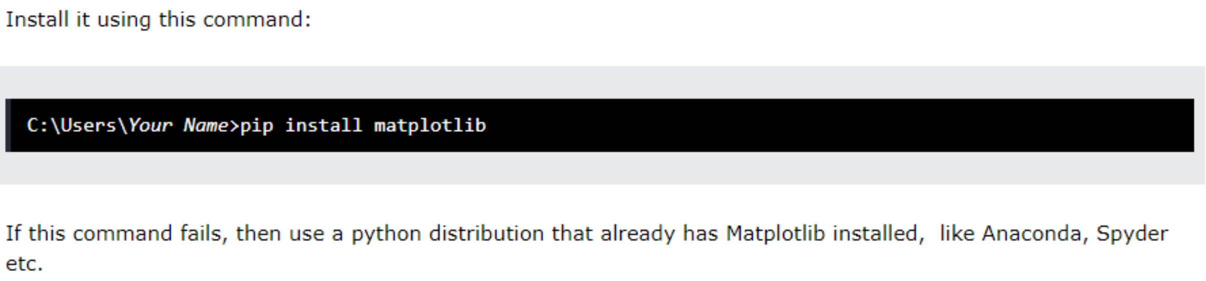
Matplotlib is a low level graph plotting library in python that serves as a visualization utility.

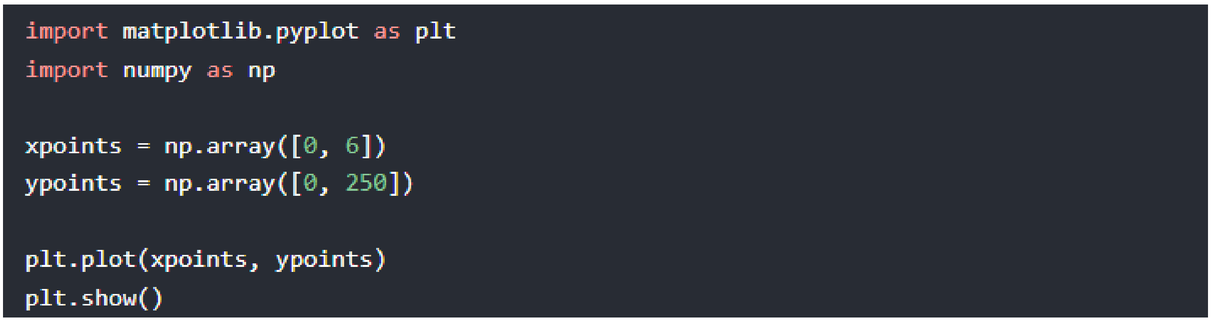


Install it using this command:

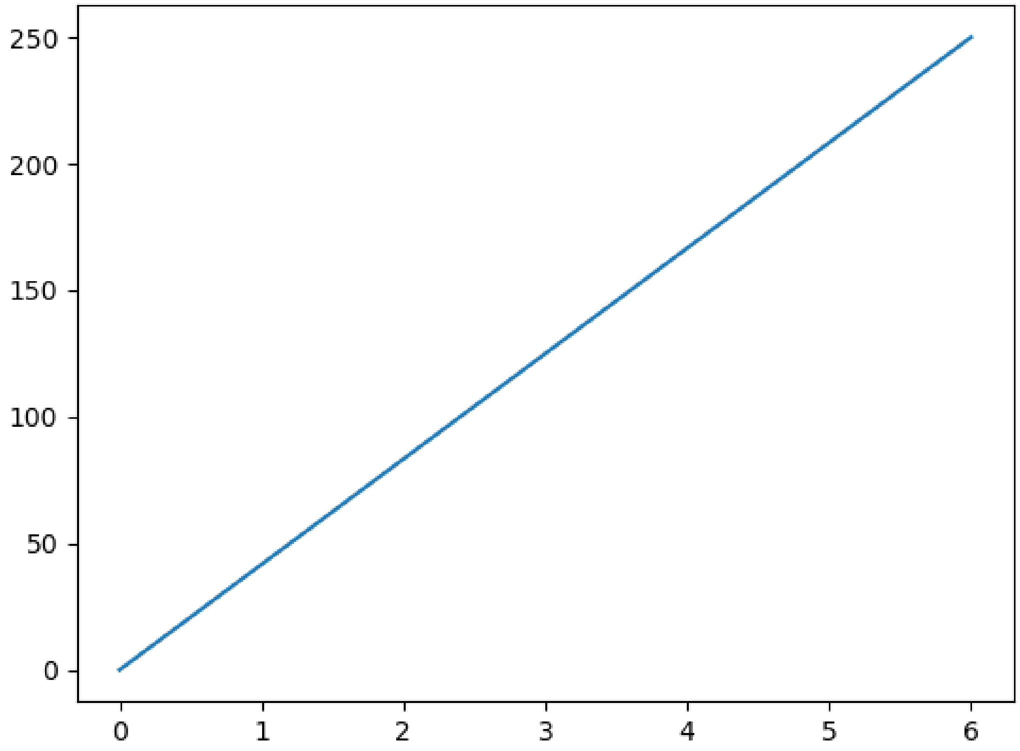
```
C:\Users\Your Name>pip install matplotlib
```

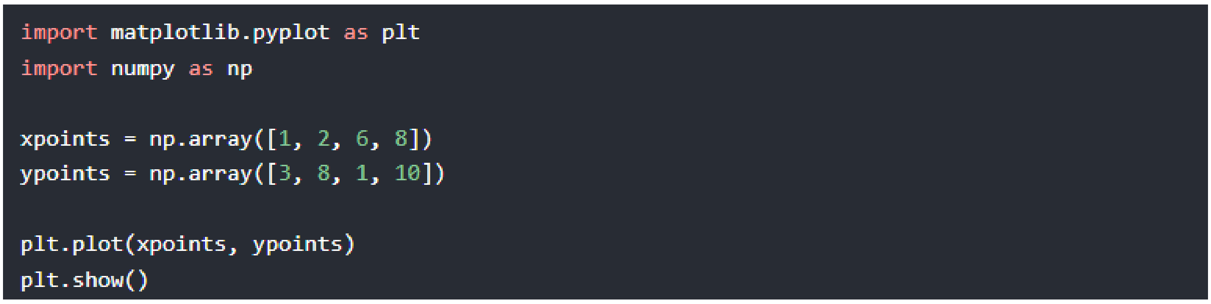
If this command fails, then use a python distribution that already has Matplotlib installed, like Anaconda, Spyder etc.

Pyplot

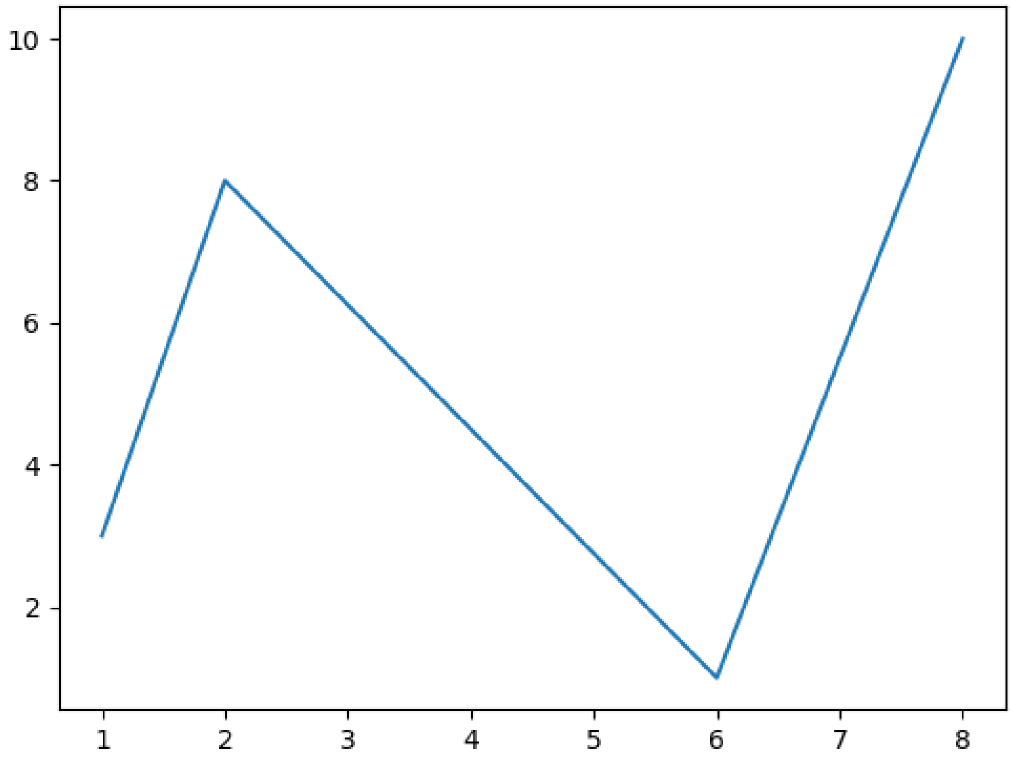


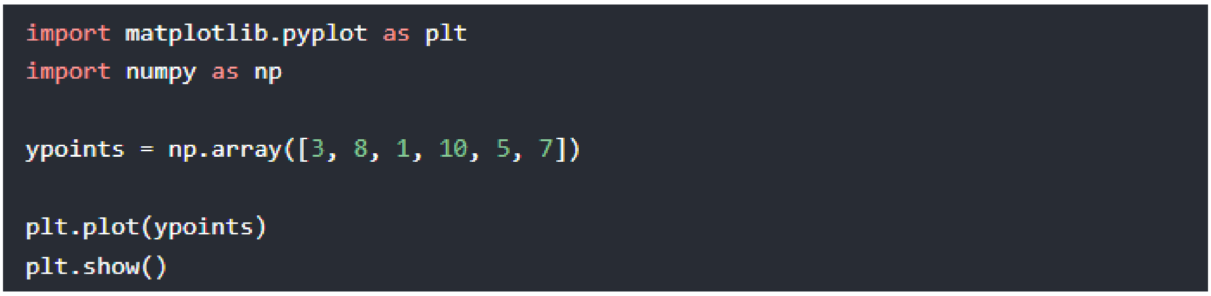
```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([0, 6])
ypoints = np.array([0, 250])

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



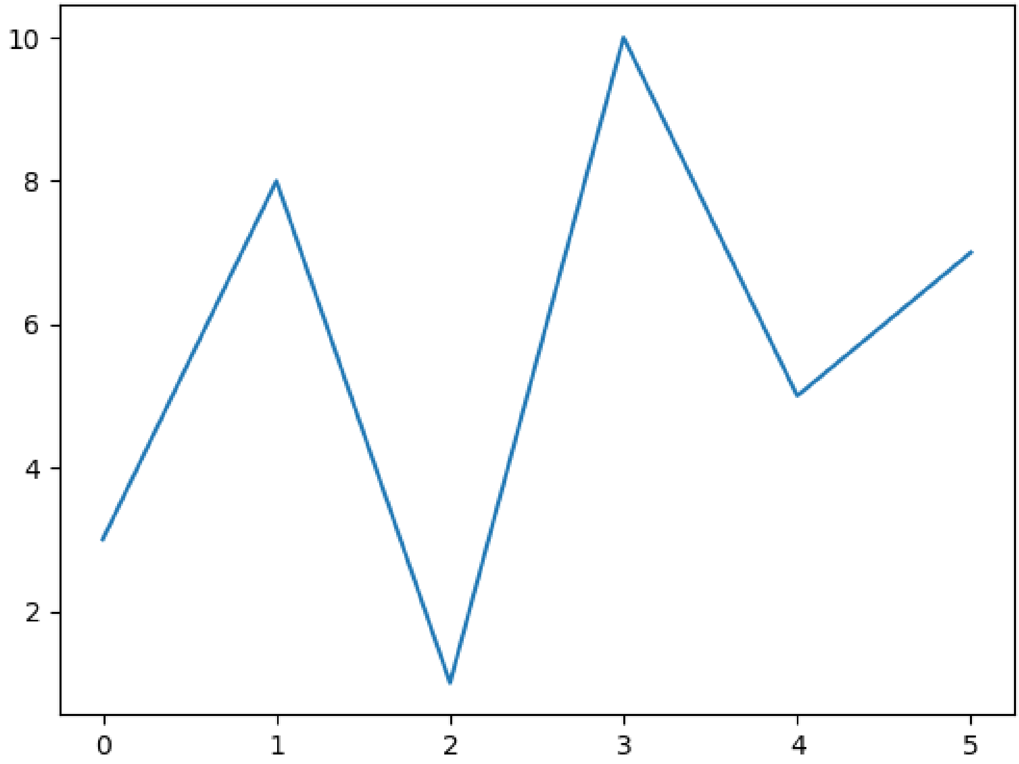


