Matplotlib Line

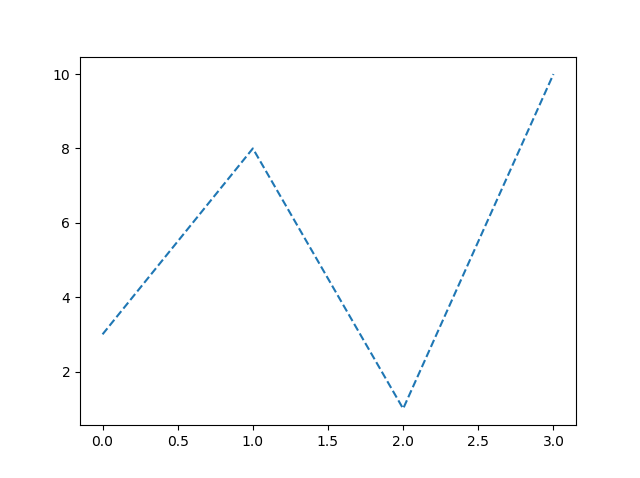
## **Linestyle**

You can use the keyword argument linestyle, or shorter ls, to change the style of the plotted line:

### **Example**

Use a dotted line:

import matplotlib.pyplot as plt  
import numpy as np  
  
ypoints = np.array([3, 8, 1, 10])  
  
plt.plot(ypoints, linestyle = 'dotted')  
plt.show()



## **Shorter Syntax**

The line style can be written in a shorter syntax:

linestyle can be written as ls.

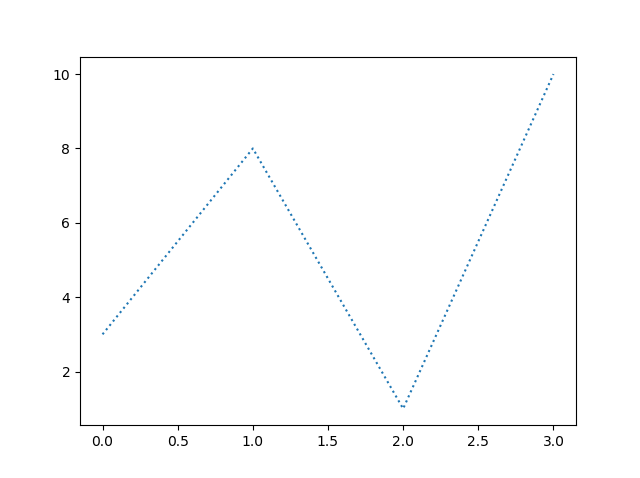
dotted can be written as :.

dashed can be written as --.

### **Example**

Shorter syntax:

plt.plot(ypoints, ls = ':')



## **Line Styles**

You can choose any of these styles:

|  |  |
| --- | --- |
| **Style** | **Or** |
| 'solid' (default) | '-' |
| 'dotted' | ':' |
| 'dashed' | '--' |
| 'dashdot' | '-.' |
| 'None' | '' or ' ' |

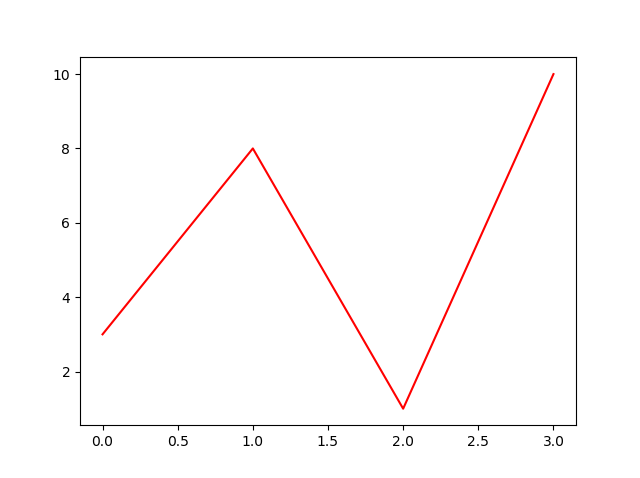
## **Line Color**

You can use the keyword argument color or the shorter c to set the color of the line:

