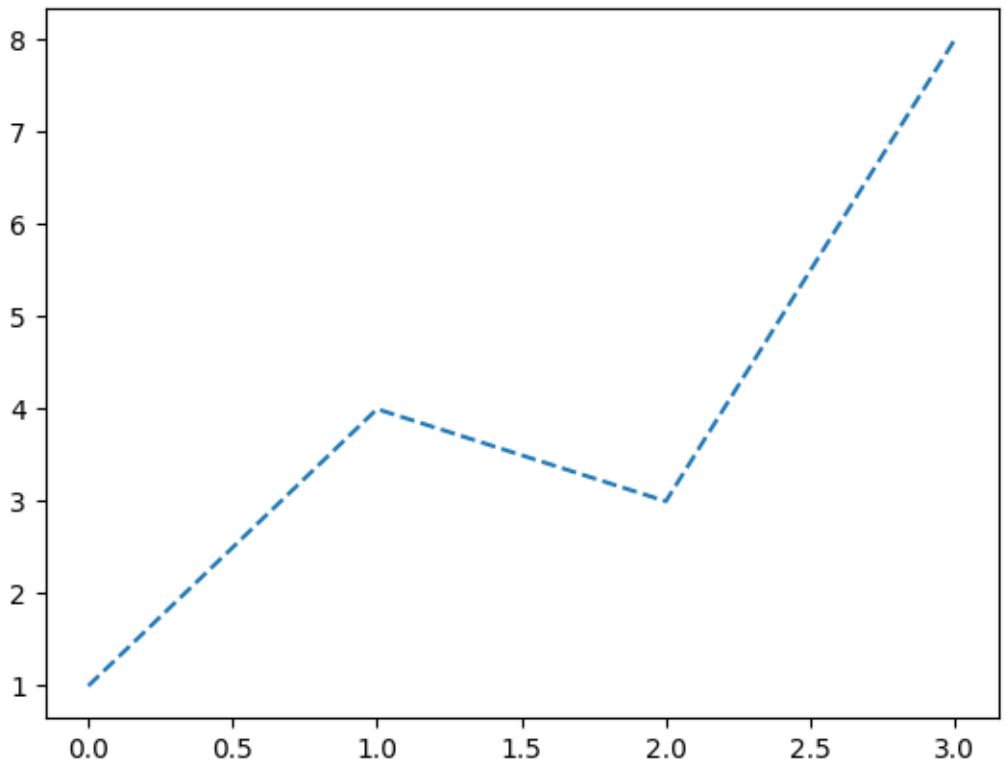
[9]:
```
# This example shows how to set the marker face color in a plot.

# 'mfc' denotes the marker face color, here it is set to red

xpoint = np.array([1,4,3,8])
plt.plot(xpoint, marker='*', ms=30, mfc='red')
plt.show()
```



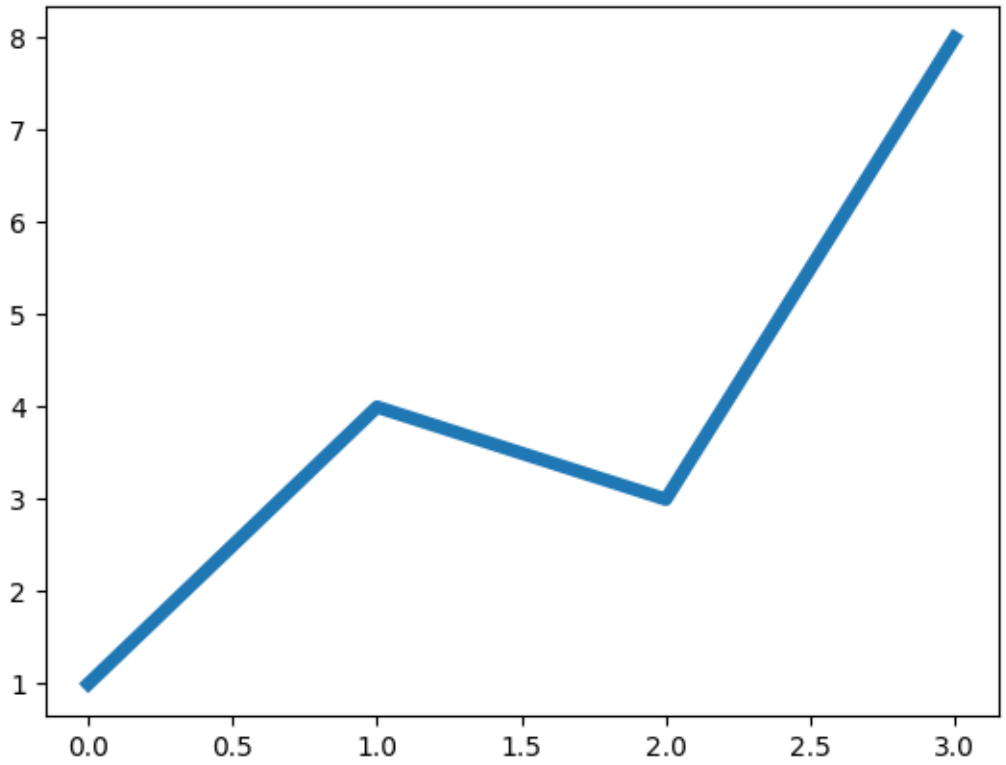
[10]:
```
# This example shows how to set the line style to dotted in a plot.

# 'ls' denotes the line style, here it is set to dotted

x=np.array([1,4,3,8])
plt.plot(x, ls='dotted')
plt.show()
```



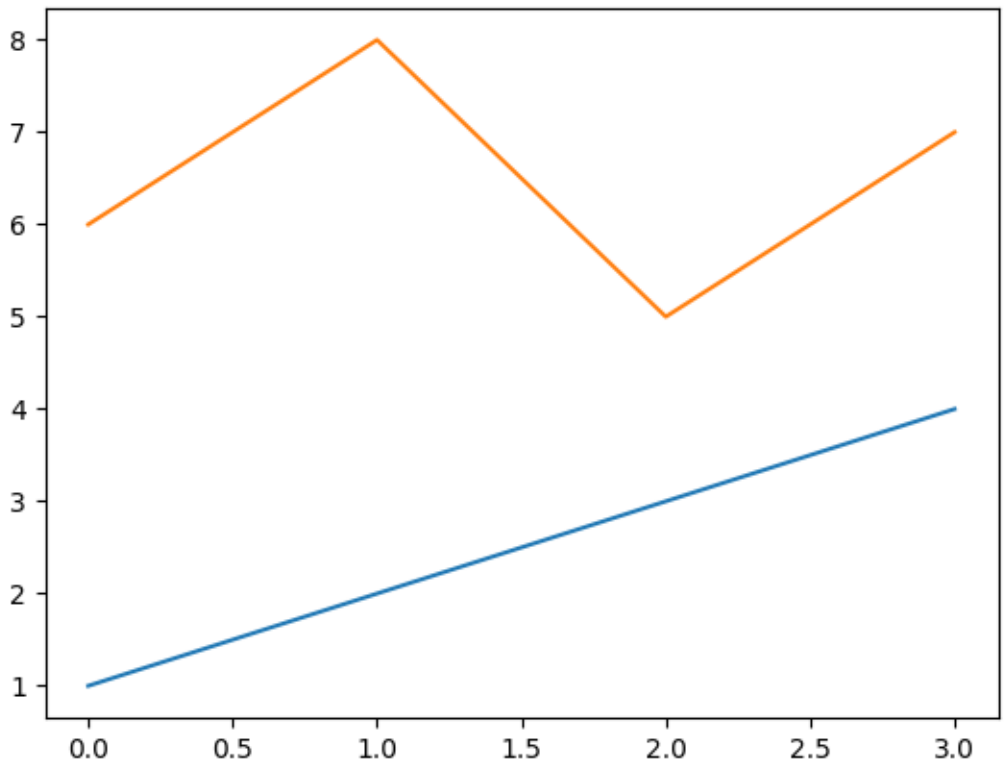
[11]:
```
# This example shows how to set the line style to dashed in a plot.

# 'ls' denotes the line style, here it is set to dashed

x=np.array([1,4,3,8])
plt.plot(x, ls='dashed')
plt.show()
```