```
import matplotlib.pyplot as plt
import numpy as np

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

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

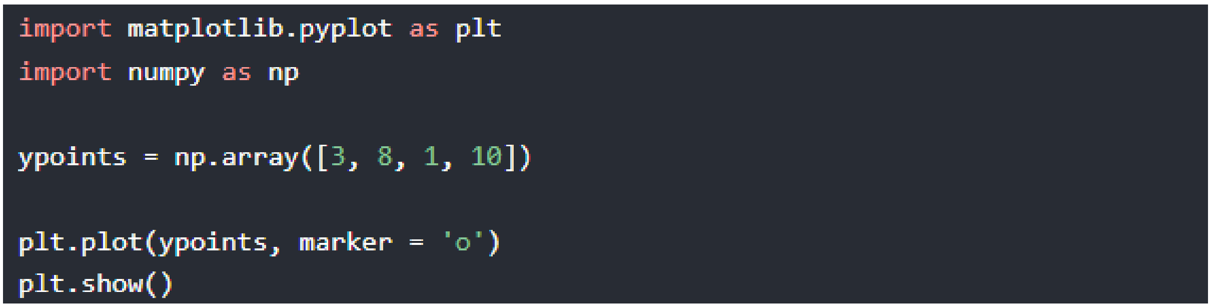




```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10, 5, 7])

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



Markers

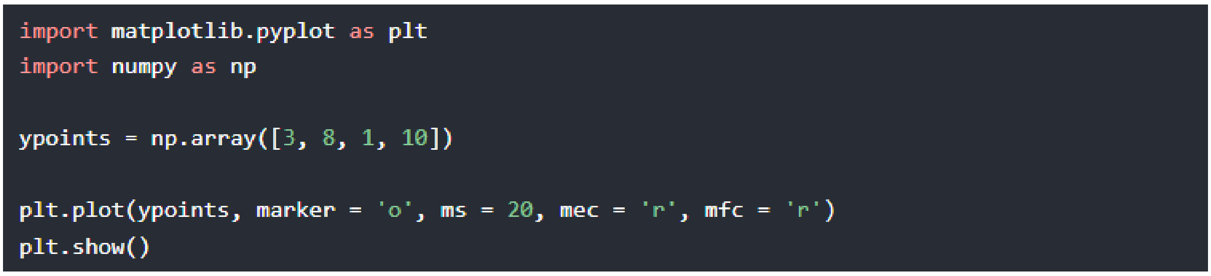


```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

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

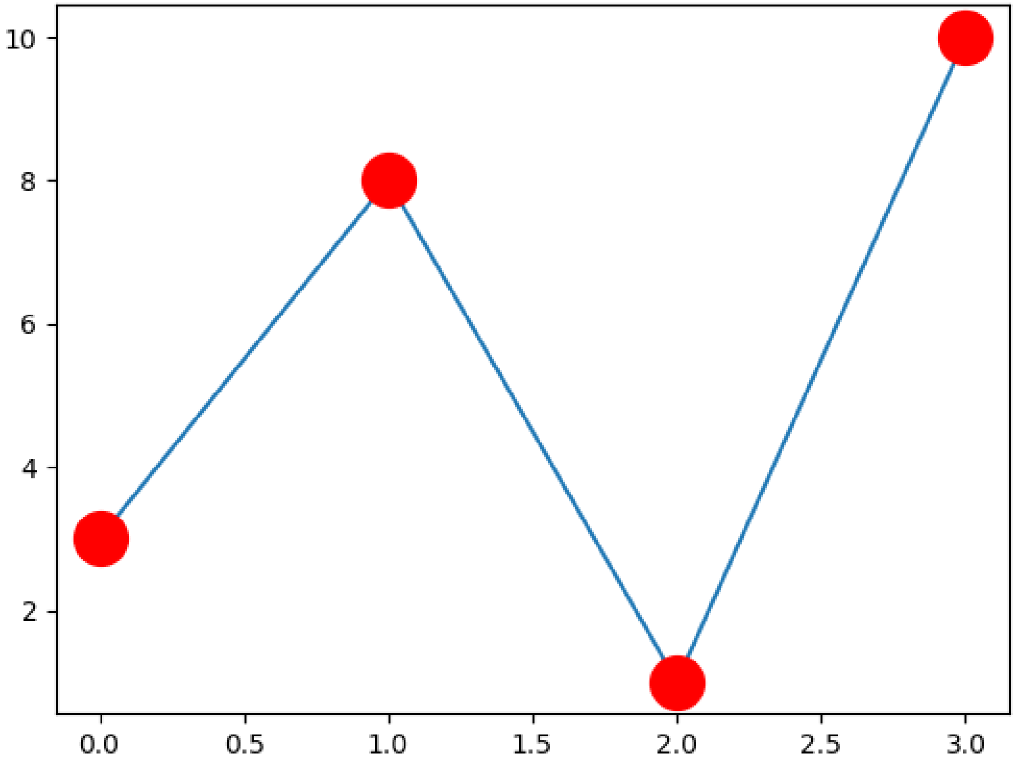
```
plt.plot(ypoints, marker = '*')
```

https://www.w3schools.com/python/matplotlib_markers.asp

You can use the keyword argument `markersize` or the shorter `ms` to set the size of the markers, `markeredgecolor` or the shorter `mec` to set the color of the *edge* of the markers, `markerfacecolor` or the shorter `mfc` to set the color inside the edge of the markers:

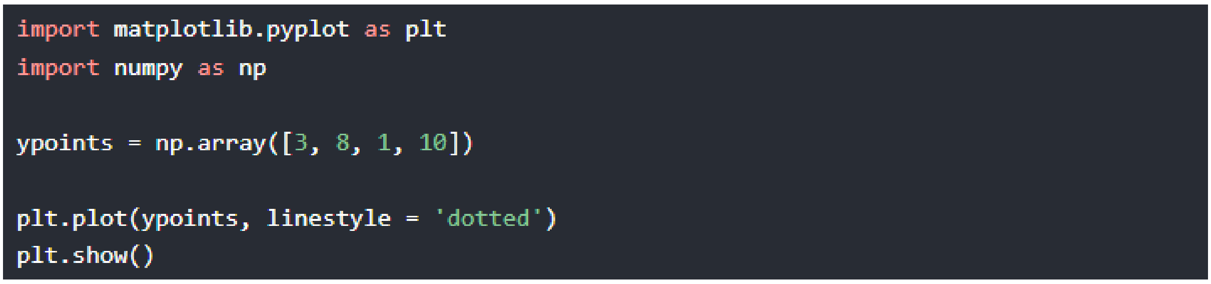


```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

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

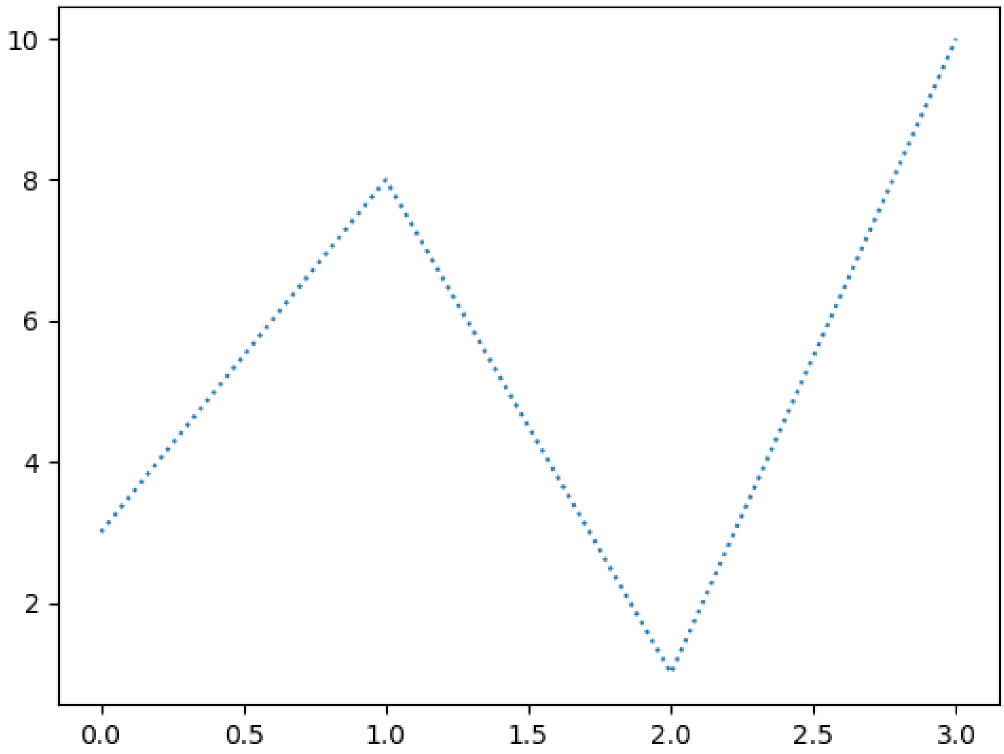


```
plt.plot(ypoints, marker = 'o', ms = 20, mec = '#4CAF50', mfc =
'#4CAF50')
```

Line

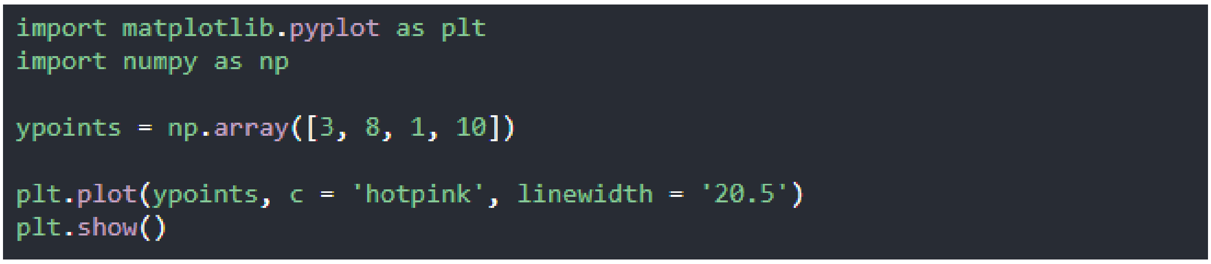