### **Example**

Set the line color to red:

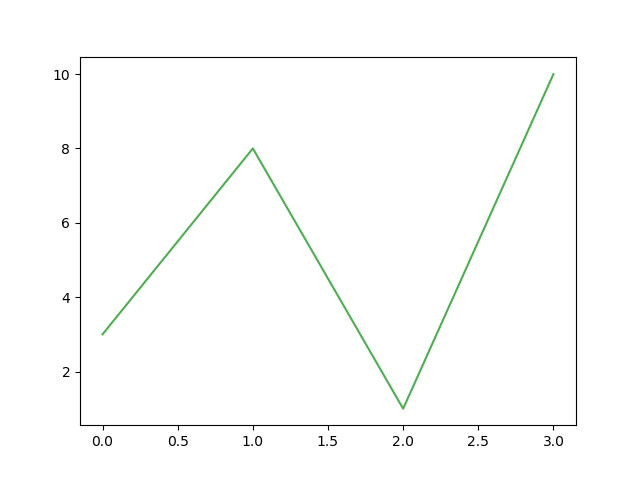
import matplotlib.pyplot as plt  
import numpy as np  
  
ypoints = np.array([3, 8, 1, 10])  
  
plt.plot(ypoints, color = 'r')  
plt.show()



### **Example**

Plot with a beautiful green line:

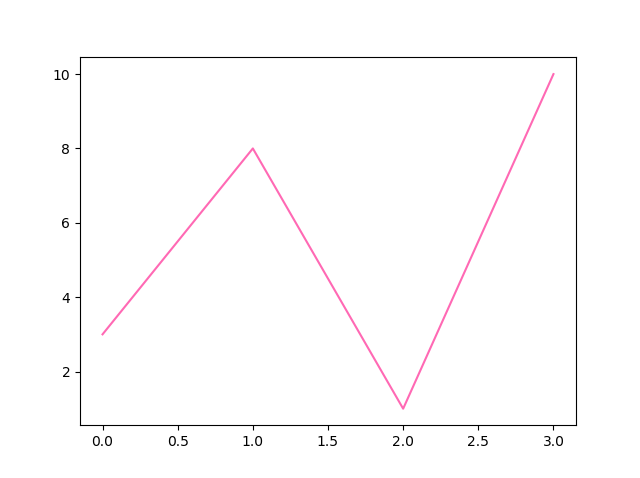
...  
plt.plot(ypoints, c = '#4CAF50')  
...



### **Example**

Plot with the color named "hotpink":

...  
plt.plot(ypoints, c = 'hotpink')  
...



## **Line Width**

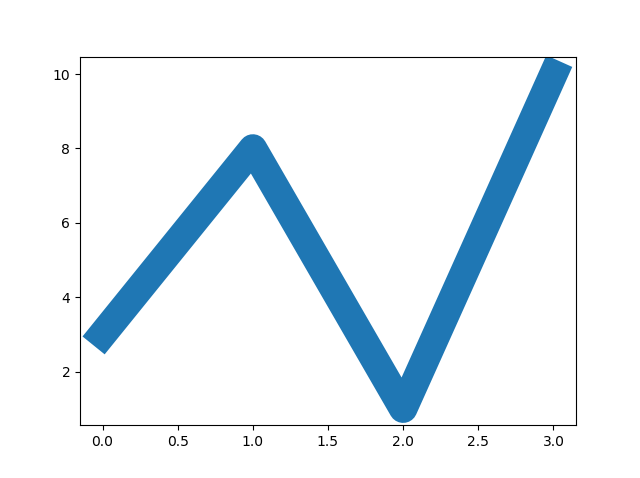
You can use the keyword argument linewidth or the shorter lw to change the width of the line.