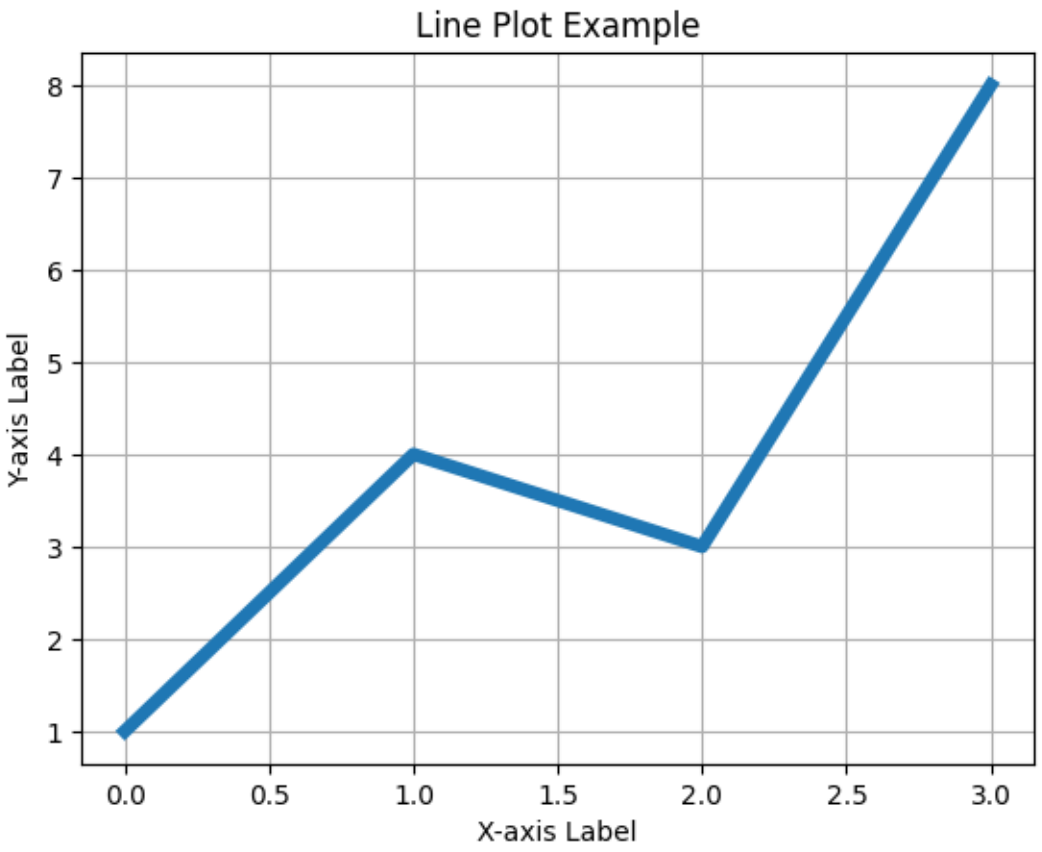
[12]:
```
# This example shows how to set the width of the line in a plot.

# 'linewidth' denotes the width of the line, here it is set to 5

x=np.array([1,4,3,8])
plt.plot(x, linewidth=5)
plt.show()
```



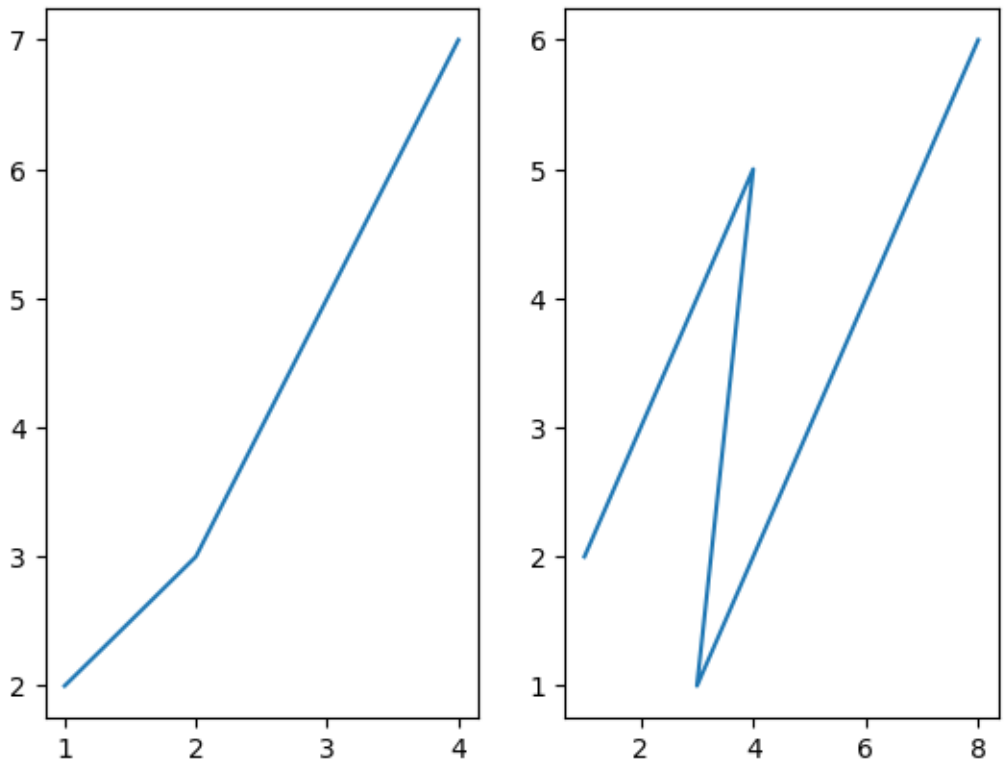
[13]:
```
# This example shows how to plot two different sets of points on the same plot.

# 'a' and 'b' are two different sets of points plotted on the same graph

a=np.array([1,2,3,4])
b=np.array([6,8,5,7])
plt.plot(a)
plt.plot(b)
plt.show()
```