```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, linestyle = 'dotted')
plt.show()
```

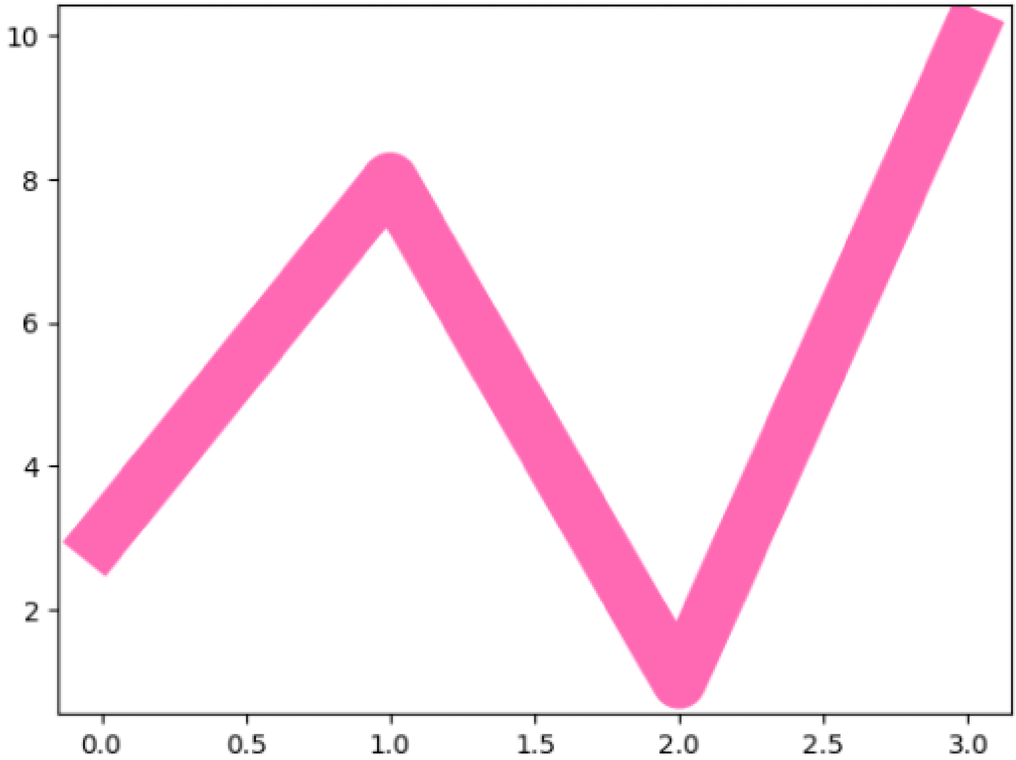


https://www.w3schools.com/python/matplotlib_line.asp

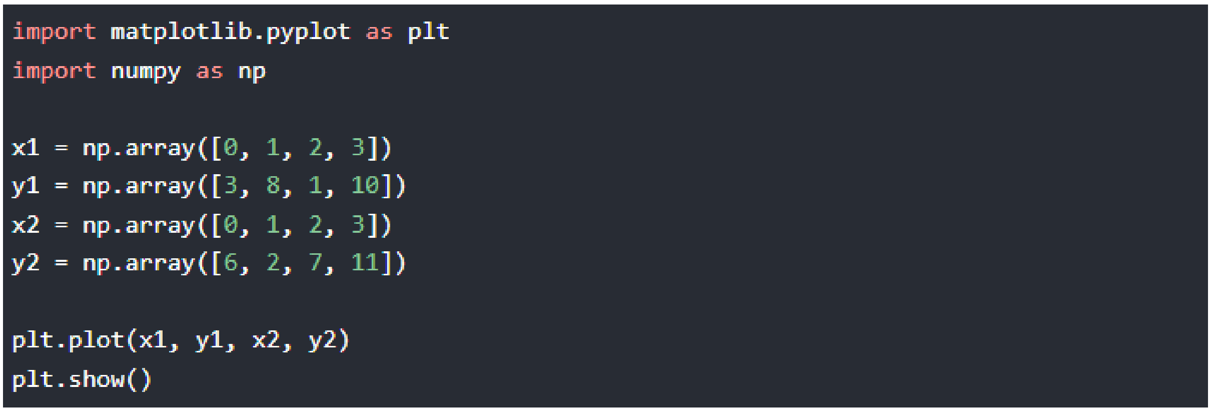
You can use the keyword argument `color` or the shorter `c` to set the color of the line, `linewidth` or the shorter `lw` to change the width of the line.



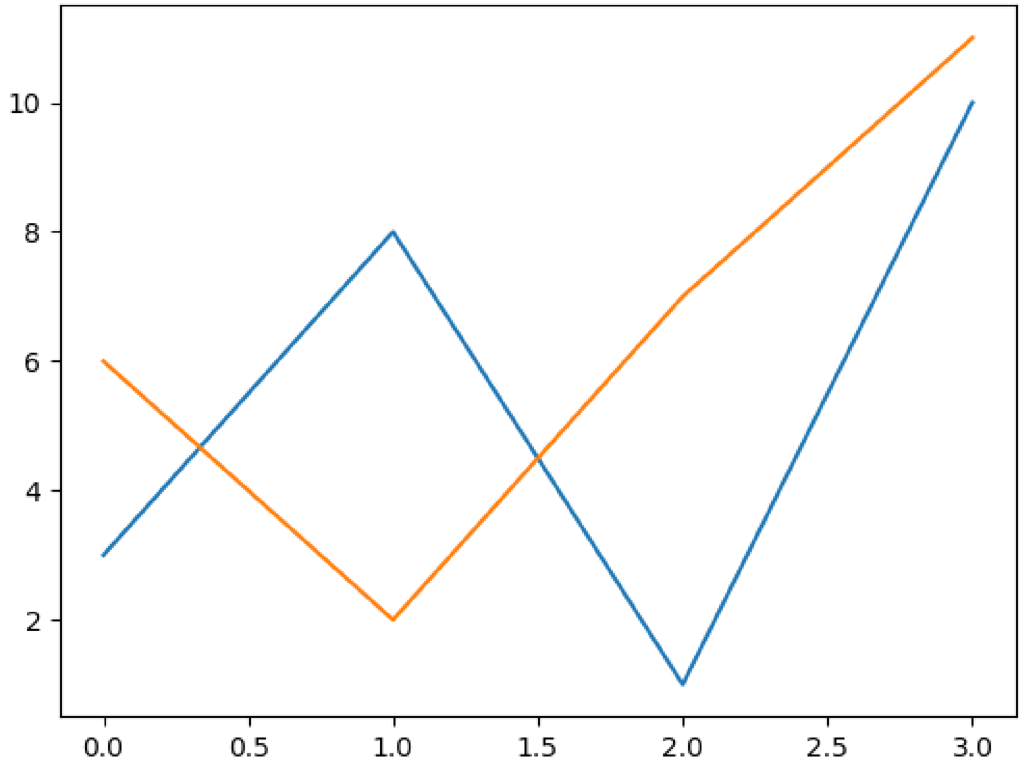
```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, c = 'hotpink', linewidth = '20.5')
plt.show()
```



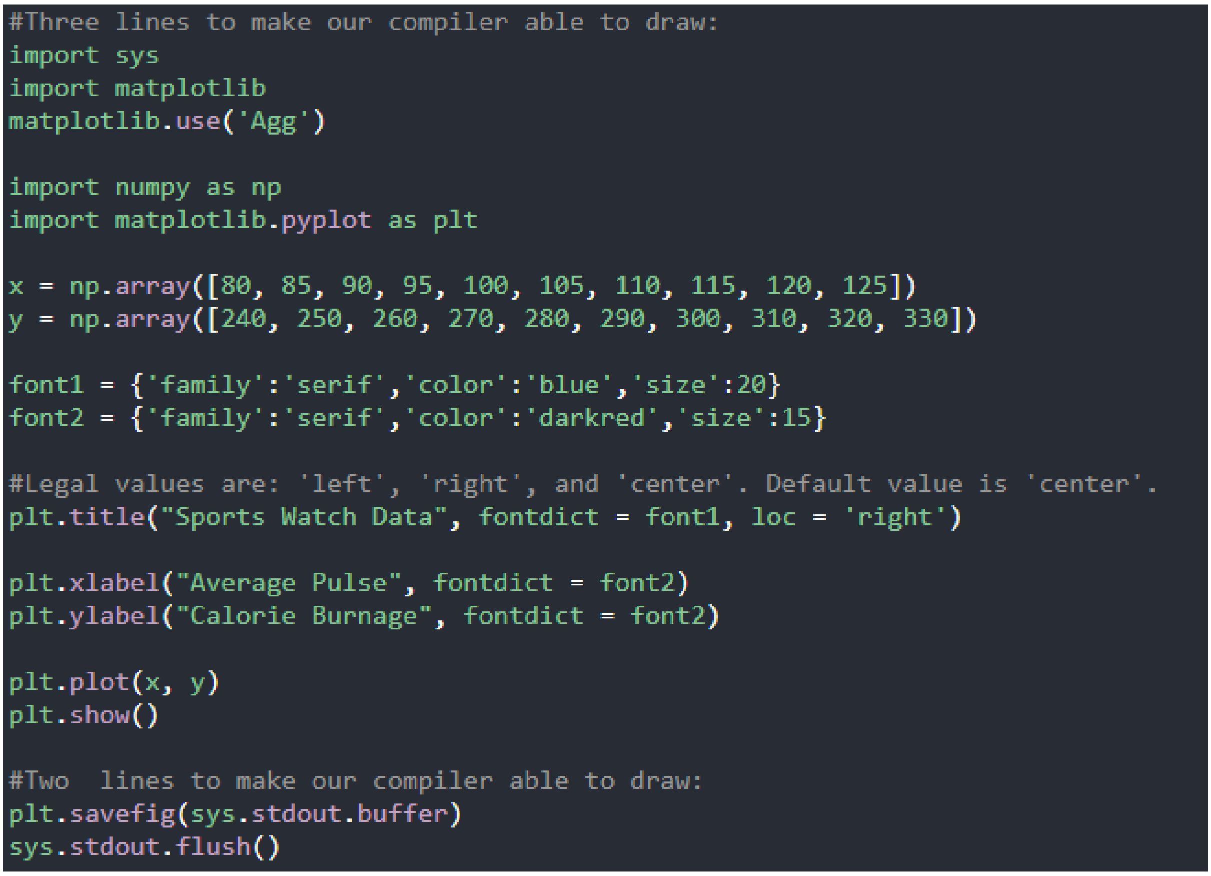
Multiple Lines



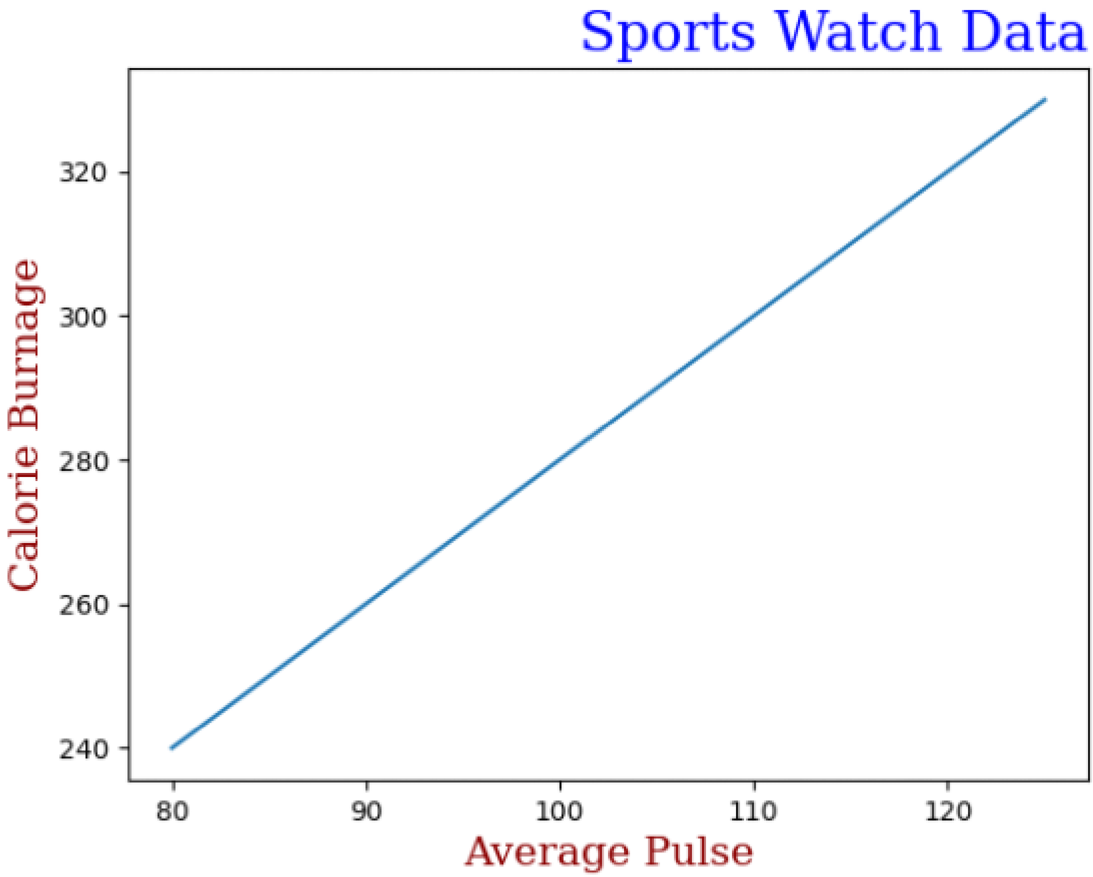
```
import matplotlib.pyplot as plt
import numpy as np