The value is a floating number, in points:

### **Example**

Plot with a 20.5pt wide line:

import matplotlib.pyplot as plt  
import numpy as np  
  
ypoints = np.array([3, 8, 1, 10])  
  
plt.plot(ypoints, linewidth = '20.5')  
plt.show()



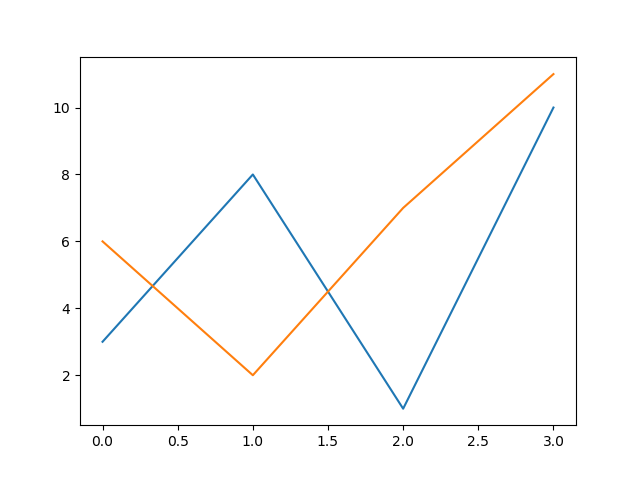
## **Multiple Lines**

You can plot as many lines as you like by simply adding more plt.plot() functions:

### **Example**

Draw two lines by specifying a plt.plot() function for each line:

import matplotlib.pyplot as plt  
import numpy as np  
  
y1 = np.array([3, 8, 1, 10])  
y2 = np.array([6, 2, 7, 11])  
  
plt.plot(y1)  
plt.plot(y2)  
  
plt.show()



You can also plot many lines by adding the points for the x- and y-axis for each line in the same plt.plot() function.

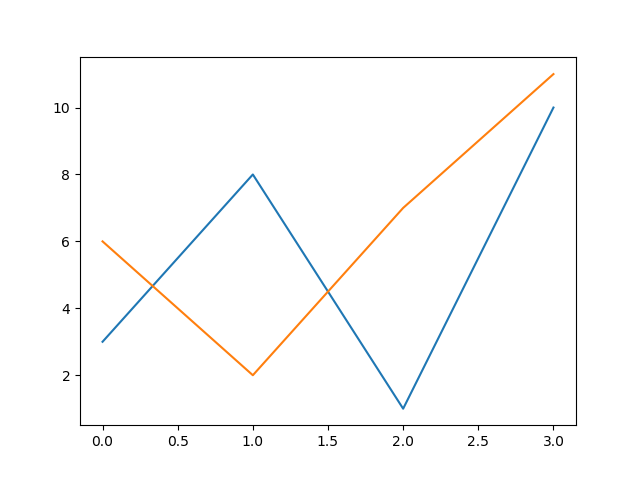
(In the examples above we only specified the points on the y-axis, meaning that the points on the x-axis got the the default values (0, 1, 2, 3).)

The x- and y- values come in pairs:

### **Example**

Draw two lines by specifiyng the x- and y-point values for both 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()



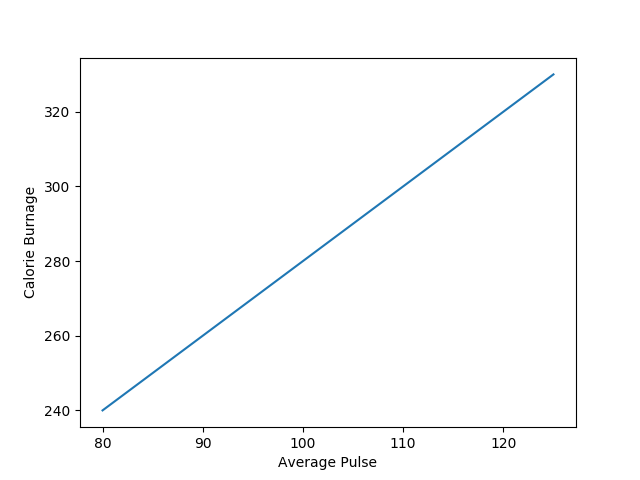
# Matplotlib Labels and Title

## **Create Labels for a Plot**

With Pyplot, you can use the xlabel() and ylabel() functions to set a label for the x- and y-axis.