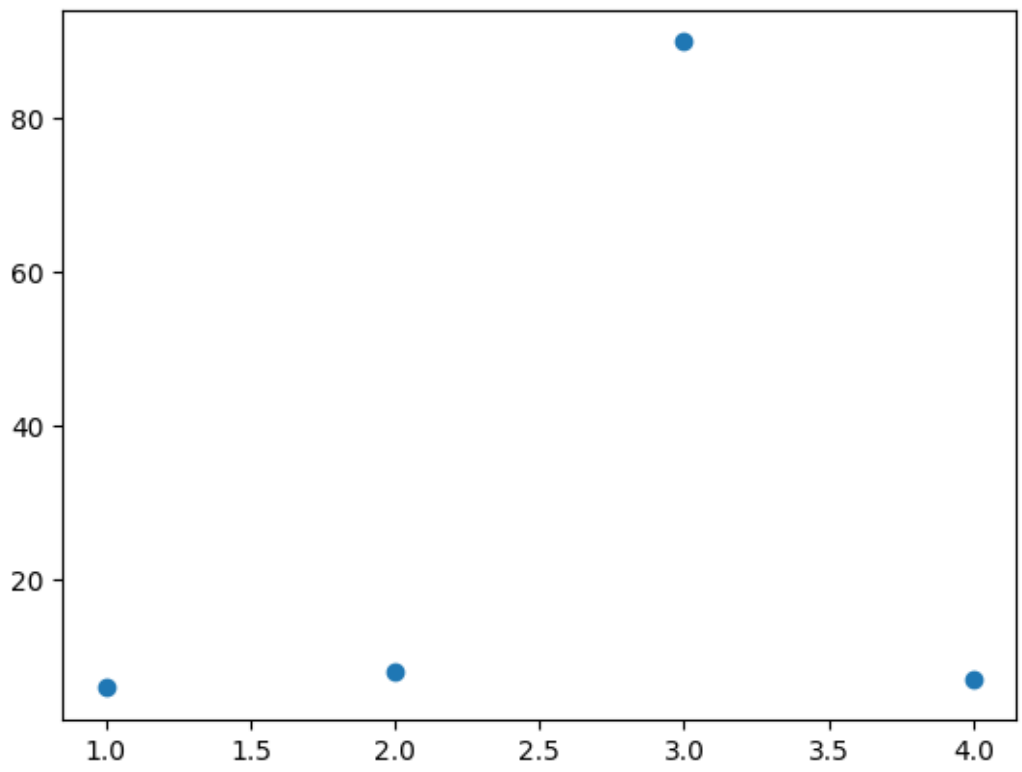
[14]:
```
# This example shows how to add a title,labels and Grid to the plot.

# Title is used to give a name to the plot
# labels are used to describe the x and y axes
# grid is used to display a grid on the plot

x=np.array([1,4,3,8])
plt.title('Line Plot Example')
plt.xlabel('X-axis Label')
plt.ylabel('Y-axis Label')
plt.grid(True)
plt.plot(x, linewidth=5)
plt.show()
```



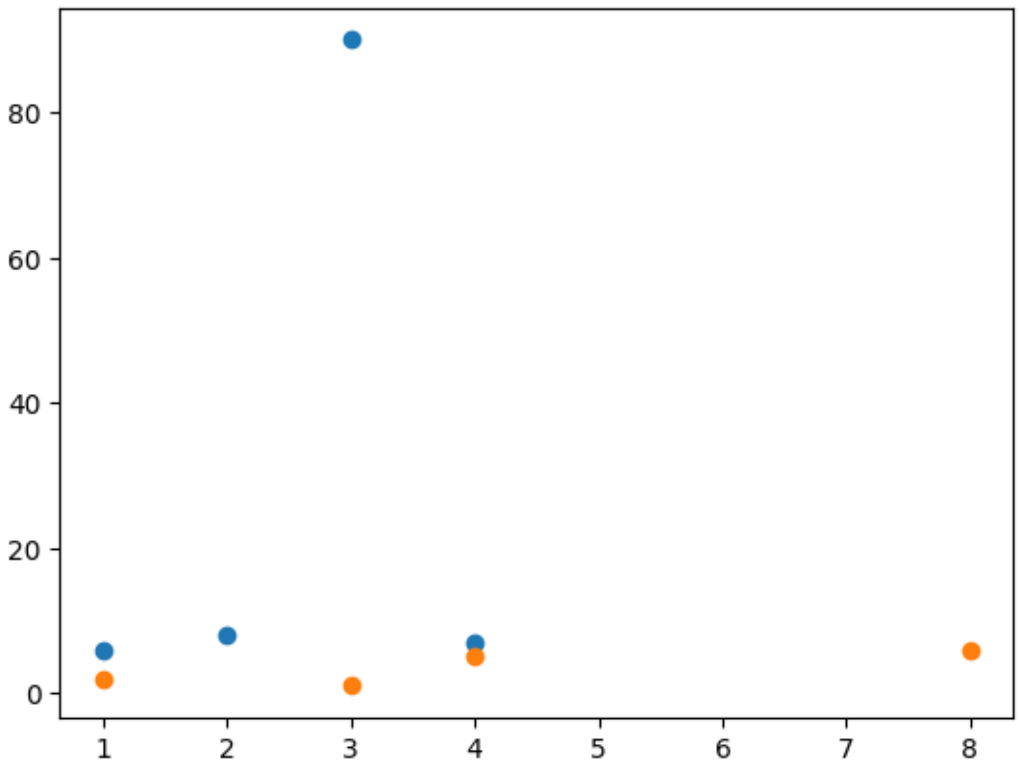
[15]:
```
# This example shows how to create subplots in a single figure.

#'subplot' is used to create multiple plots in a single figure

x=np.array([1,2,3,4])
y=np.array([2,3,5,7])
plt.subplot(1,2,1)
plt.plot(x, y)

x1=np.array([1,4,3,8])
y1=np.array([2,5,1,6])
plt.subplot(1,2,2)
plt.plot(x1, y1)

plt.show()
```



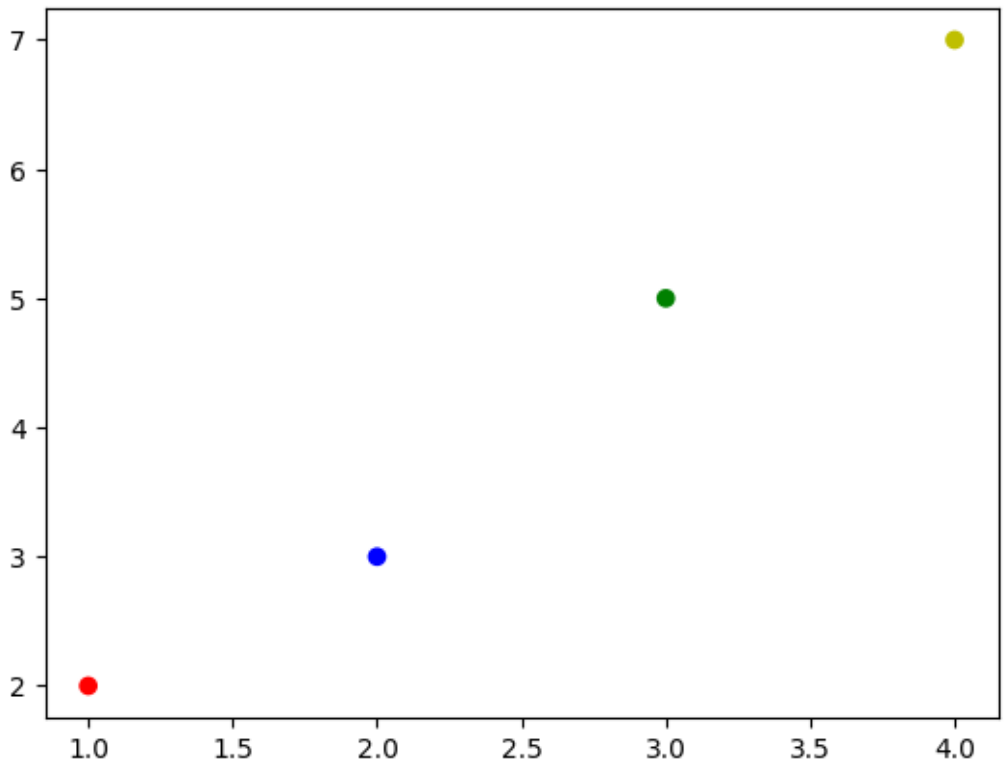
[16]:
```
# Scatter plot example
# A scatter plot is used to display values for typically two variables for a␣
 ↪set of data.

x=np.array([1,2,3,4])
y=np.array([6,8,90,7])
plt.scatter(x,y)
plt.show()
```