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()
```



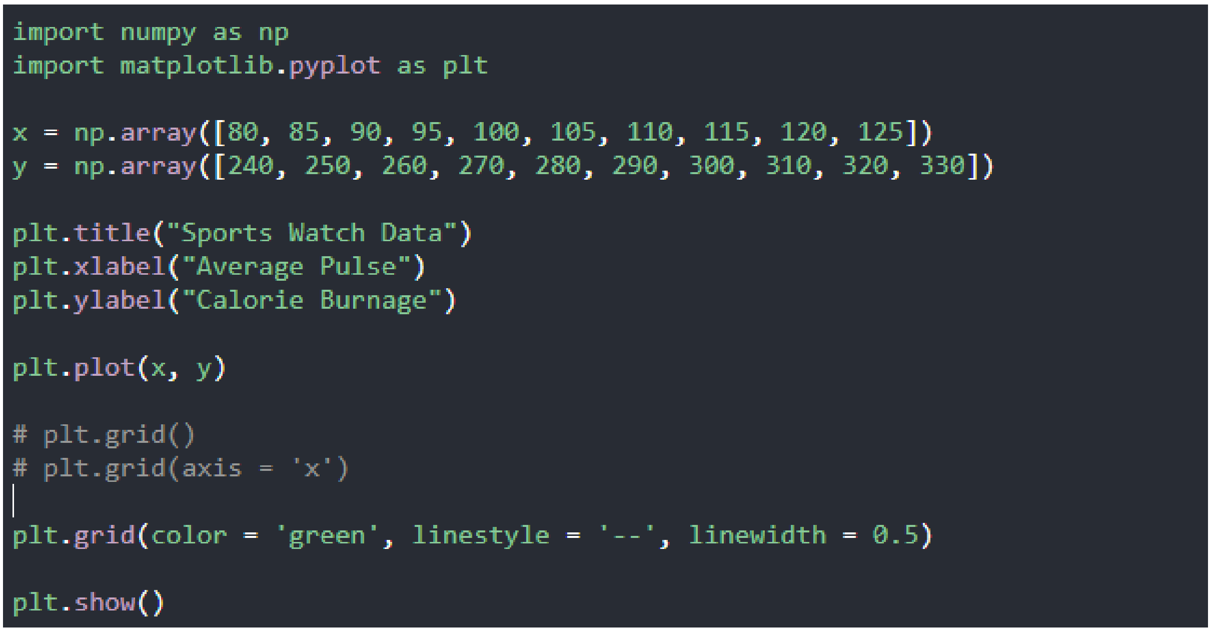
Labels and Title



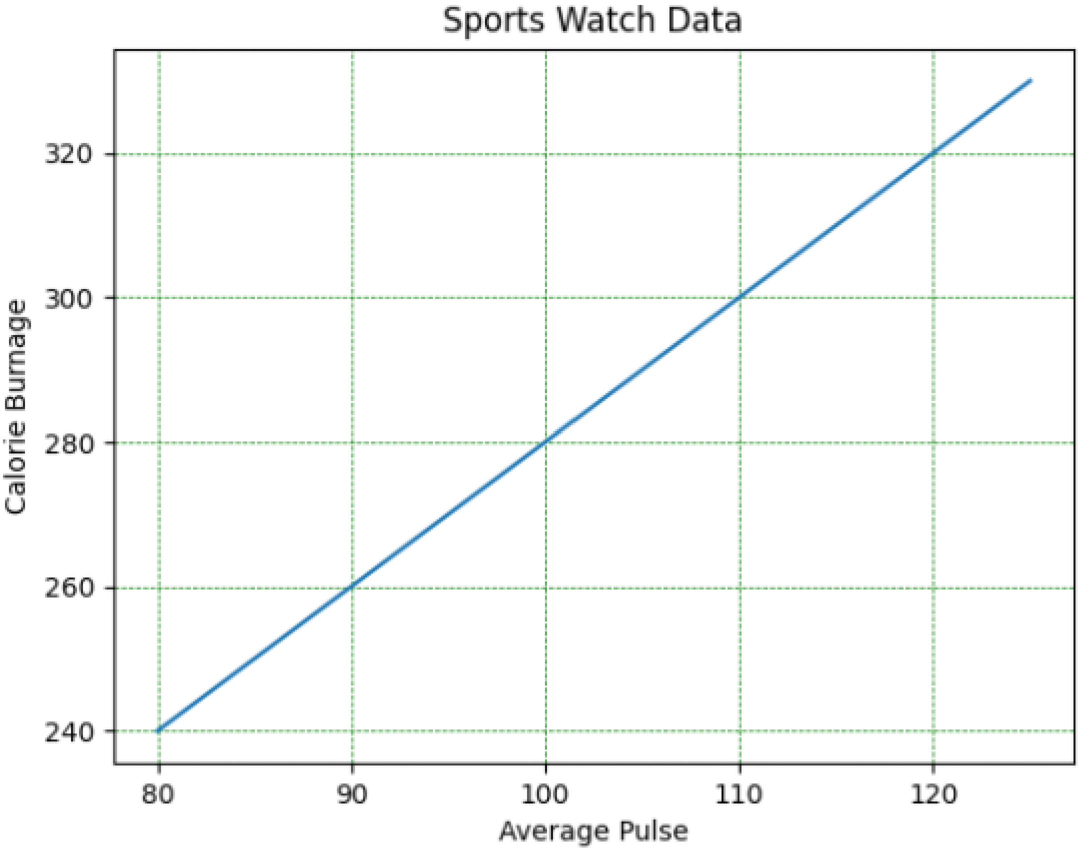
```
#Three lines to make our compiler able to draw:
import sys
import matplotlib
matplotlib.use('Agg')

import numpy as np
import matplotlib.pyplot as plt

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}

#Legal values are: 'left', 'right', and 'center'. Default value is 'center'.
plt.title("Sports Watch Data", fontdict = font1, loc = 'right')

plt.xlabel("Average Pulse", fontdict = font2)
plt.ylabel("Calorie Burnage", fontdict = font2)

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

#Two  lines to make our compiler able to draw:
plt.savefig(sys.stdout.buffer)
sys.stdout.flush()
```



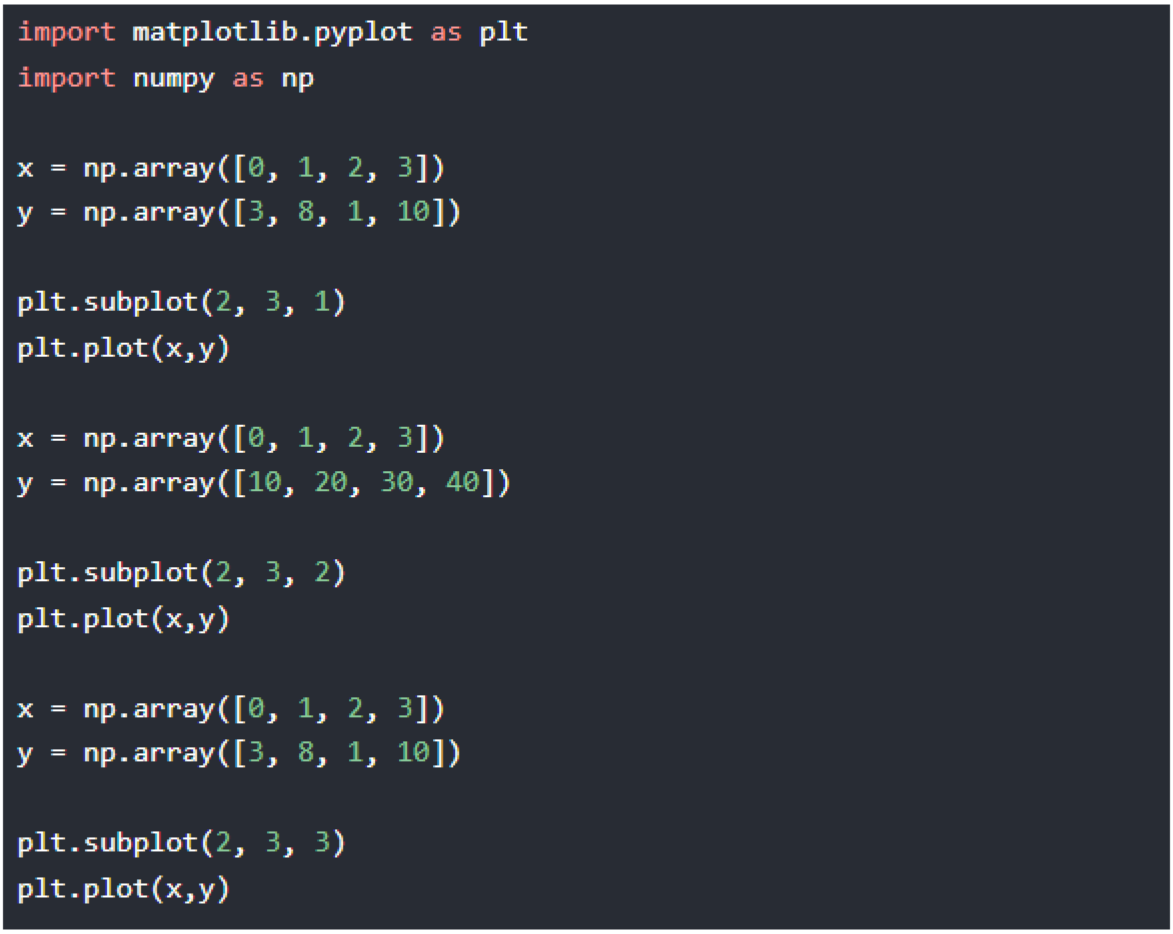
Grid Lines



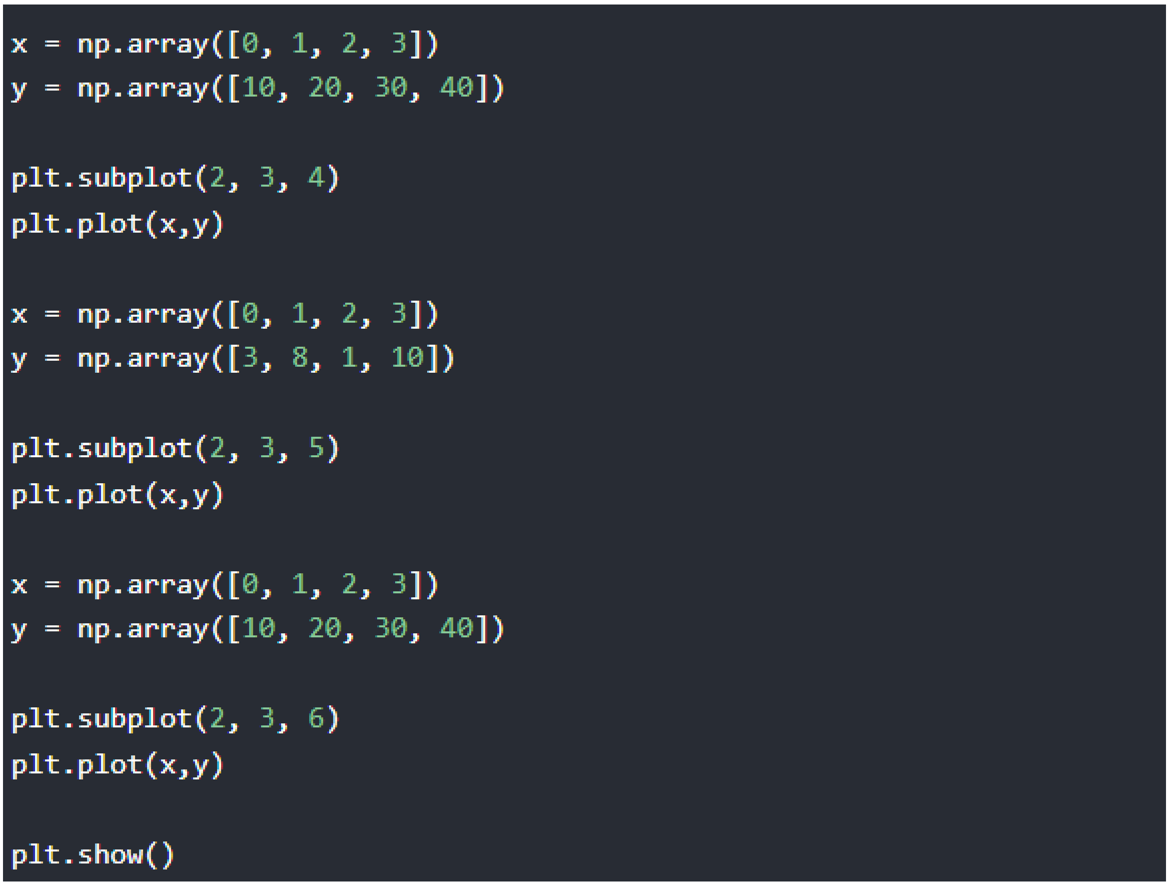
```
import numpy as np
import matplotlib.pyplot as plt

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.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")

plt.plot(x, y)

# plt.grid()
# plt.grid(axis = 'x')

plt.grid(color = 'green', linestyle = '--', linewidth = 0.5)