### **Example**

Add labels to the x- and y-axis:

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.plot(x, y)  
  
plt.xlabel("Average Pulse")  
plt.ylabel("Calorie Burnage")  
  
plt.show()

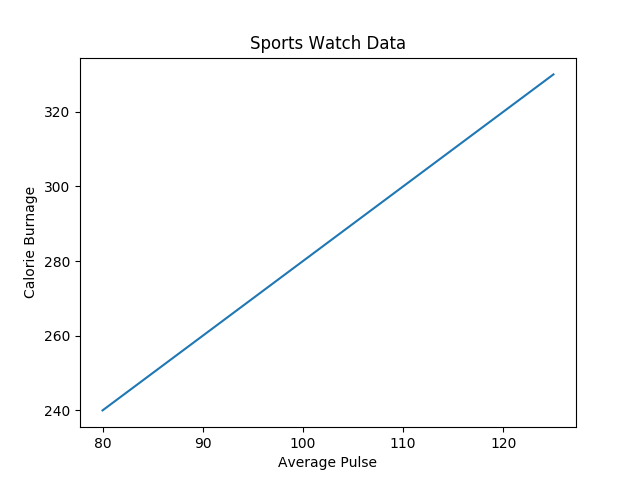
## **Create a Title for a Plot**

With Pyplot, you can use the title() function to set a title for the plot.

### **Example**

Add a plot title and labels for the x- and y-axis:

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.plot(x, y)  
  
plt.title("Sports Watch Data")  
plt.xlabel("Average Pulse")  
plt.ylabel("Calorie Burnage")  
  
plt.show()



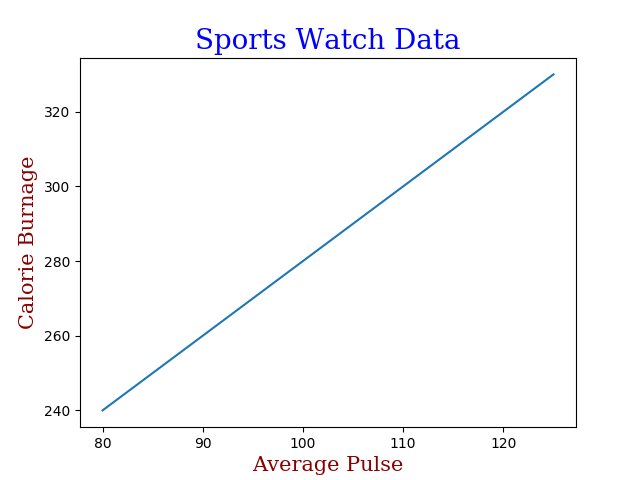
## **Set Font Properties for Title and Labels**

You can use the fontdict parameter in xlabel(), ylabel(), and title() to set font properties for the title and labels.

### **Example**

Set font properties for the title and labels:

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}  
  
plt.title("Sports Watch Data", fontdict = font1)  
plt.xlabel("Average Pulse", fontdict = font2)  
plt.ylabel("Calorie Burnage", fontdict = font2)  
  
plt.plot(x, y)  
plt.show()



## **Position the Title**

You can use the loc parameter in title() to position the title.

Legal values are: 'left', 'right', and 'center'. Default value is 'center'.

### **Example**

Position the title to the left:

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", loc = 'left')  
plt.xlabel("Average Pulse")  
plt.ylabel("Calorie Burnage")  
  
plt.plot(x, y)  
plt.show()