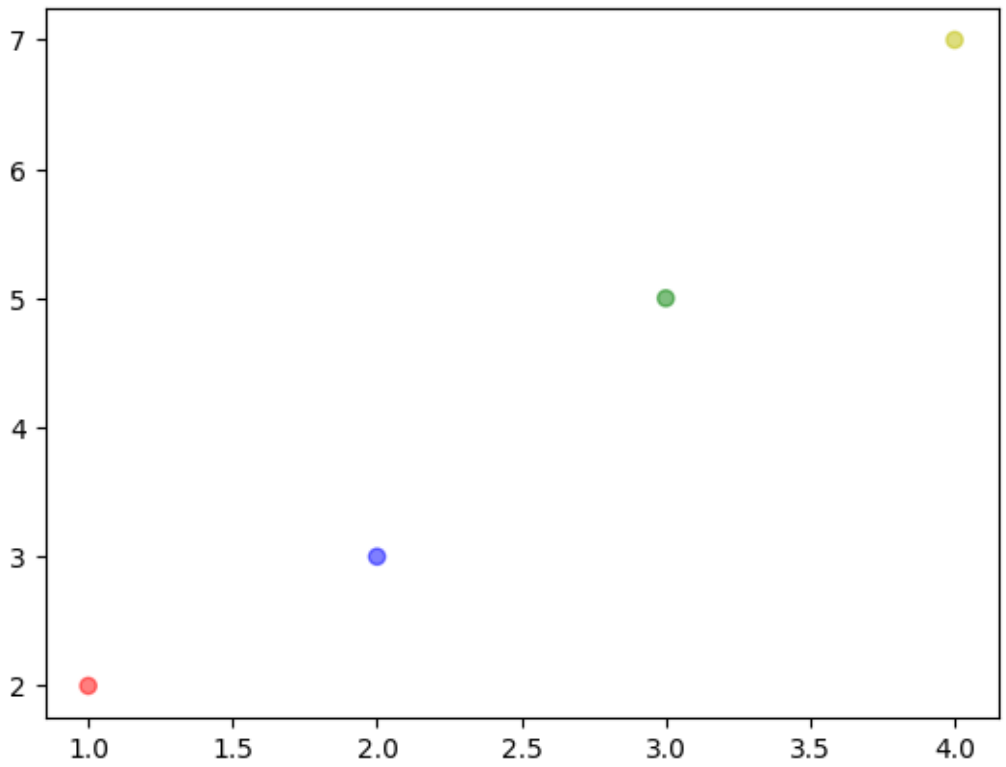
[17]:
```
# This example shows how to plot two different sets of points on the same␣
↪scatter plot.

# Scatter plot with two sets of points

x=np.array([1,2,3,4])
y=np.array([6,8,90,7])
plt.scatter(x,y)

x1=np.array([1,4,3,8])
y1=np.array([2,5,1,6])
plt.scatter(x1,y1)
plt.show()
```



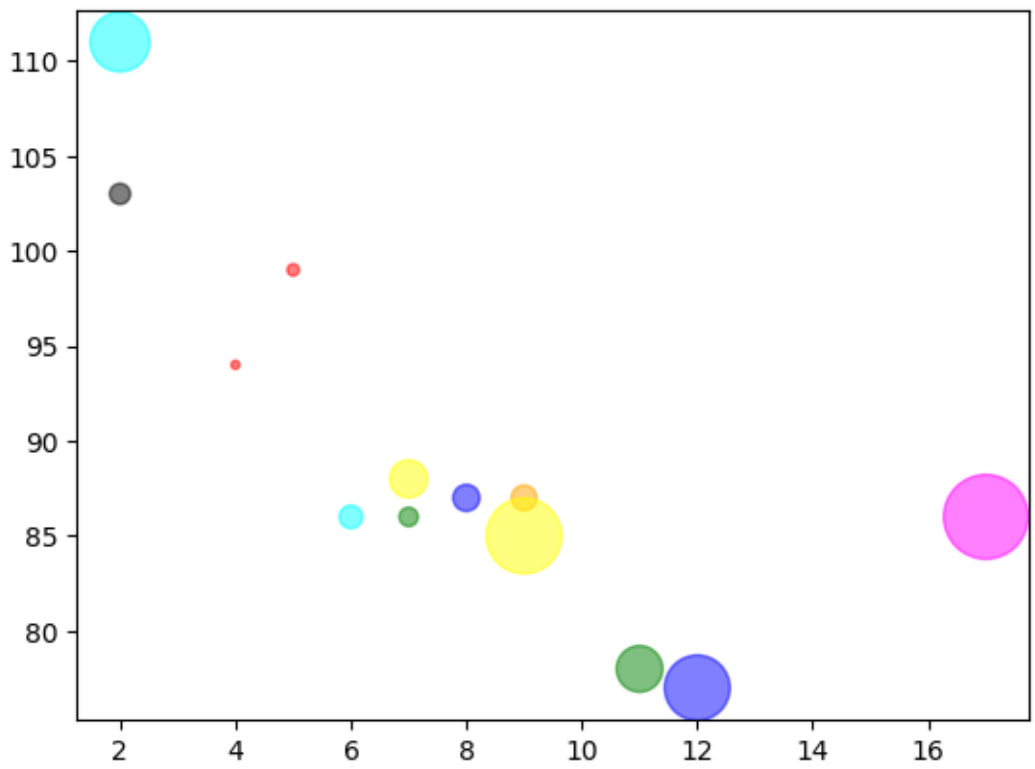
[18]:
```
# This example shows how to use different colors for the points in a scatter plot.

# Scatter plot with colors
# 'c' denotes the color of the points

a=np.array([1,2,3,4])
b=np.array([2,3,5,7])
c=np.array(['r','b','g','y'])
plt.scatter(a, b, c=c)
plt.show()
```



[19]:
```
# This example shows how to set the transparency of the points in a scatter␣
↪plot.

# Alpha blending for scatter plot
# 'alpha' denotes the transparency of the points

a=np.array([1,2,3,4])
b=np.array([2,3,5,7])
c=np.array(['r','b','g','y'])
plt.scatter(a, b, c=c, alpha=0.5)
plt.show()
```