plt.show()
```


Sports Watch Data
Calorie Burnage
Average Pulse
240
260
280
300
320
80
90
100
110
120

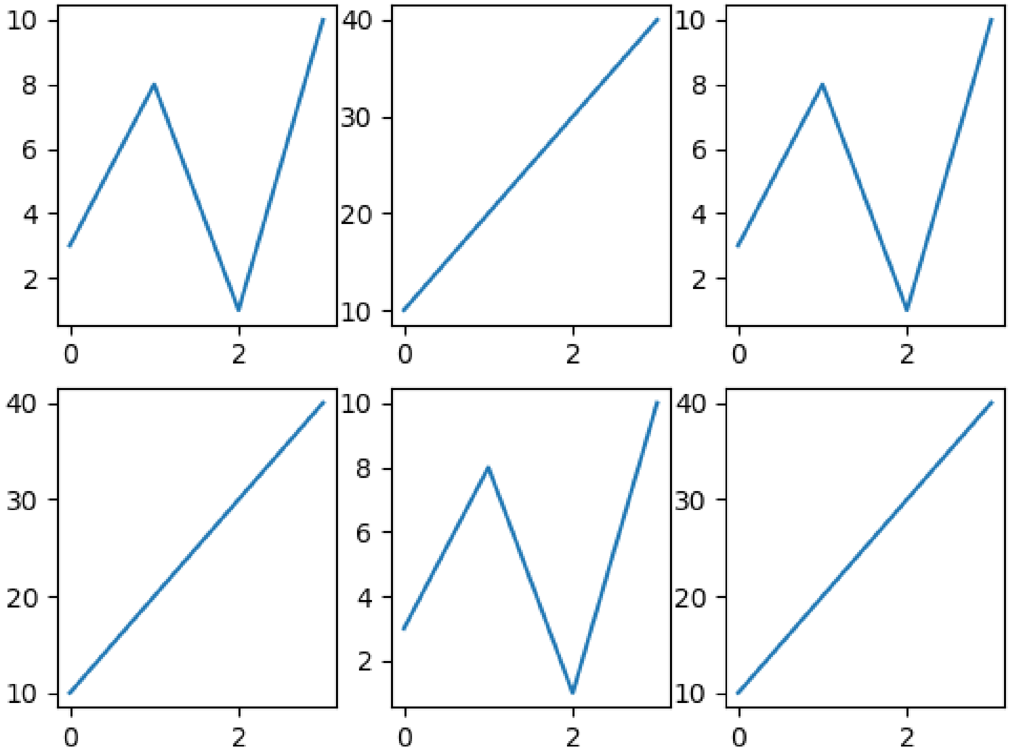
Subplots

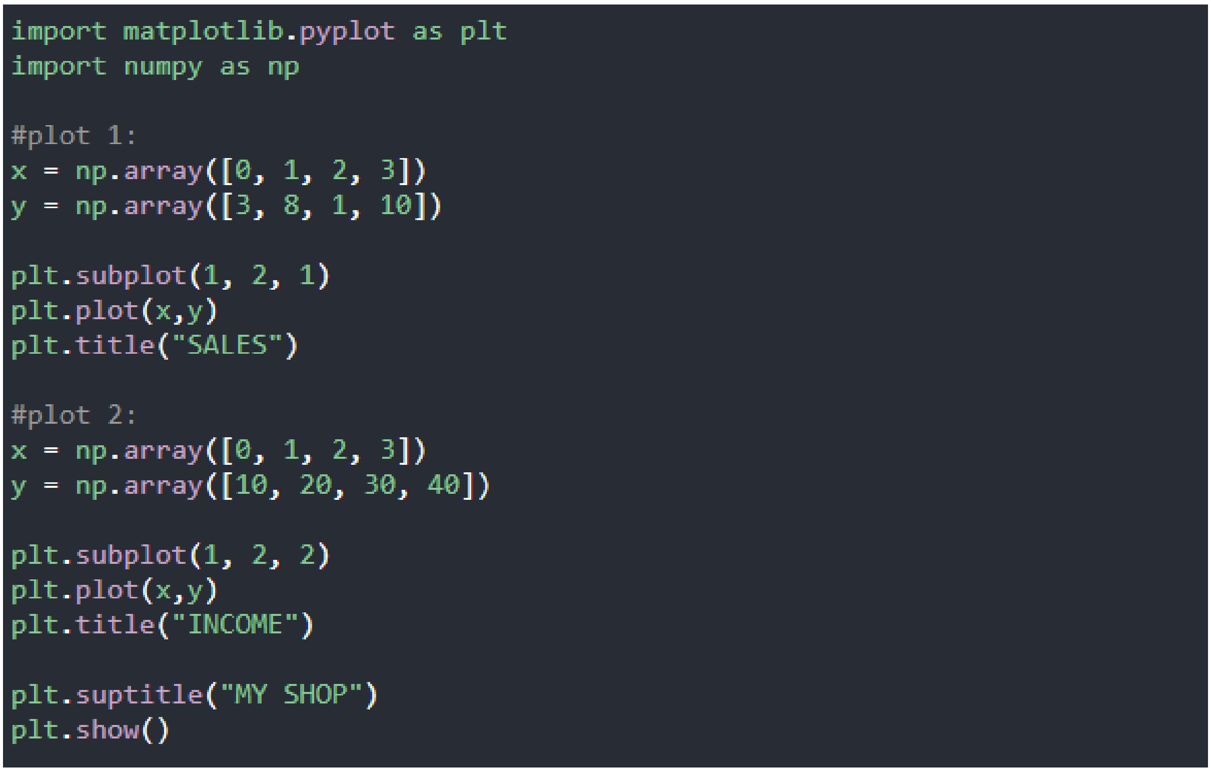


```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(2, 3, 1)
plt.plot(x,y)

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

plt.subplot(2, 3, 2)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(2, 3, 3)
plt.plot(x,y)
```

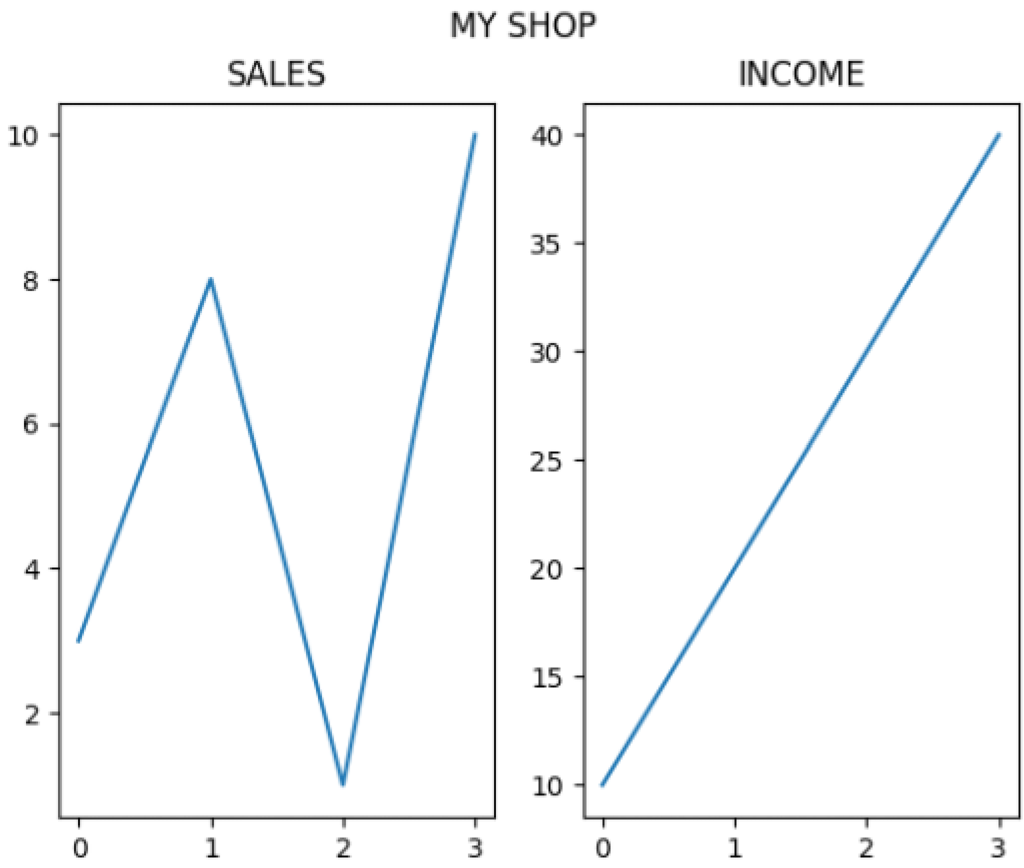


```
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])

plt.subplot(2, 3, 4)
plt.plot(x,y)

x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])

plt.subplot(2, 3, 5)
plt.plot(x,y)

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

plt.subplot(2, 3, 6)
plt.plot(x,y)

plt.show()
```



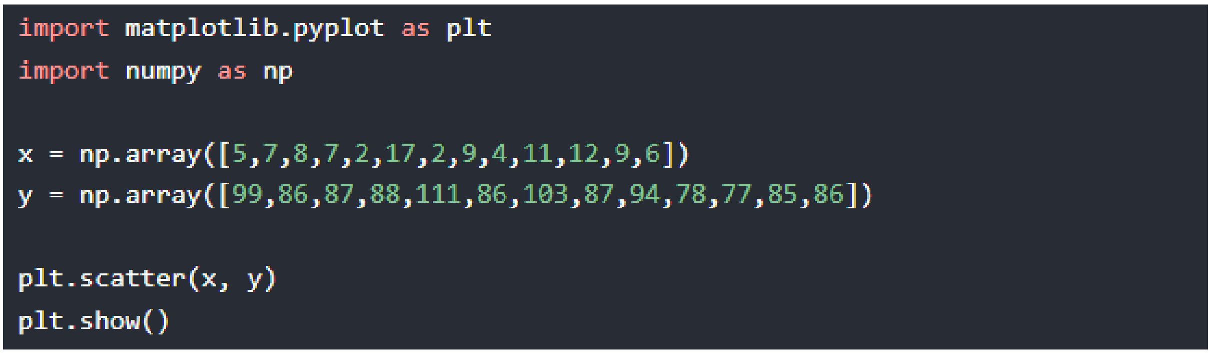


```
import matplotlib.pyplot as plt
import numpy as np

#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()
```

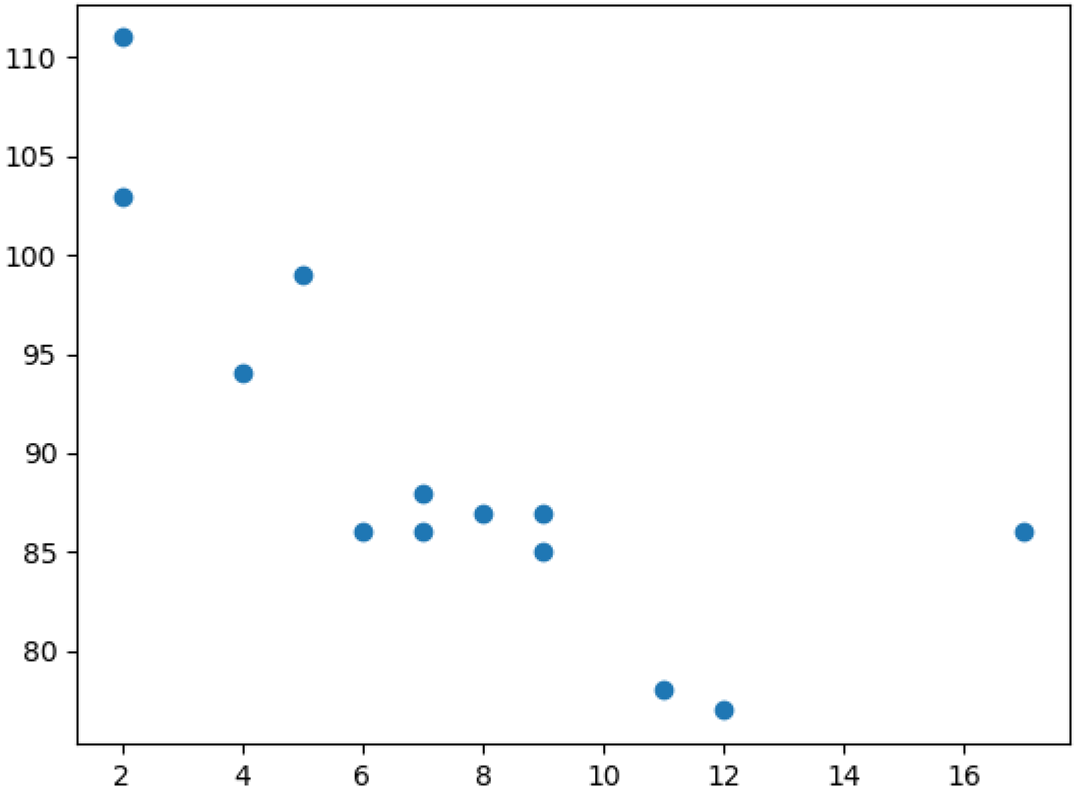


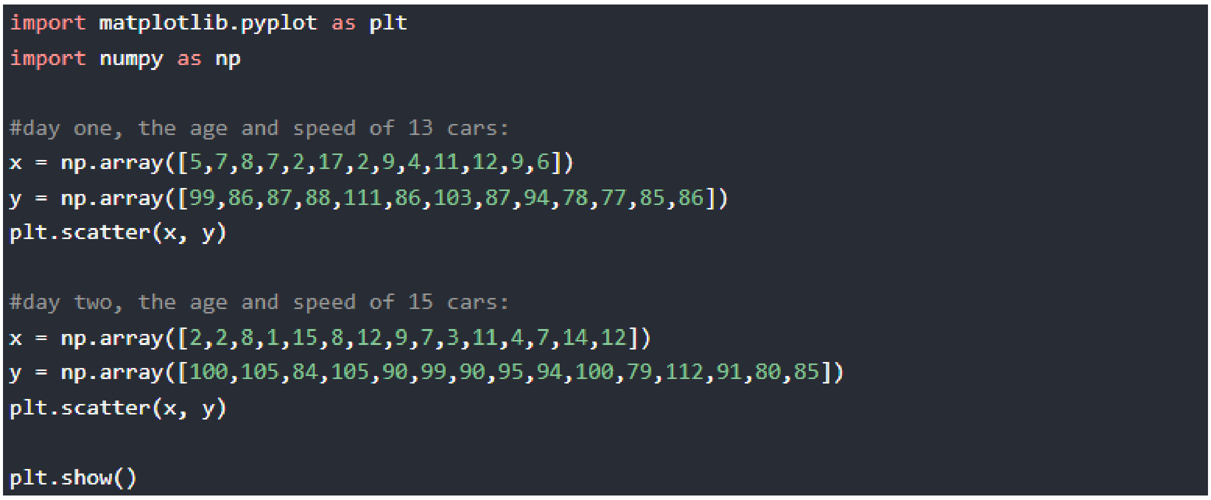
Scatter

The `scatter()` function plots one dot for each observation. It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis:

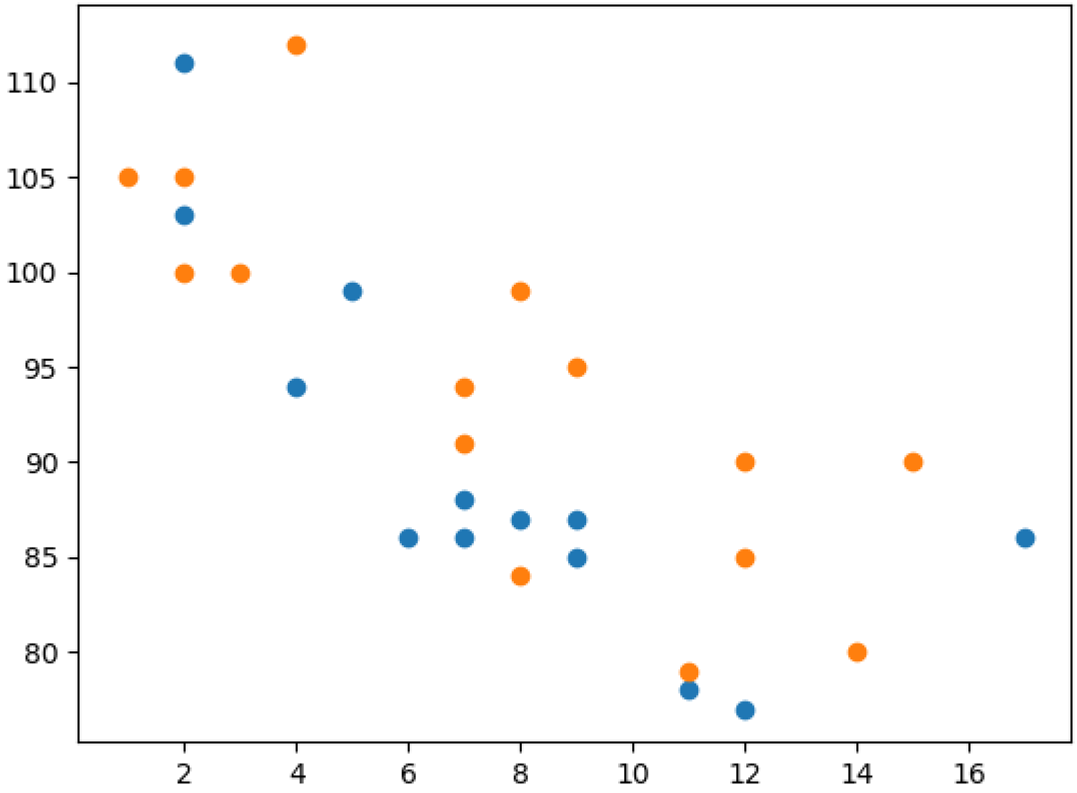


