Matplotlib Adding Grid Lines

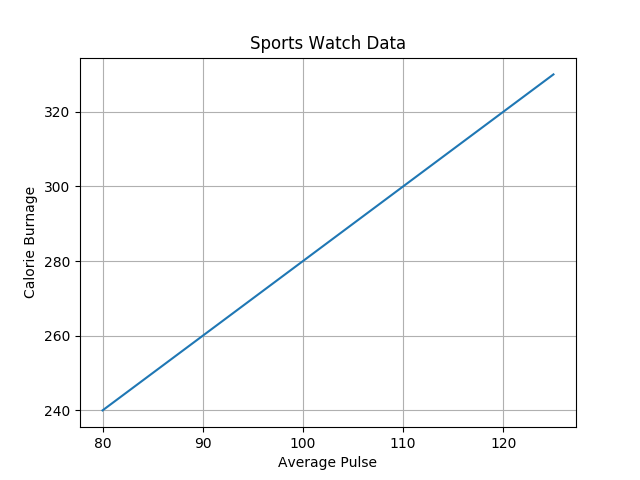
## **Add Grid Lines to a Plot**

With Pyplot, you can use the grid() function to add grid lines to the plot.

### **Example**

Add grid lines to the plot:

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.show()



## **Specify Which Grid Lines to Display**

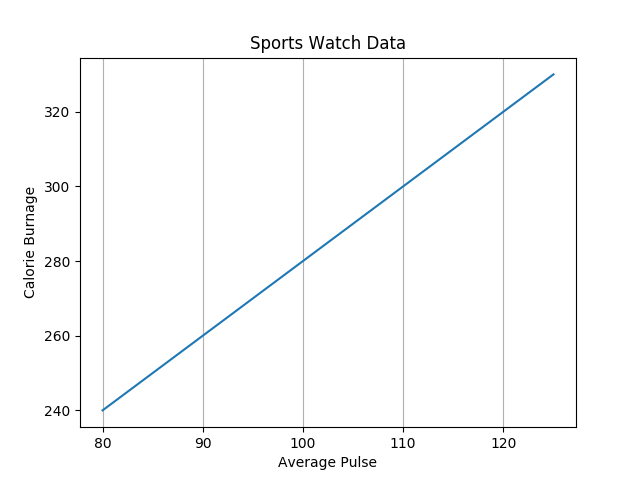
You can use the axis parameter in the grid() function to specify which grid lines to display.

Legal values are: 'x', 'y', and 'both'. Default value is 'both'.

### **Example**

Display only grid lines for the x-axis:

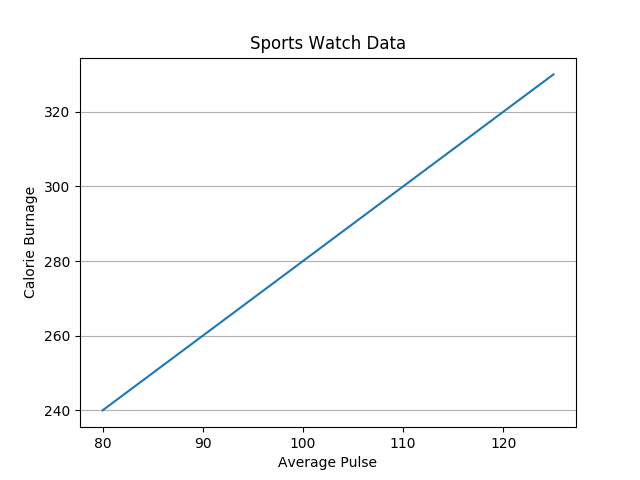
import numpy as np  
import matplotlib.pyplot as plt  
  
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])  
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])  
  
plt.title("Sports Watch Data")  
plt.xlabel("Average Pulse")  
plt.ylabel("Calorie Burnage")  
  
plt.plot(x, y)  
  
plt.grid(axis = 'x')  
  
plt.show()



### **Example**

Display only grid lines for the 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.title("Sports Watch Data")  
plt.xlabel("Average Pulse")  
plt.ylabel("Calorie Burnage")  
  
plt.plot(x, y)  
  
plt.grid(axis = 'y')  
  
plt.show()



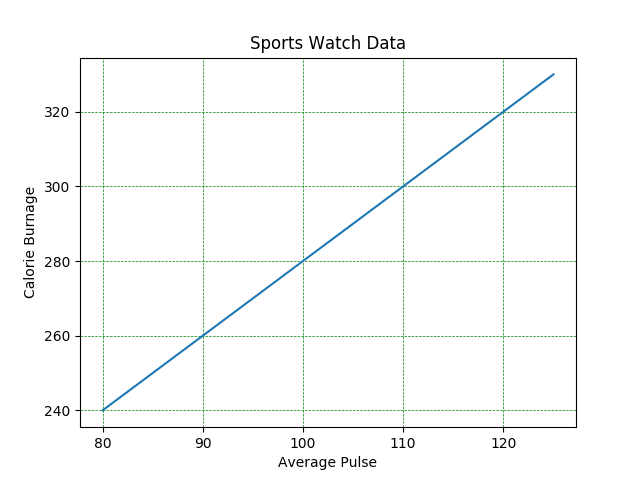
## **Set Line Properties for the Grid**

You can also set the line properties of the grid, like this: grid(color = 'color', linestyle = 'linestyle', linewidth = number).