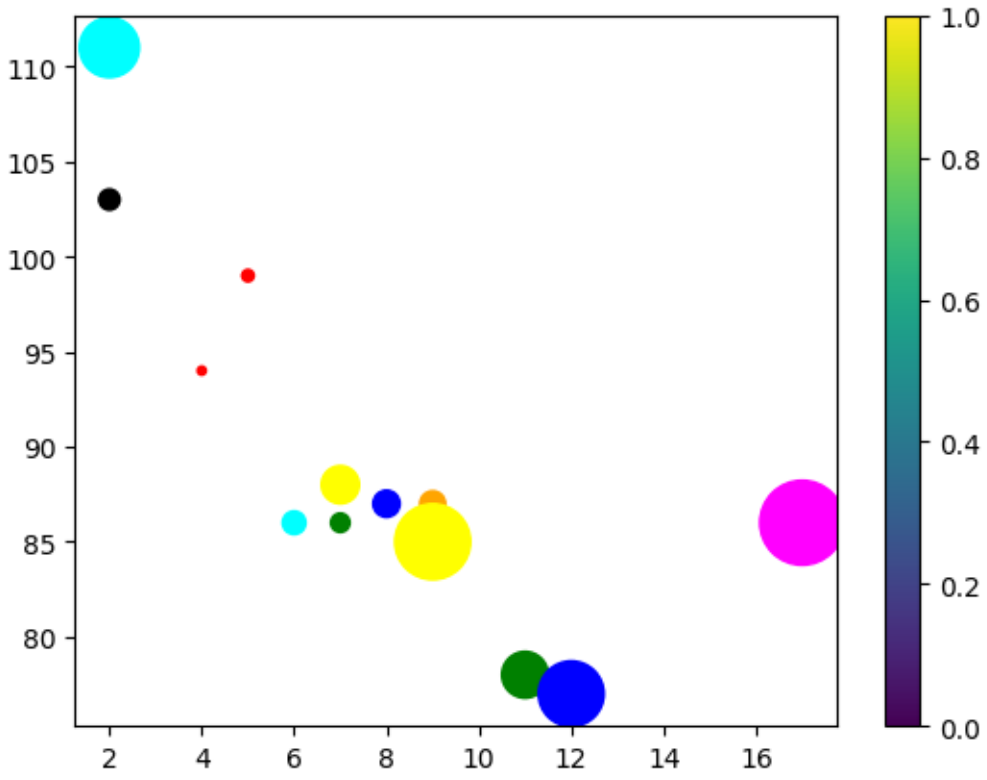


[20]:
```
# This example shows how to plot points with different sizes and colors in a␣
 ↪scatter plot.

# Scatter plot with sizes and colors
# 's' denotes the size of the points

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])
c = np.array(['red', 'green', 'blue', 'yellow', 'cyan', 'magenta', 'black',␣
 ↪'orange', 'red', 'green', 'blue', 'yellow', 'cyan'])
plt.scatter(x, y, s = sizes, c = c , alpha=0.5)
plt.show()
```



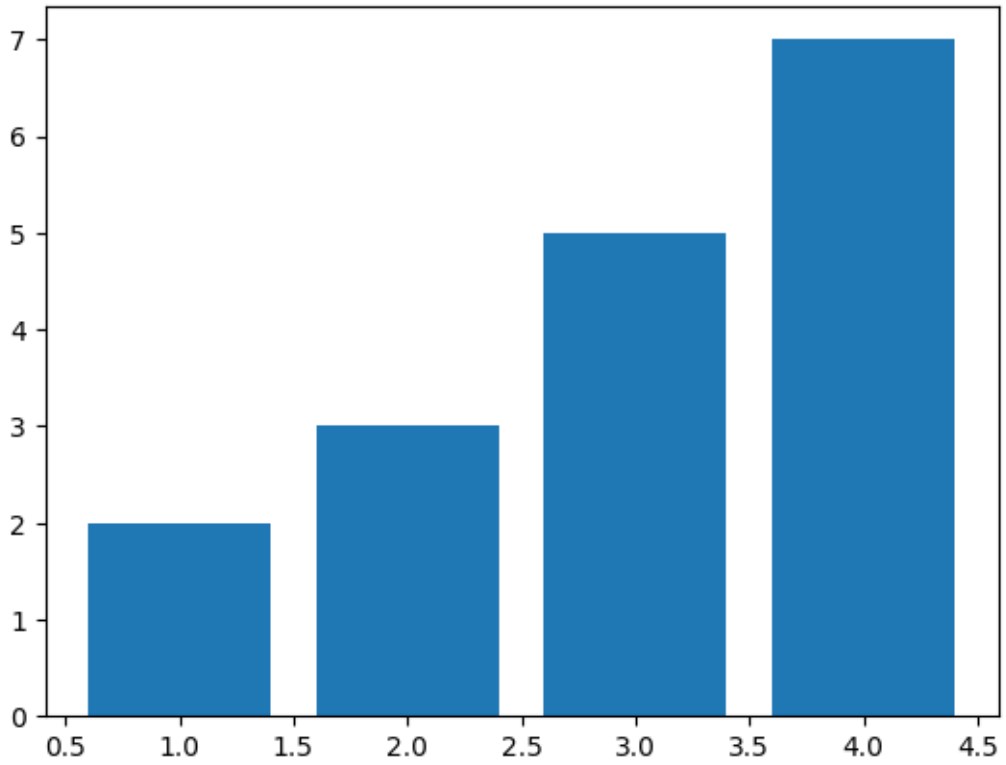
[21]:
```
# This example shows how to use a colormap in a scatter plot.

# 'cmap' denotes the colormap used for the points
# cmap is used to map the colors to the points based on their values

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])
c = np.array(['red', 'green', 'blue', 'yellow', 'cyan', 'magenta', 'black',␣
 ↪'orange', 'red', 'green', 'blue', 'yellow', 'cyan'])
plt.scatter(x, y, s = sizes, c = c, cmap='viridis')
plt.colorbar()
plt.show()
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_3748\2321600996.py:10: UserWarning: No
data for colormapping provided via 'c'. Parameters 'cmap' will be ignored
  plt.scatter(x, y, s = sizes, c = c, cmap='viridis')
```