```
import matplotlib.pyplot as plt
import numpy as np

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()
```

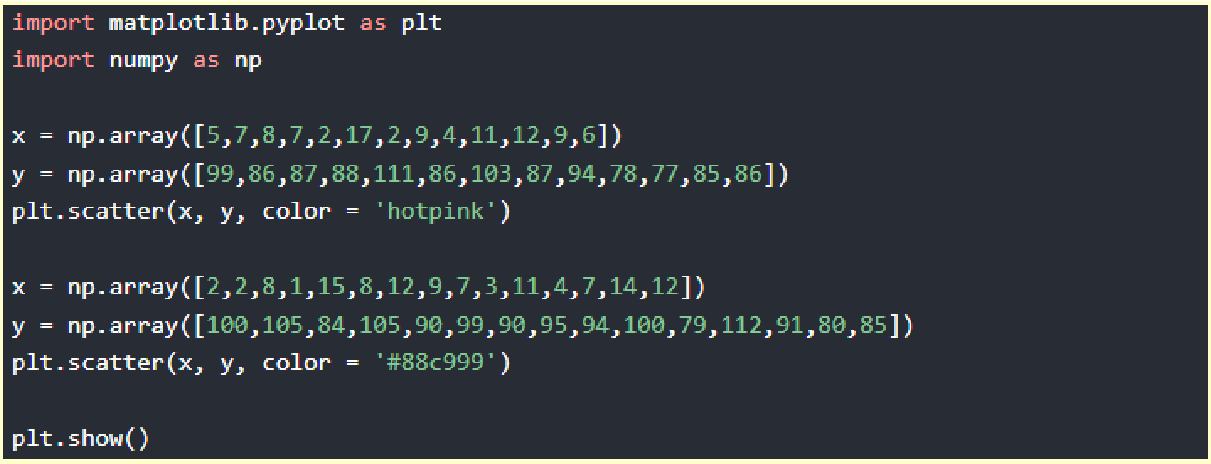




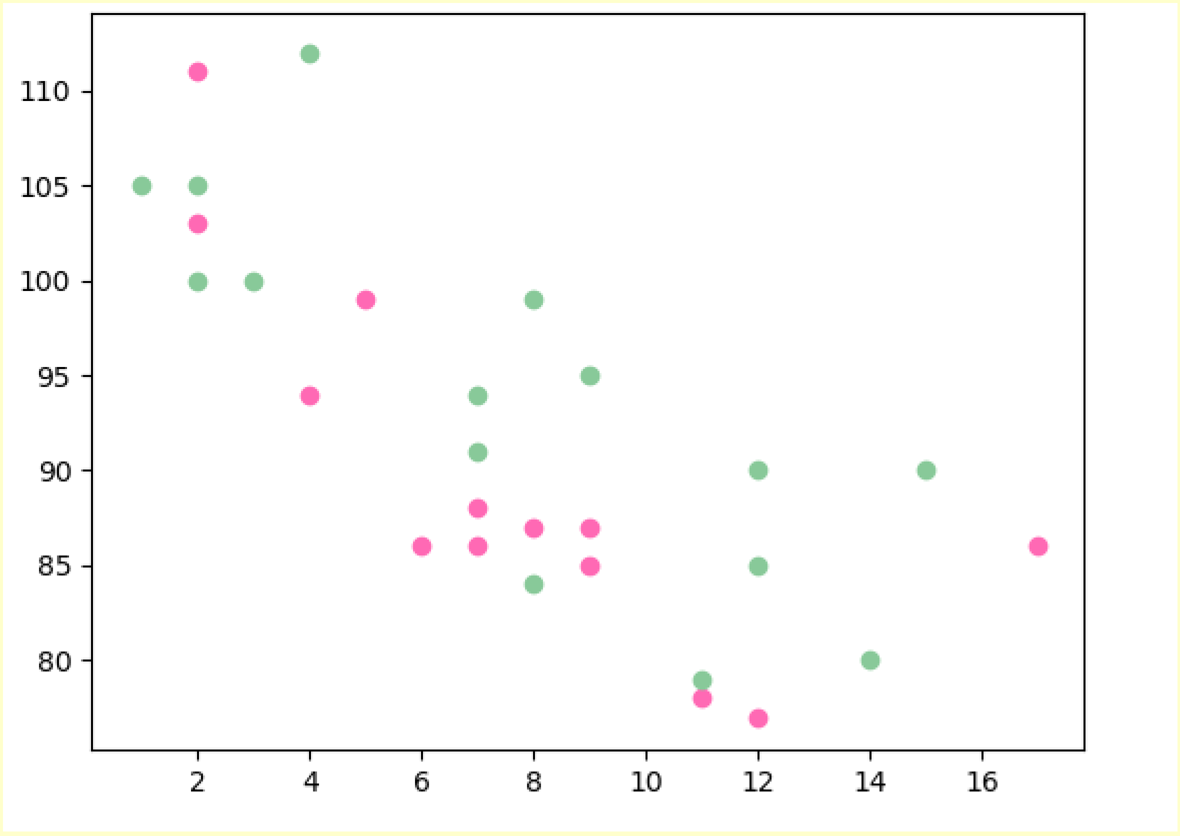
```
import matplotlib.pyplot as plt
import numpy as np

#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)

plt.show()
```



The two plots are plotted with two different colors, by default blue and orange, you will learn how to change colors later in this chapter.



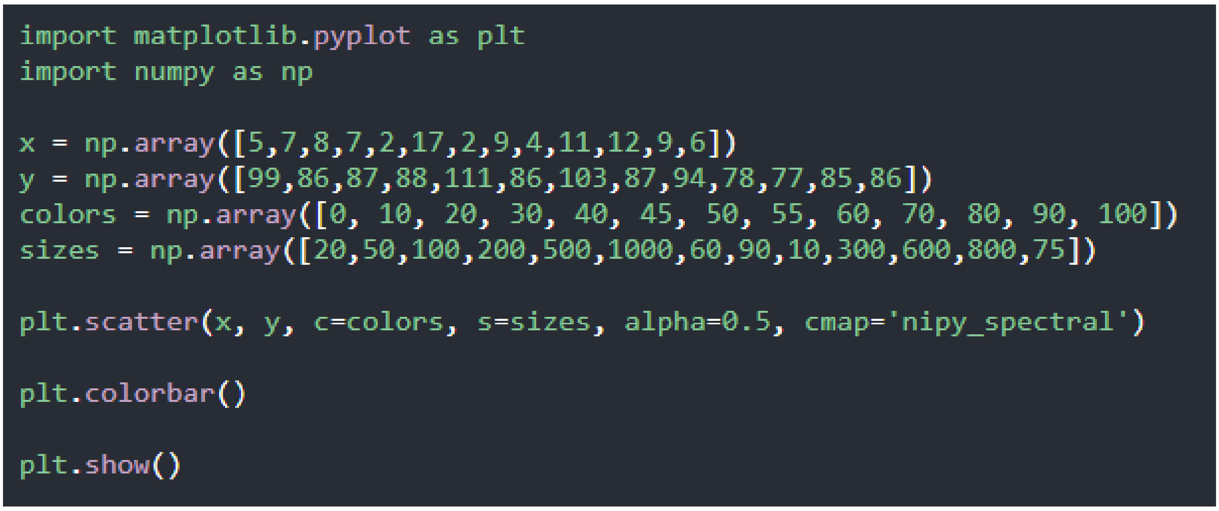
```
import matplotlib.pyplot as plt
import numpy as np

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, color = 'hotpink')

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 = '#88c999')

plt.show()
```



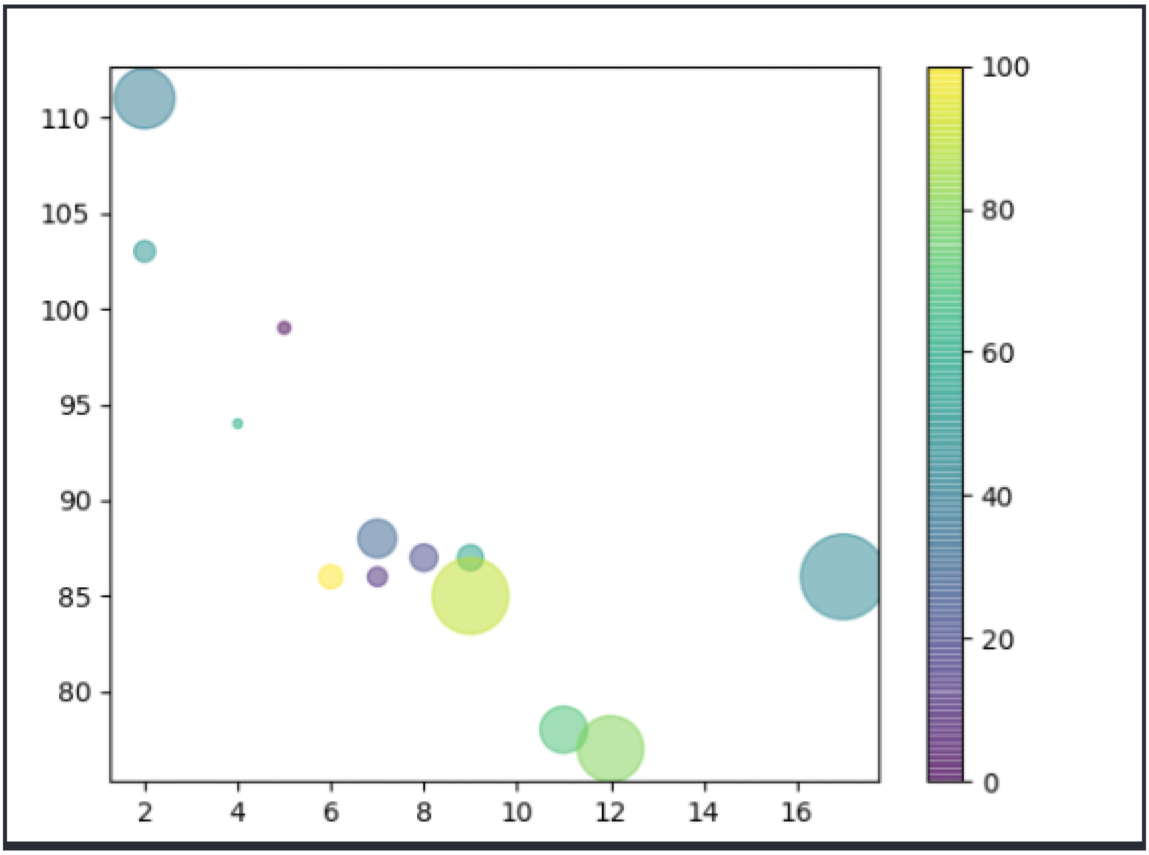
Color Each Dot

```
np.array(["red","green","blue","yellow","pink","black","orange","purple","beige","brown","gray","cyan","magenta"])
```

```
plt.scatter(x, y, c=colors)
```

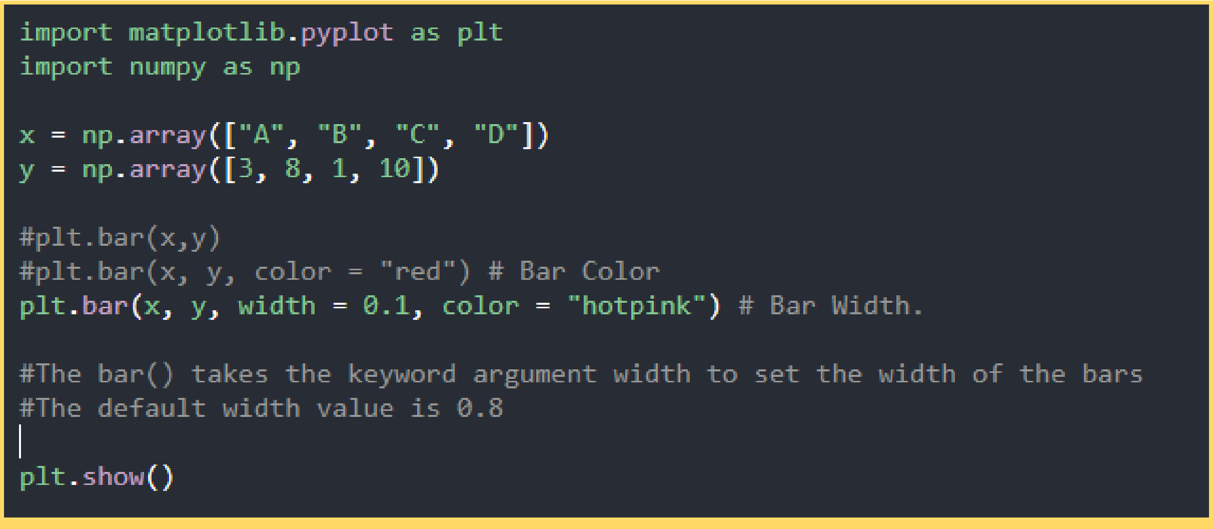


```
import matplotlib.pyplot as plt
import numpy as np

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])
colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])
sizes = np.array([20,50,100,200,500,1000,60,90,10,300,600,800,75])

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

plt.colorbar()

plt.show()
```

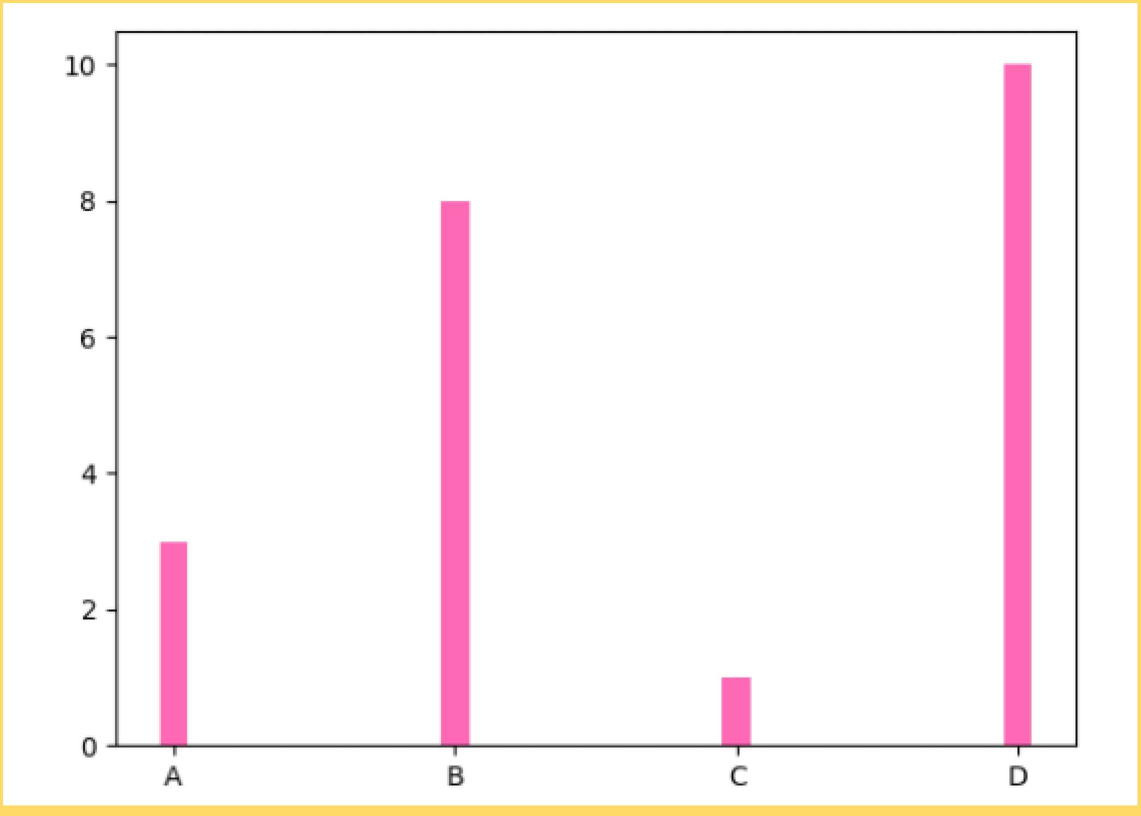


https://www.w3schools.com/python/matplotlib_scatter.asp

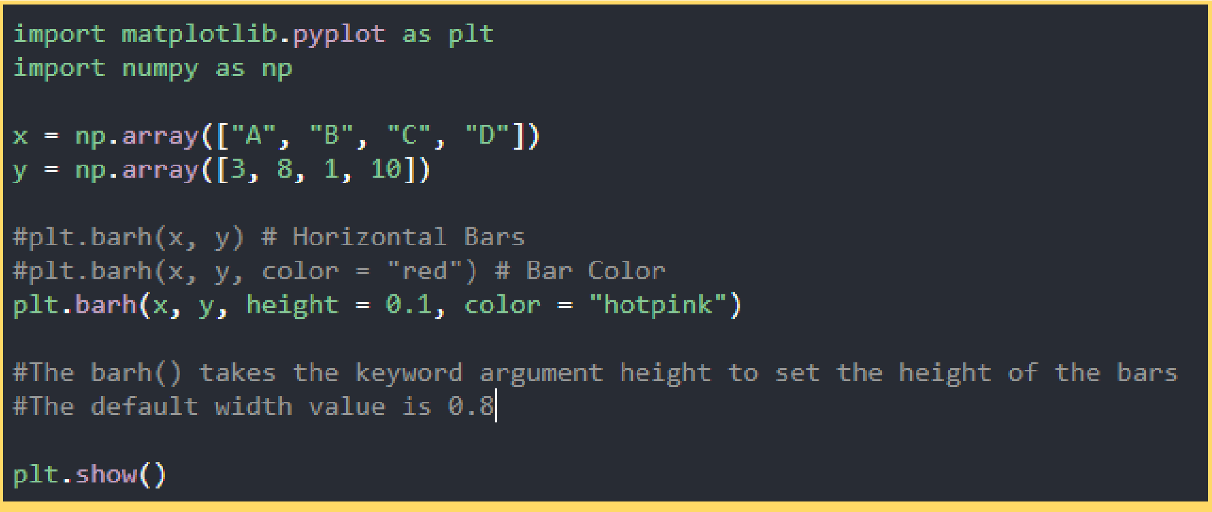
Bars



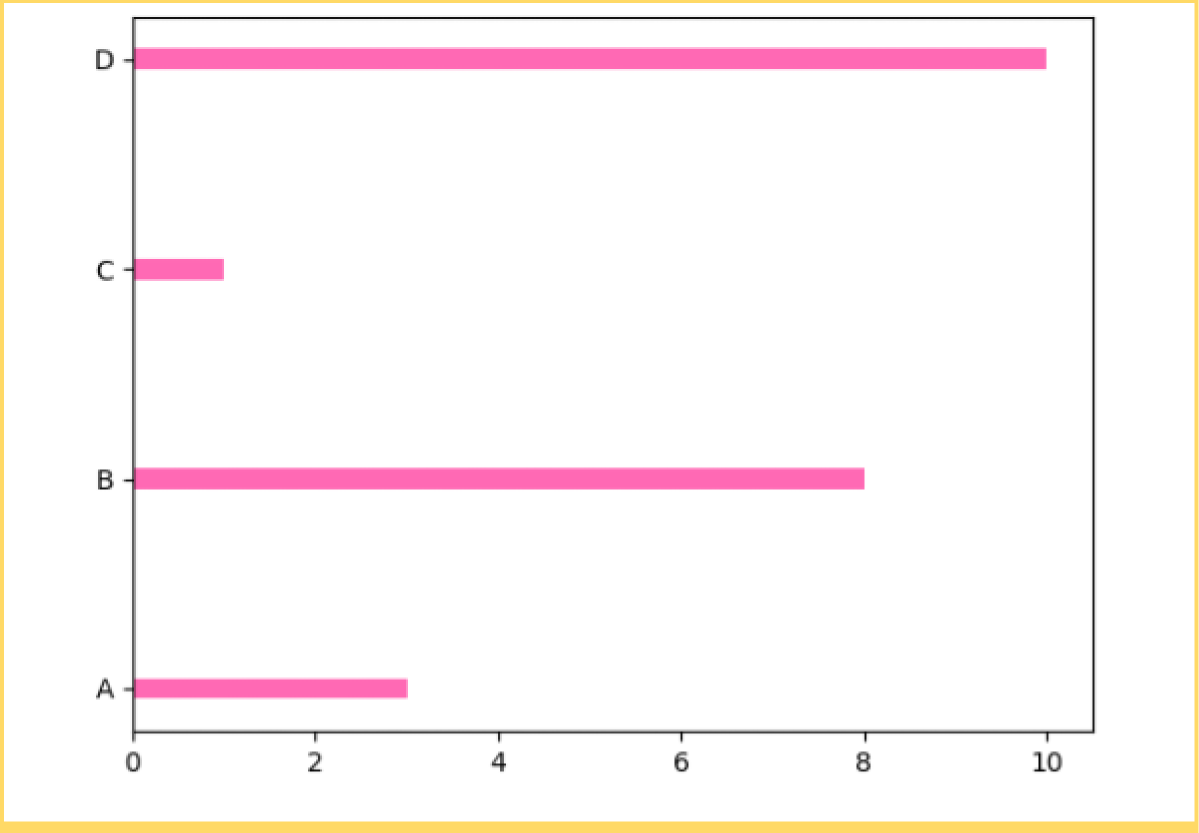
```python
import matplotlib.pyplot as plt
import numpy as np

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

#plt.bar(x,y)
#plt.bar(x, y, color = "red") # Bar Color
plt.bar(x, y, width = 0.1, color = "hotpink") # Bar Width.

#The bar() takes the keyword argument width to set the width of the bars
#The default width value is 0.8

plt.show()
```



The bars to be displayed **horizontally** instead of vertically, use the barh() function. The barh() takes the keyword argument height to set the height of the bars. The default width value is 0.8

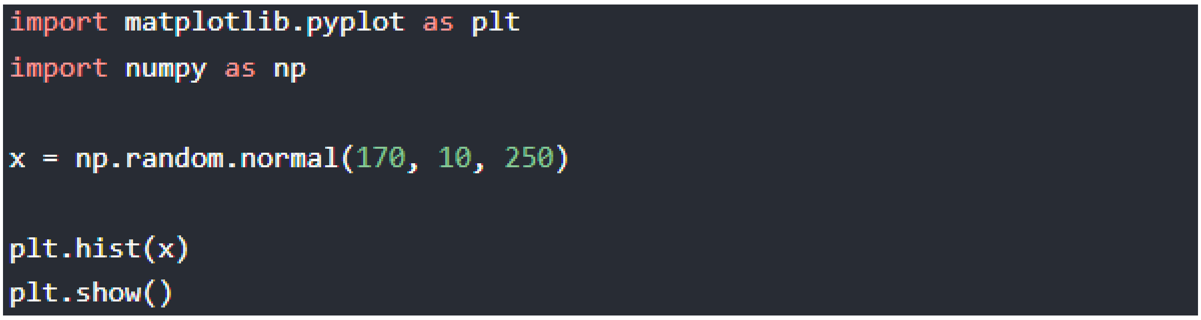


```
import matplotlib.pyplot as plt
import numpy as np

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

#plt.barh(x, y) # Horizontal Bars
#plt.barh(x, y, color = "red") # Bar Color
plt.barh(x, y, height = 0.1, color = "hotpink")

#The barh() takes the keyword argument height to set the height of the bars
#The default width value is 0.8

plt.show()
```

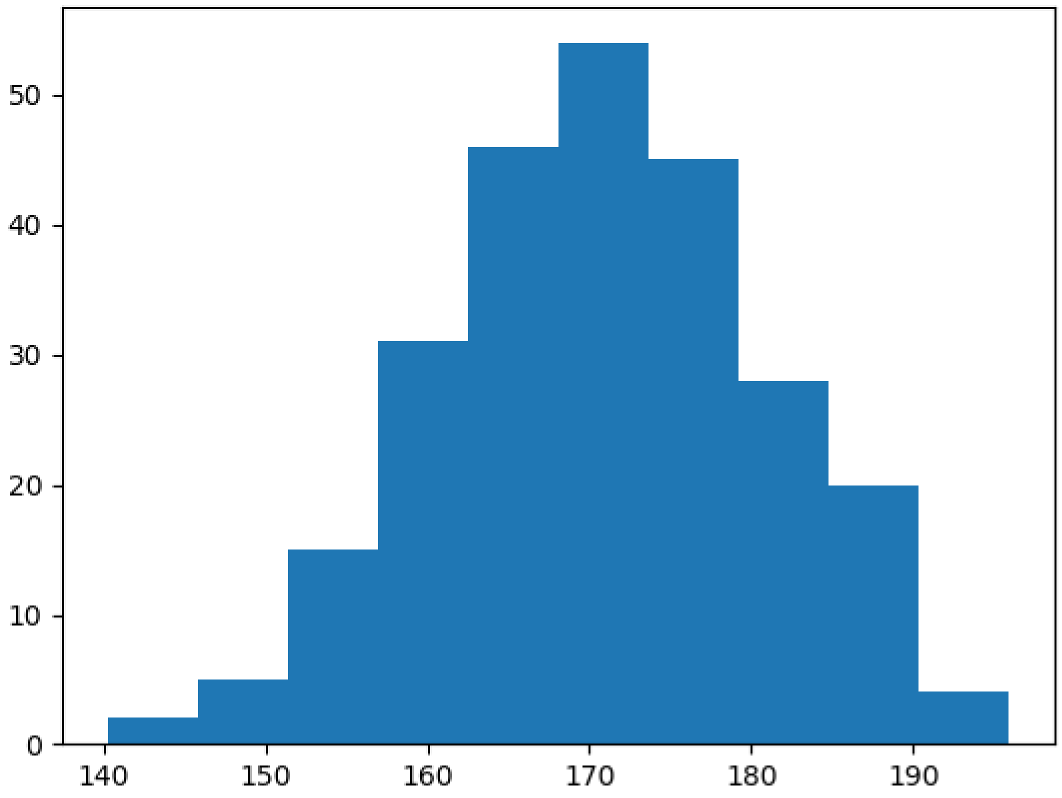


Histograms

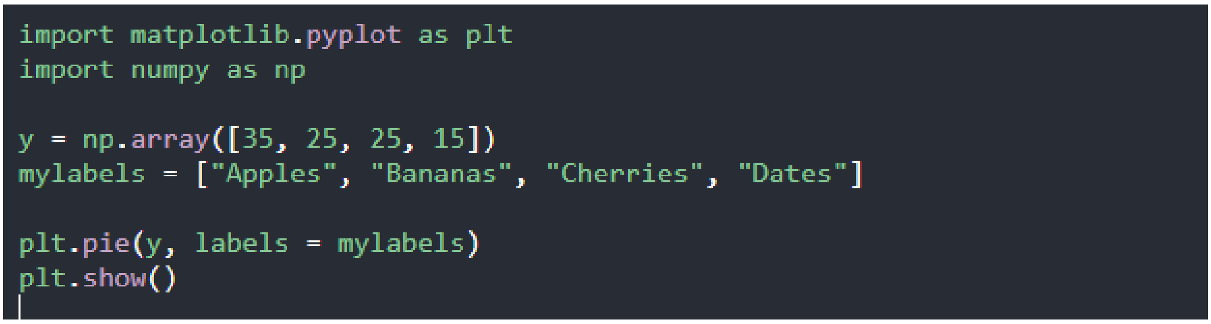
A histogram is a graph showing *frequency* distributions. It is a graph showing the number of observations within each given interval.



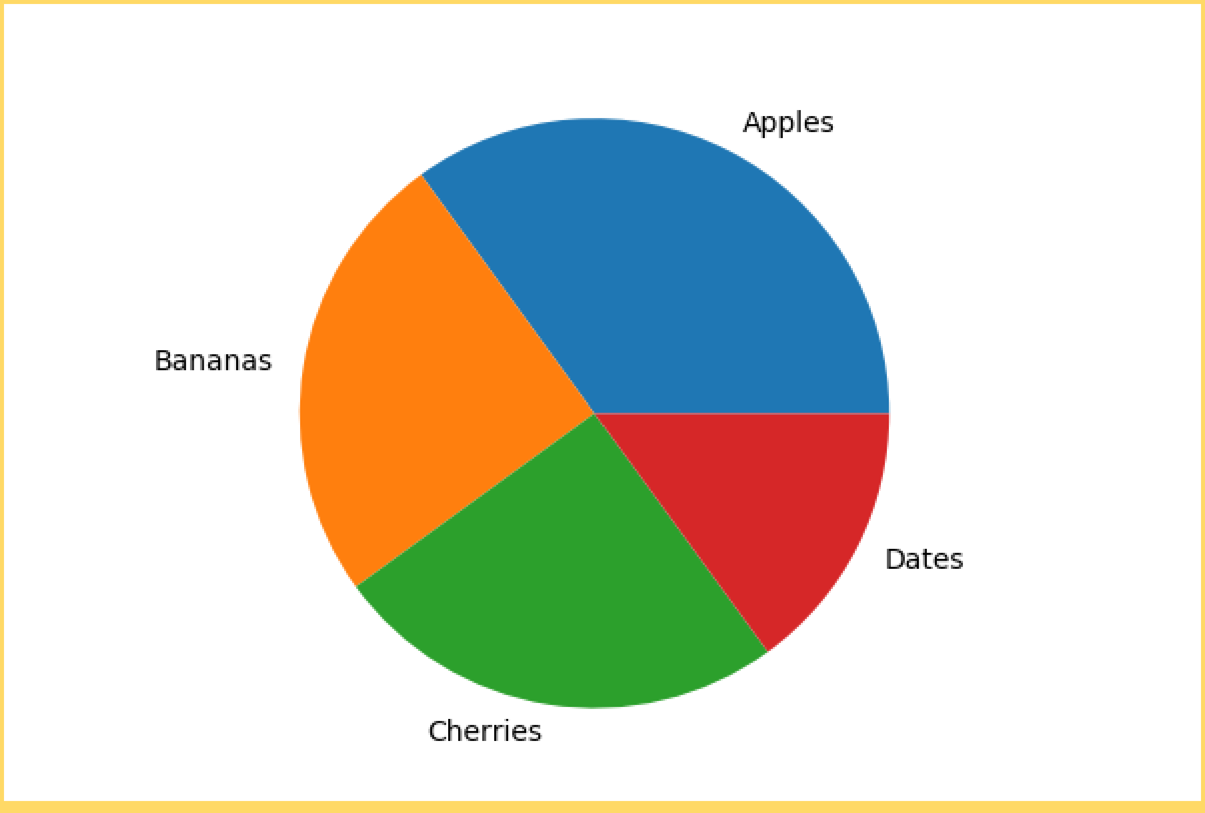
```
import matplotlib.pyplot as plt
import numpy as np

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

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



Pie Charts



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

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]

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



The default **start angle** is at the x-axis, but you can change the start angle.Default angle is 0.

Explode parameter,Each value represents how far from the center each wedge is displayed.Pull the "Apples" wedge 0.2 from the center of the pie.

Add a **shadow** to the pie chart by setting the shadows parameter to True

The **colors** parameter, if specified, must be an array with one value for each wedge.

'r' - Red

'g' - Green

'b' - Blue

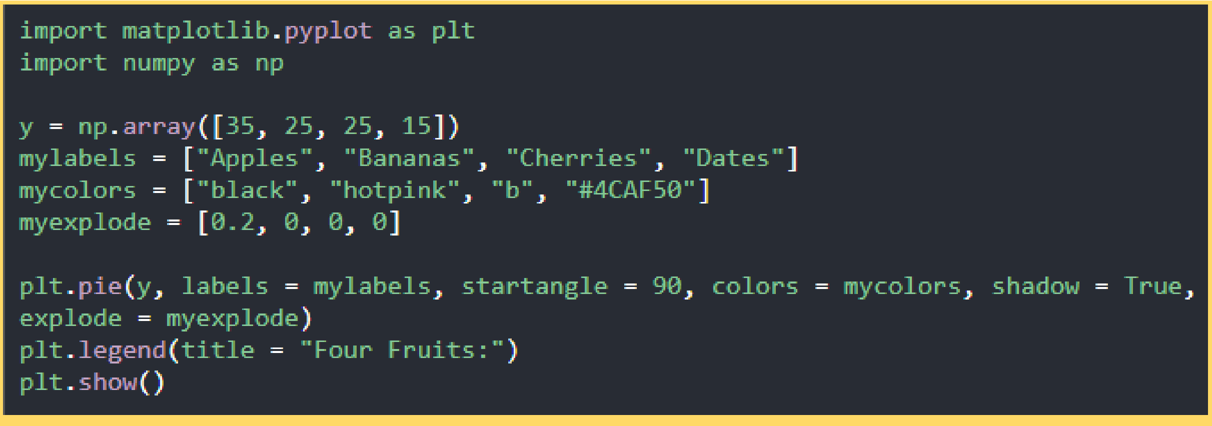
'c' - Cyan

'm' - Magenta

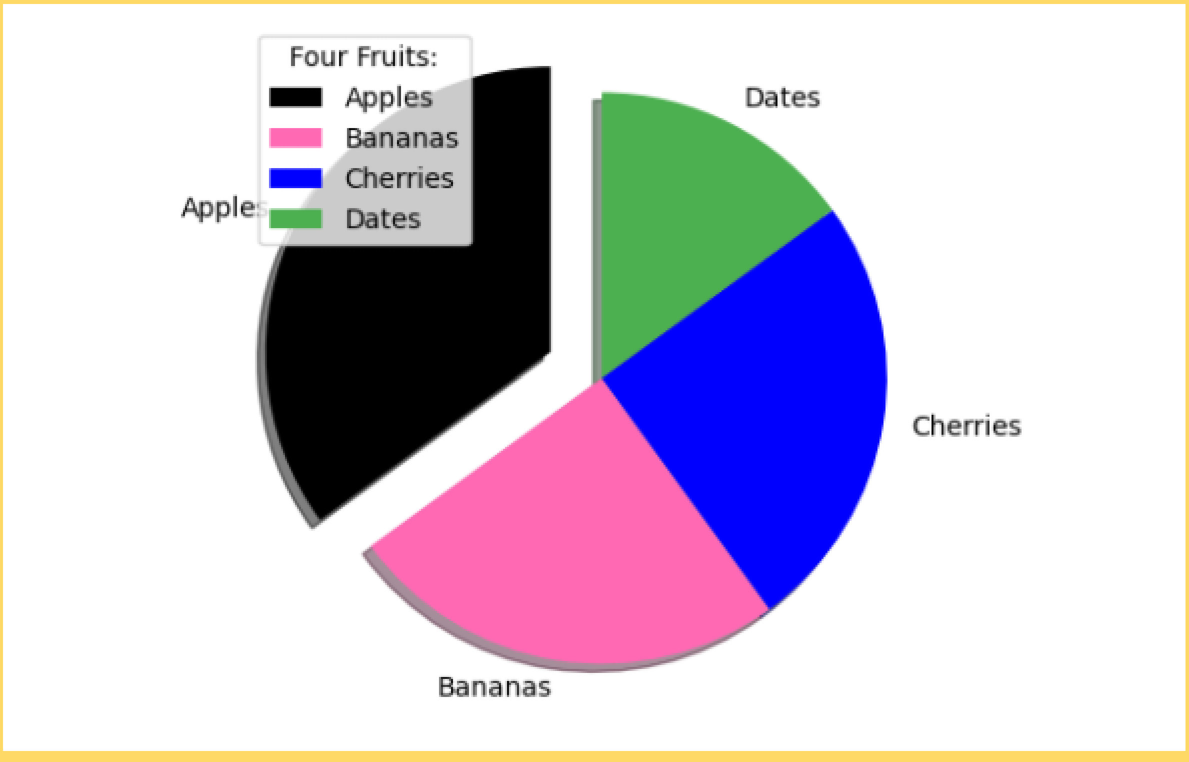
'y' - Yellow

'k' - Black

'w' - White



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

y = np.array([35, 25, 25, 15])
mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
mycolors = ["black", "hotpink", "b", "#4CAF50"]
myexplode = [0.2, 0, 0, 0]

plt.pie(y, labels = mylabels, startangle = 90, colors = mycolors, shadow = True,
explode = myexplode)
plt.legend(title = "Four Fruits:")
plt.show()
```


Four Fruits:
■ Apples
🟥 Bananas
🟦 Cherries
🟩 Dates
Apples
Dates
Cherries
Bananas