### **Example**

Set the line properties of the grid:

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



# Matplotlib Subplot

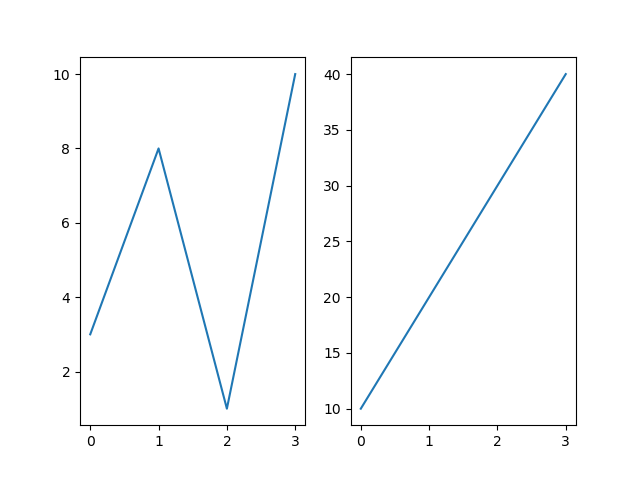
## **Display Multiple Plots**

With the subplot() function you can draw multiple plots in one figure:

### **Example**

Draw 2 plots:

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)  
  
#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()



## **The subplot() Function**

The subplot() function takes three arguments that describes the layout of the figure.

The layout is organized in rows and columns, which are represented by the first and second argument.

The third argument represents the index of the current plot.

plt.subplot(1, 2, 1)  
#the figure has 1 row, 2 columns, and this plot is the first plot.

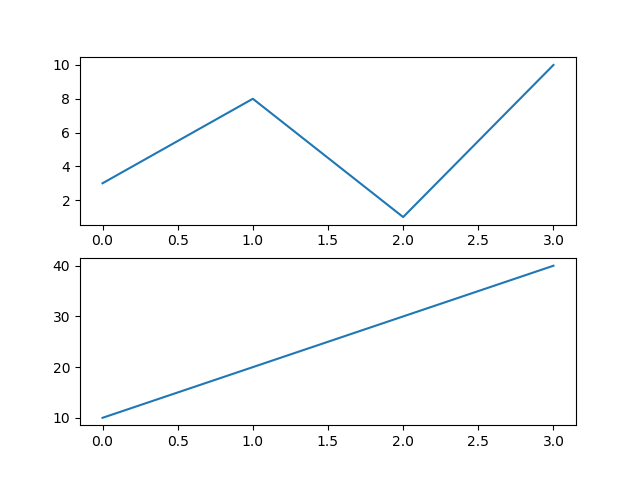
plt.subplot(1, 2, 2)  
#the figure has 1 row, 2 columns, and this plot is the second plot.

So, if we want a figure with 2 rows an 1 column (meaning that the two plots will be displayed on top of each other instead of side-by-side), we can write the syntax like this:

### **Example**

Draw 2 plots on top of each other:

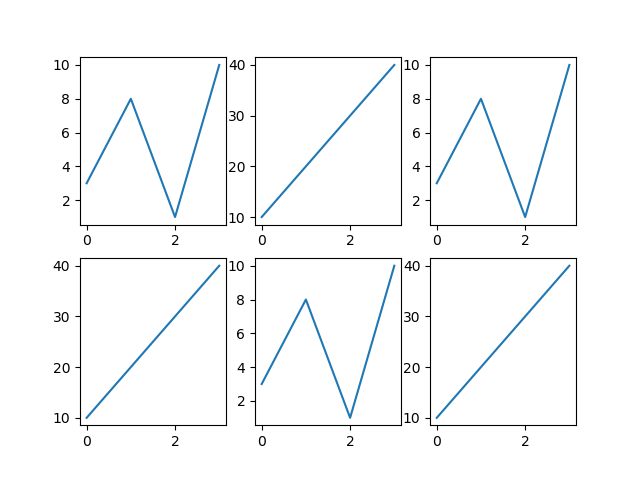
import matplotlib.pyplot as plt  
import numpy as np  
  
#plot 1:  
x = np.array([0, 1, 2, 3])  
y = np.array([3, 8, 1, 10])  
  
plt.subplot(2, 1, 1)  
plt.plot(x,y)  
  
#plot 2:  
x = np.array([0, 1, 2, 3])  
y = np.array([10, 20, 30, 40])  
  
plt.subplot(2, 1, 2)  
plt.plot(x,y)  
  
plt.show()



### **Example**

Draw 6 plots:

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()



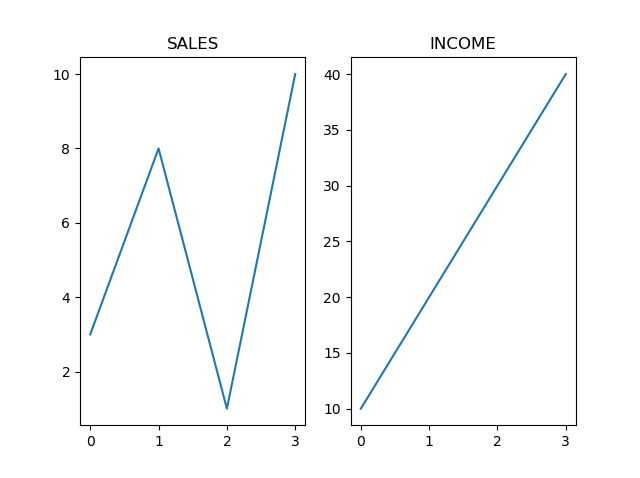
## **Title**

You can add a title to each plot with the title() function:

### **Example**

2 plots, with titles:

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.show()



## **Super Title**

You can add a title to the entire figure with the suptitle() function:

### **Example**

Add a title for the entire figure:

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()