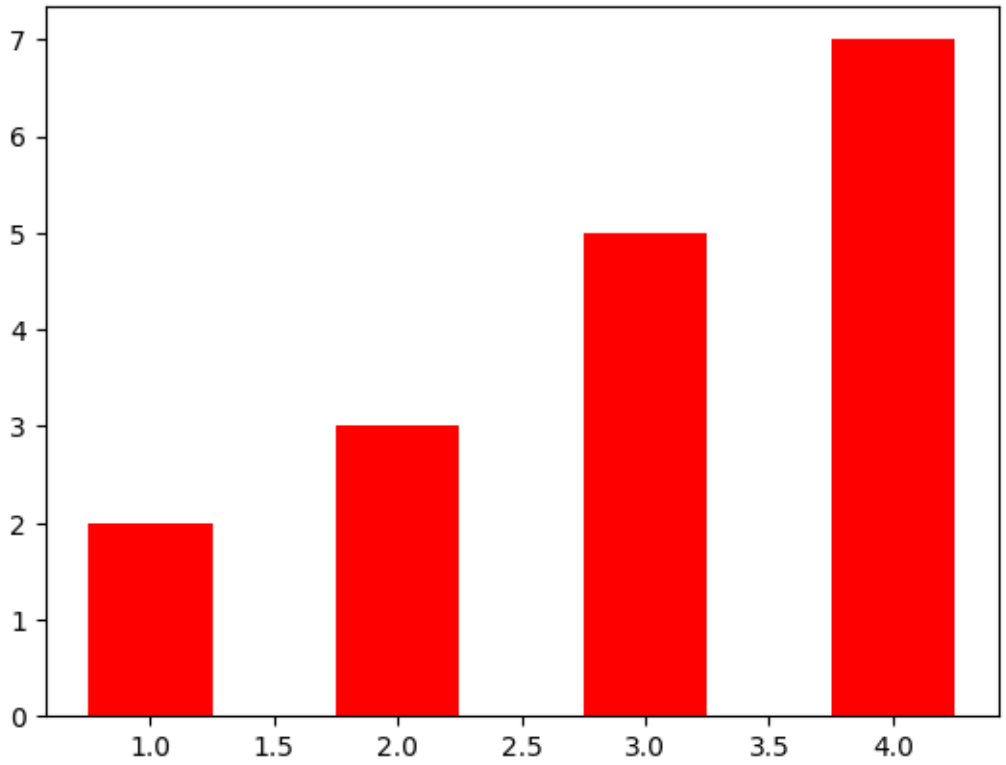


0.1 Bar Plot

[22]:
```
# Bar Plot
# A bar plot is used to display categorical data with rectangular bars.

x=np.array([1,2,3,4])
y=np.array([2,3,5,7])
plt.bar(x, y)
plt.show()
```



[23]:
```
# This example shows how to create a bar plot with a specific color.

# Bar Plot with color

x=np.array([1,2,3,4])
y=np.array([2,3,5,7])
plt.bar(x, y,color='red', width=0.5,)
plt.show()
```



0.2 Hist Plot

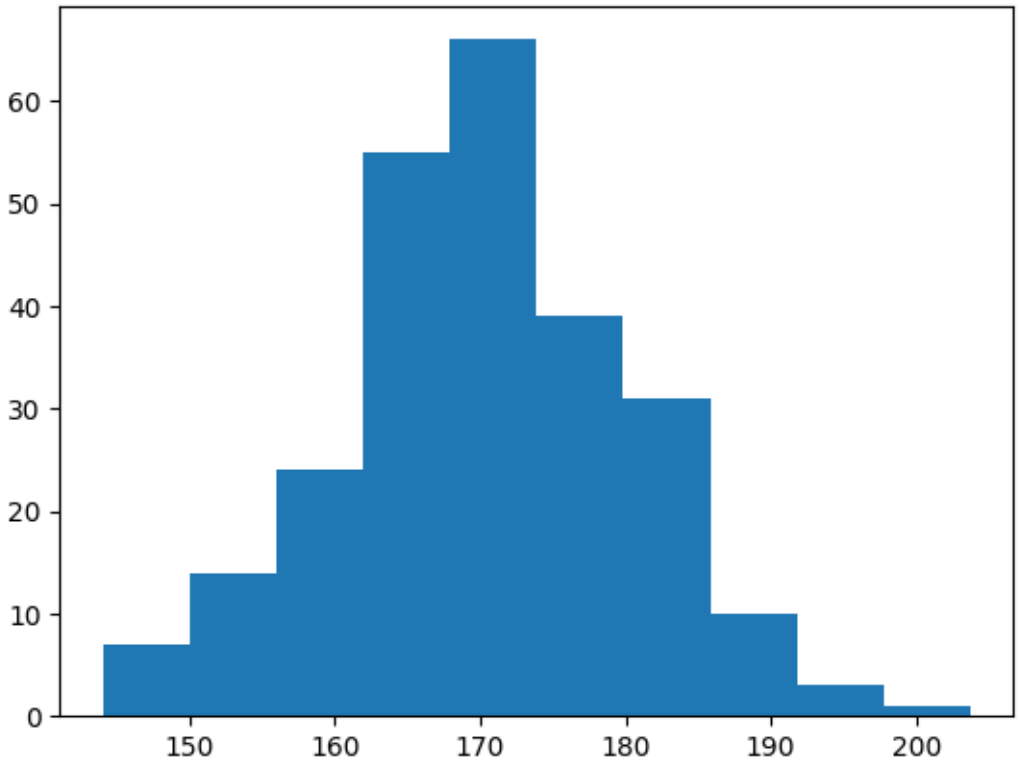
[24]:
```
# Hist Plot
# A histogram is used to represent the distribution of numerical data by␣
 ↪dividing the data into bins.

x=np.random.normal(170, 10, 250)
x
```

[24]:
```
array([167.51074121, 173.50591421, 162.72381222, 171.61637559,
       177.90174069, 173.64680238, 175.07722859, 170.59334645,
       162.03186832, 175.35216078, 184.14487574, 177.4263598 ,
       167.11390415, 163.27837508, 164.41320673, 170.15326745,
       159.22938344, 159.80425999, 170.82911686, 157.54627739,
       186.982758  , 177.56770079, 173.17867942, 166.19206834,
       182.77323774, 180.9953627 , 186.57243443, 165.05390603,
       163.32914634, 174.94839586, 167.57177998, 169.15350374,
       163.35043339, 173.64057667, 165.69242934, 156.9661402 ,
       176.42401442, 166.0736083 , 168.5682078 , 145.74340735,
       176.92663069, 185.16309483, 176.4839425 , 164.59120384,
       181.79207271, 175.50082301, 163.50101398, 155.65705402,
       175.28236623, 168.55565548, 177.16672028, 168.16621402,
```

160.77591992, 160.96885223, 148.31927722, 167.16241409,
160.62240101, 181.13224744, 168.50387768, 155.83347998,
165.51036354, 169.75982663, 171.91897612, 175.80162866,
159.12888904, 173.30927544, 176.10009255, 155.53189278,
164.0201893 , 172.39993742, 181.64169024, 182.17563614,
184.78126362, 165.49026362, 161.18977204, 186.71649992,
174.34099186, 167.20089938, 163.00008258, 165.05130018,
174.13754187, 163.53734127, 165.92180193, 195.63620704,
177.61940522, 182.71196162, 166.74223366, 177.2777118 ,
171.02636636, 170.60440327, 168.90815957, 152.56184251,
171.23711366, 180.55159873, 174.66323105, 166.05507173,
150.18583599, 170.83537961, 160.61836893, 163.20170052,
157.98671929, 169.34343369, 161.17933559, 182.13498854,
168.45589156, 161.73664566, 172.92705657, 169.5729981 ,
165.90248238, 184.41176905, 179.39315169, 174.35679126,
148.24734211, 150.0404803 , 170.06750661, 159.87953218,
173.65881461, 170.06417952, 174.87666094, 168.05383585,
167.58292364, 172.24179137, 153.55970564, 174.86895832,
166.62032339, 174.67681278, 159.32824263, 172.41301162,
171.3239349 , 189.29423117, 178.8331535 , 153.78776315,
170.07496801, 168.90307947, 165.76538923, 203.70209724,
148.47629648, 175.34003725, 181.51815882, 148.72299201,
181.66753138, 197.55687721, 181.26504351, 169.95238064,
173.22557098, 185.91737955, 164.36051222, 172.13880496,
170.82049769, 176.338508 , 166.998995 , 168.80989185,
181.35994793, 187.33503931, 183.94392934, 167.23719329,
158.43022721, 154.43306107, 176.18338096, 184.11545933,
161.18720904, 169.97142468, 185.28859916, 182.67426643,
169.48206823, 151.59370699, 171.64238622, 176.58090813,
162.82825677, 169.31408299, 191.64799338, 188.20887336,
167.50599232, 177.1750306 , 160.65185841, 182.40193631,
176.86248127, 173.69163366, 170.63355993, 171.35307469,
179.14358026, 179.76085408, 176.45038419, 167.41287345,
152.77482043, 166.81880083, 162.17223614, 156.80316757,
159.18248891, 164.00334429, 178.59029318, 173.49140082,
167.99651474, 164.83127103, 180.21929348, 171.98608922,
173.34414227, 154.59027759, 164.13699399, 181.98877507,
172.53783565, 155.35356379, 182.74395386, 191.92088427,
173.55042172, 162.03404021, 166.17869411, 171.52254691,
144.09282612, 164.75628044, 153.24981065, 176.90014602,
162.84871286, 163.18246783, 181.59585114, 167.30251245,
180.85185443, 172.2440549 , 166.15018543, 164.10577738,
184.92264461, 167.13237657, 160.78795021, 187.40003722,
173.64027844, 187.89031582, 158.11653632, 177.60246809,
173.09008278, 152.88938529, 174.22148318, 172.22069594,
181.52105646, 166.56534183, 162.36103785, 175.76896241,
164.34643439, 172.0107475 , 170.53974044, 165.85085139,

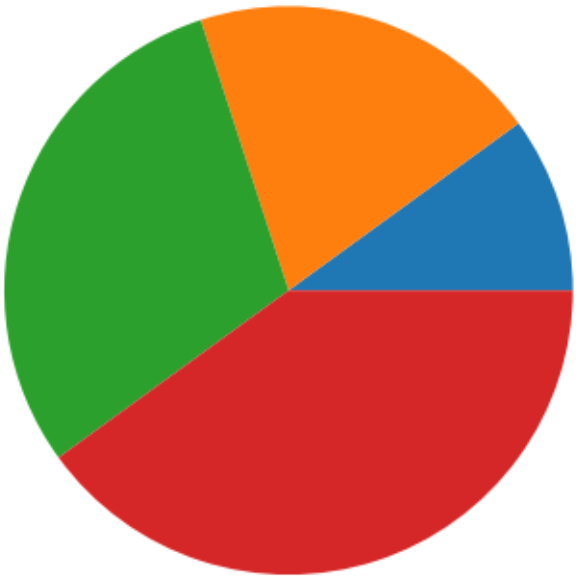
```
172.70520232, 168.72909994, 181.01219321, 160.38269964,
161.075961  , 169.39252292, 171.52367384, 168.15460823,
169.57843977, 181.53992851])
```

[25]:
```
plt.hist(x)
plt.show()
```

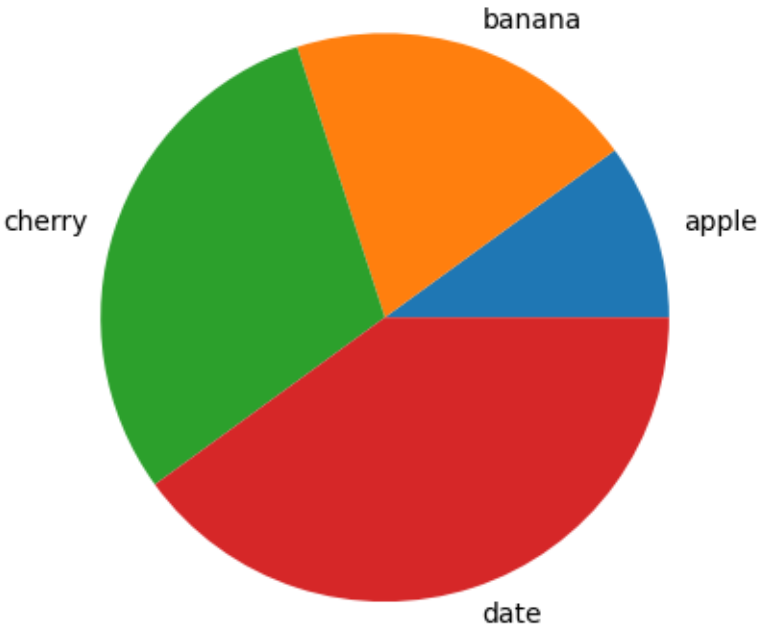


0.3 Pie Plot

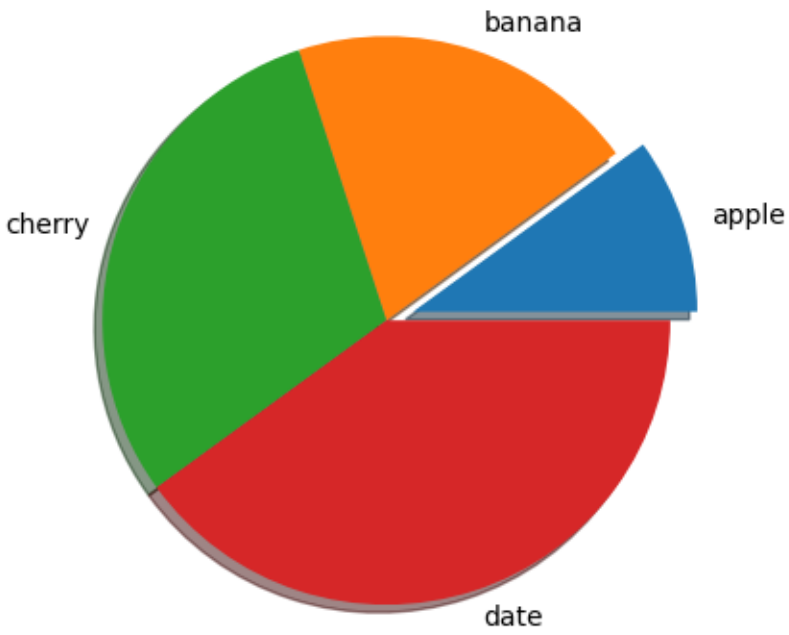
[26]:
```
# Pie Plot
# A pie chart is a circular statistical graphic divided into slices to␣
 ↪illustrate numerical proportions.
# Each slice of the pie represents a category's contribution to the total.

x=np.array([1,2,3,4])
plt.pie(x)
plt.show()
```



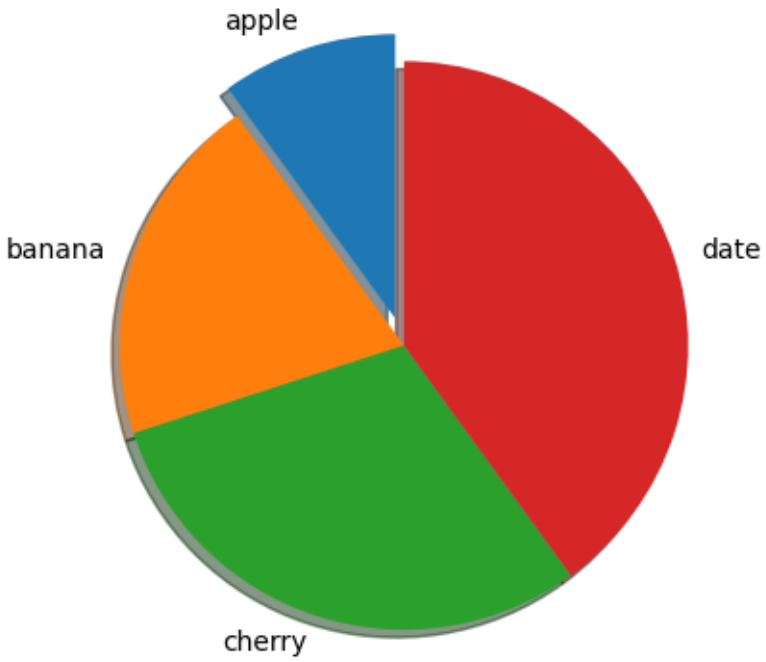
[27]:
```
# This example shows how to add labels to the pie chart.

# Pie Plot with labels
# This code creates a pie chart with labels for each segment which helps in␣
 ↪identifying the categories represented by each slice.

x=np.array([1,2,3,4])
labels=["apple", "banana", "cherry", "date"]
plt.pie(x, labels=labels)
plt.show()
```



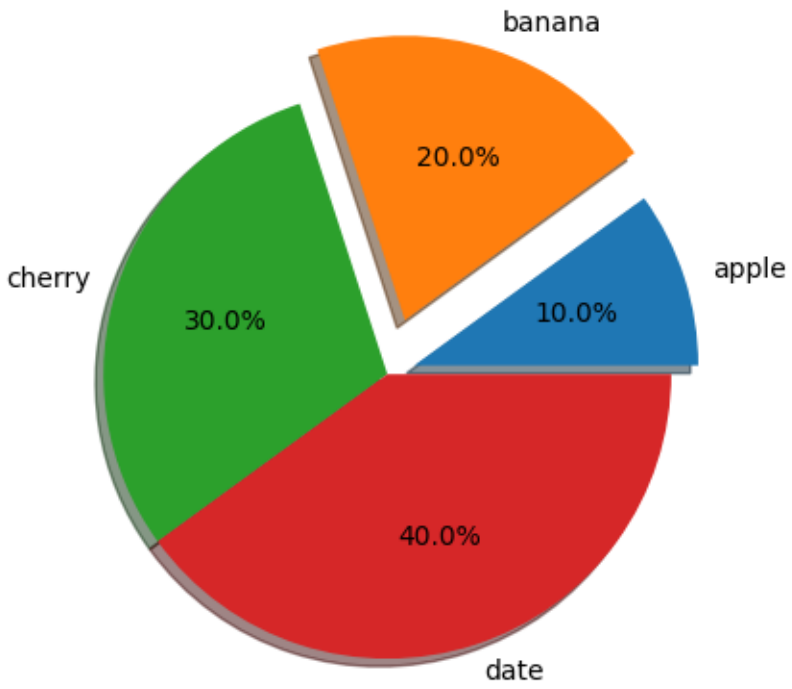
[28]:
```
# This example shows how to create a pie chart with an exploded slice and␣
 ↪shadow effect.

# Pie Plot with explode and shadow
# 'explode' is used to separate the slices of the pie chart
# 'shadow' adds a shadow effect to the pie chart

x=np.array([1,2,3,4])
labels=["apple", "banana", "cherry", "date"]
myexplode = (0.1, 0, 0, 0)
plt.pie(x, labels=labels, explode=myexplode, shadow=True)
plt.show()
```



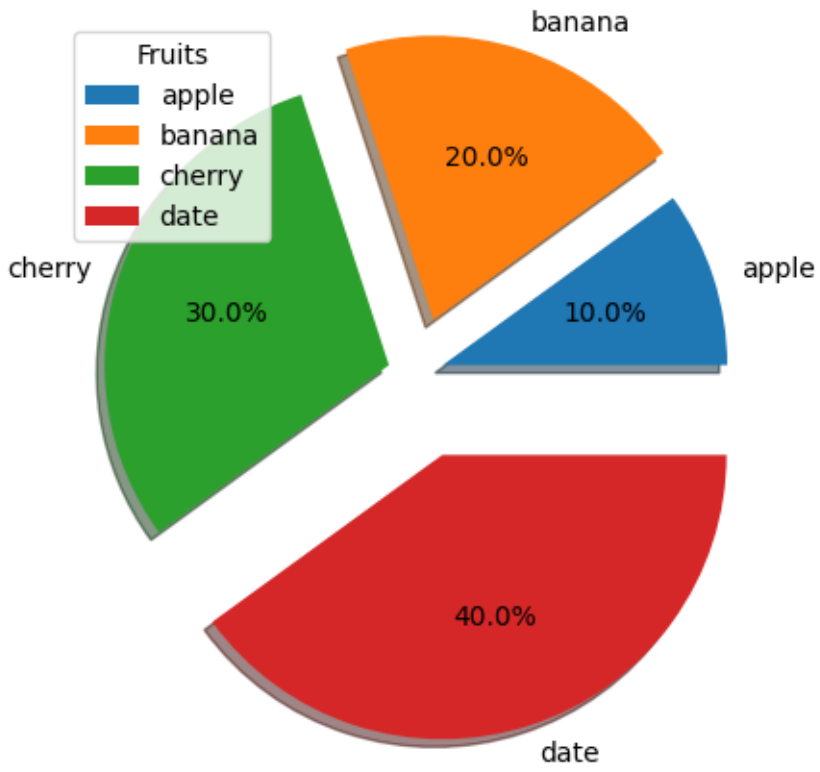
[29]:
```
# This example shows how to set the start angle to the pie chart.

# Start angle in Pie Plot
# 'startangle' rotates the start of the pie chart

x=np.array([1,2,3,4])
labels=["apple", "banana", "cherry", "date"]
myexplode = (0.1, 0, 0, 0)
plt.pie(x, labels=labels, explode=myexplode,startangle=90, shadow=True)
plt.show()
```



[30]:
```
# This example shows how to display percentages in the pie chart.

# Autopct in Pie Plot
# 'autopct' is used to display the percentage of each slice
# '%1.1f%%' formats the percentage to one decimal place

x=np.array([1,2,3,4])
labels=["apple", "banana", "cherry", "date"]
myexplode = (0.1, 0.2, 0, 0)
plt.pie(x, labels=labels, explode=myexplode, shadow=True,autopct='%1.1f%%')
plt.show()
```



[31]:
```
# This example shows how to add a legend to the pie chart.

# Legend in Pie Plot
# 'legend' is used to add a legend to the pie chart
# 'title' sets the title of the legend
# loc='upper left' positions the legend in the upper left corner

x=np.array([1,2,3,4])
labels=["apple", "banana", "cherry", "date"]
myexplode = (0.1, 0.2, 0.1, 0.3)
plt.pie(x,
        labels=labels,
        explode=myexplode,
        shadow=True,
        autopct='%1.1f%%',
        )
plt.legend(title="Fruits", loc="upper left")
plt.show()
```


Fruits
apple
banana
cherry
date
banana
20.0%
apple
10.0%
cherry
30.0%
40.0%
date

[ ]: