Matplotlib

Plotting graphs in Python

Matplotlib Import and Version

- import matplotlib

- print(matplotlib.__version__)

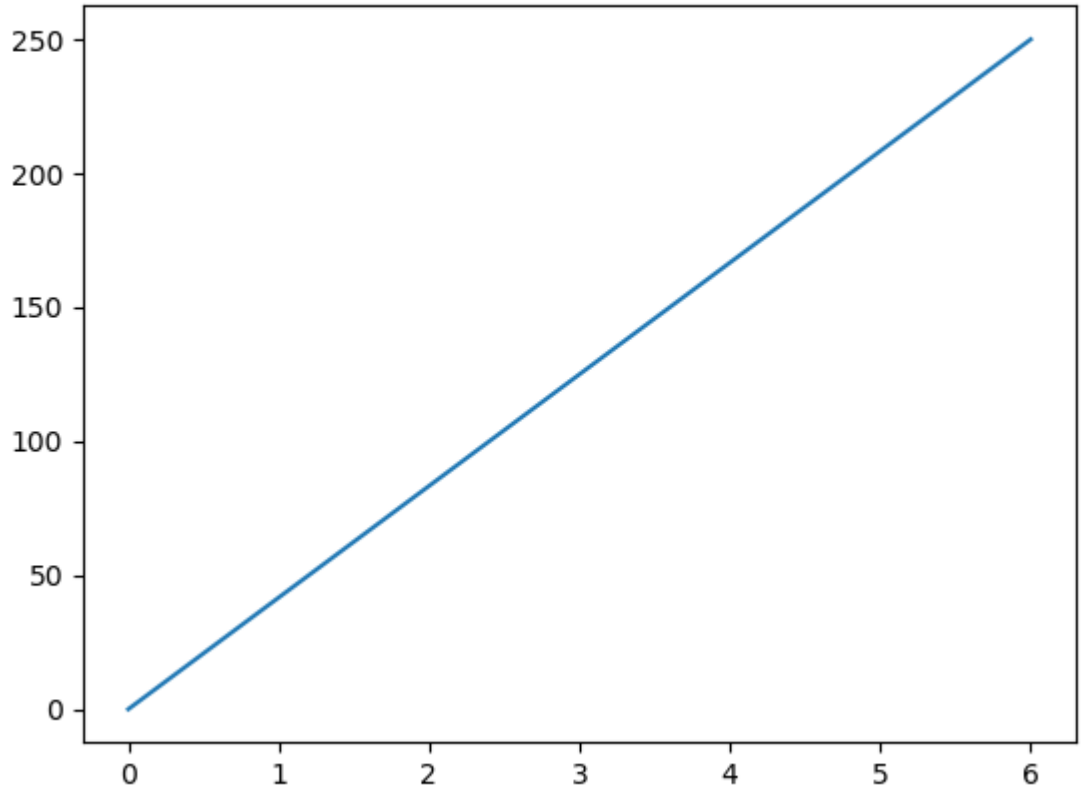
Matplotlib Pyplot

- 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()

Matplotlib Plyplot Example



Plotting x and y points

- The plot() function is used to draw points (markers) in a diagram.
- By default, the plot() function draws a line from point to point.
- The function takes parameters for specifying points in the diagram.
- Parameter 1 is an array containing the points on the **x-axis**.
- Parameter 2 is an array containing the points on the **y-axis**

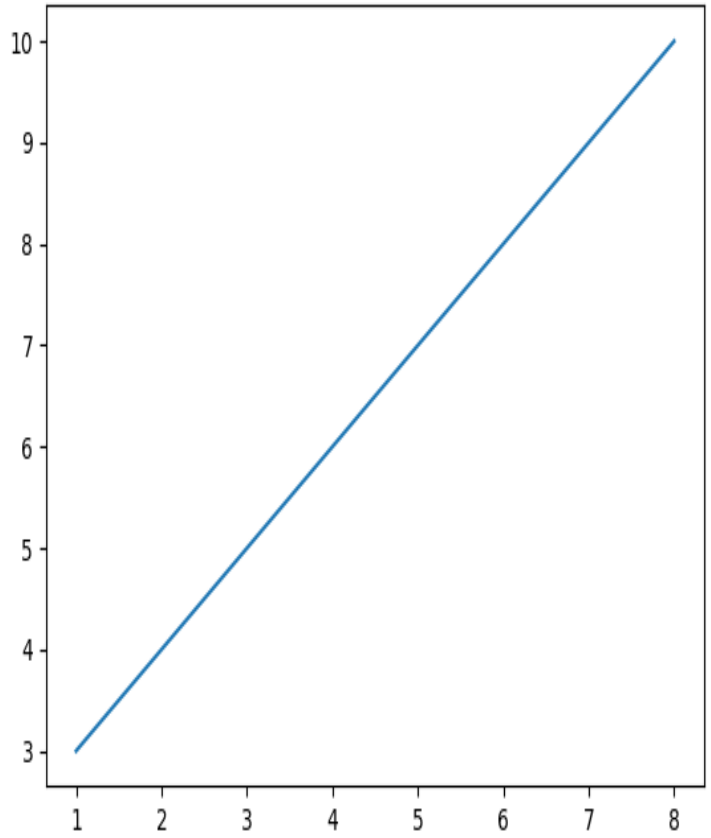
Draw a line in a diagram from position (1, 3) to position (8, 10):

- import matplotlib.pyplot as plt
  import numpy as np

  xpoints = np.array([1, 8])

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

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

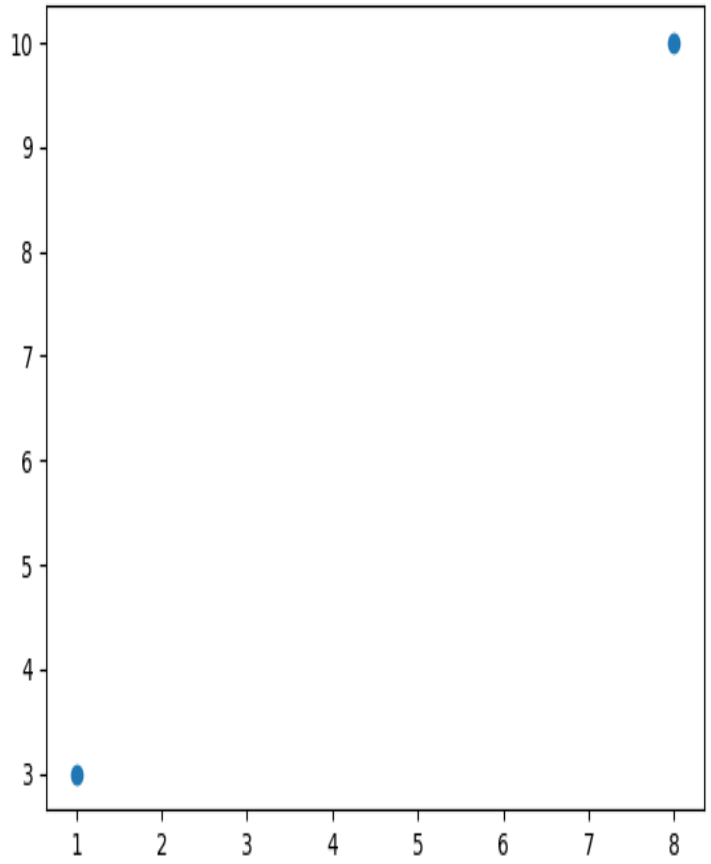


Plotting Without Line

- import matplotlib.pyplot a
  s plt
  import numpy as np

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

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

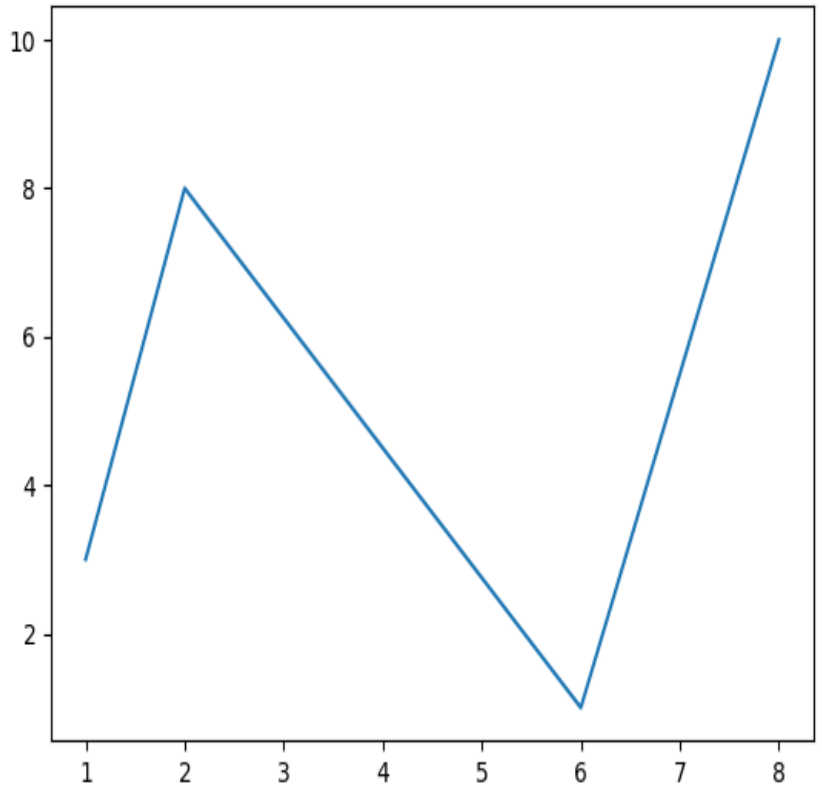


Multiple Points

- 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()

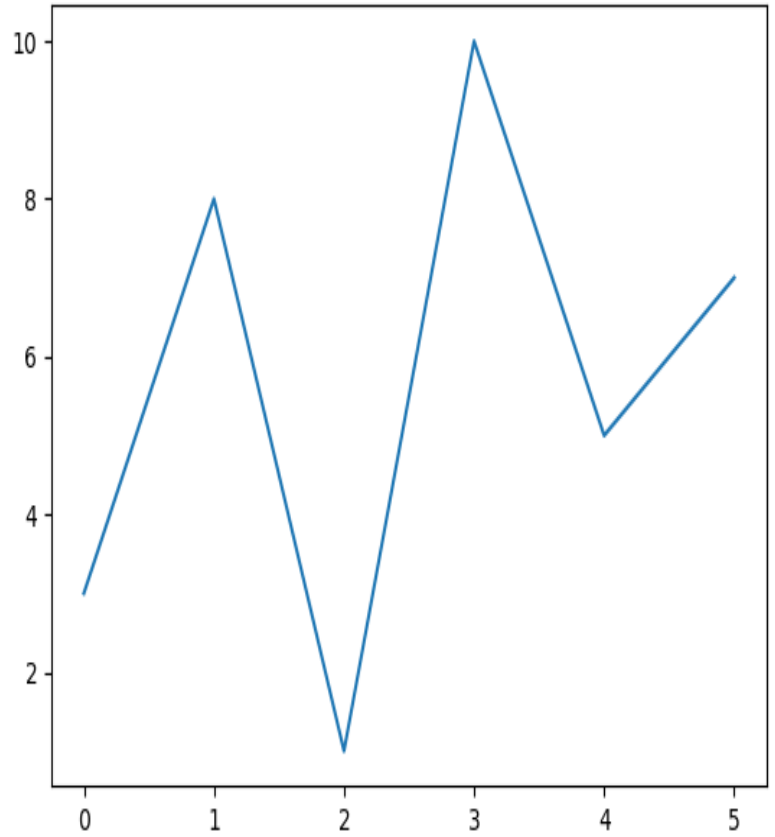


Default X-Points

- import matplotlib.pyplot as plt
  import numpy as np

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

  plt.plot(ypoints)
  plt.show()



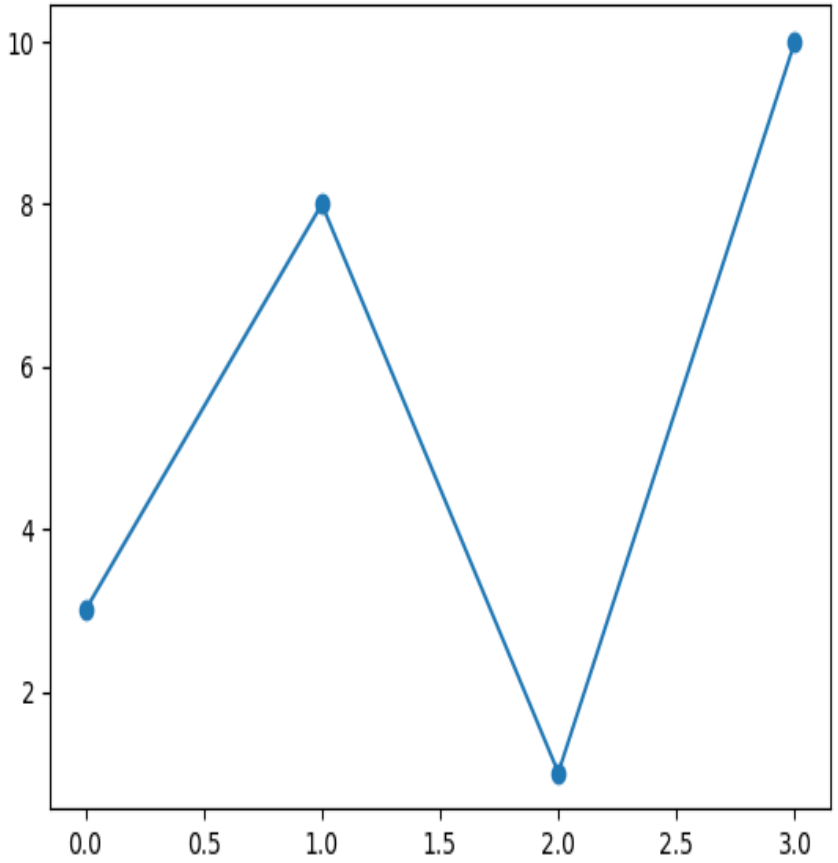
Matplotlib Markers

- Mark ech point as a **circle**

- import matplotlib.pyplot as plt
  import numpy as np

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

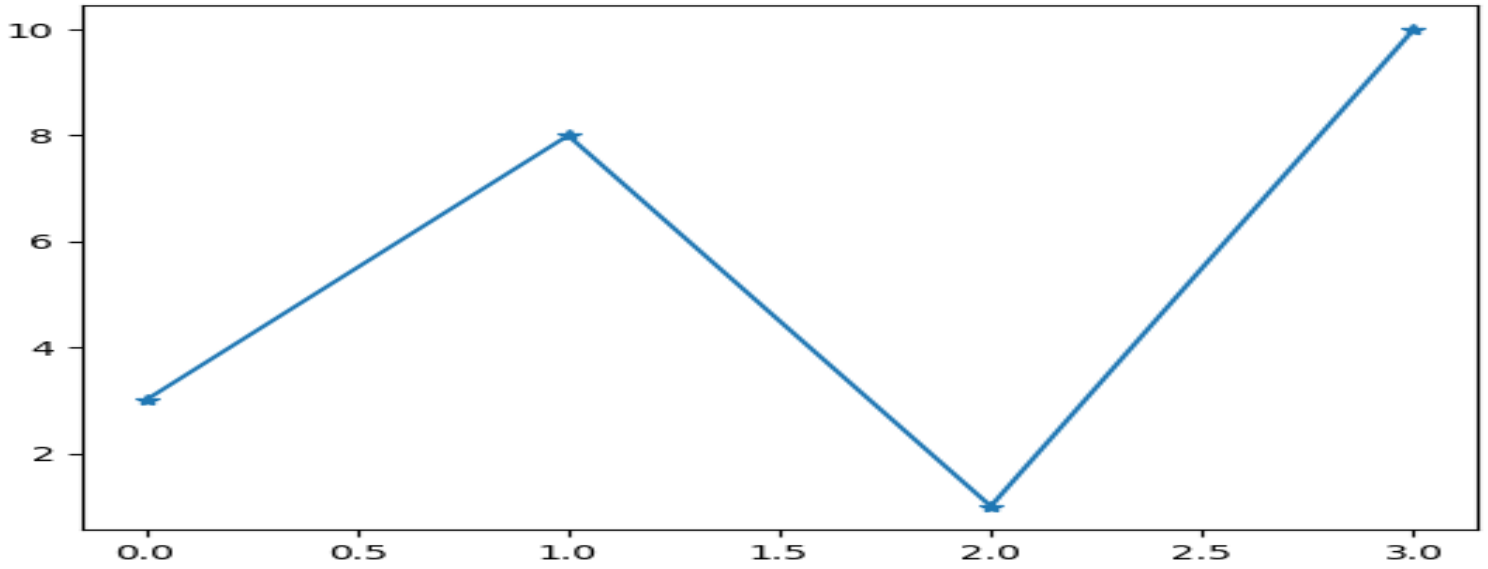
  plt.plot(ypoints, marker = 'o')
  plt.show()



Matplotlib Markers (contd.)

- Mark ech point as a **star**

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



Marker	Description
'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

- These are the different markers that can be used in different cases.

Format Strings fmt

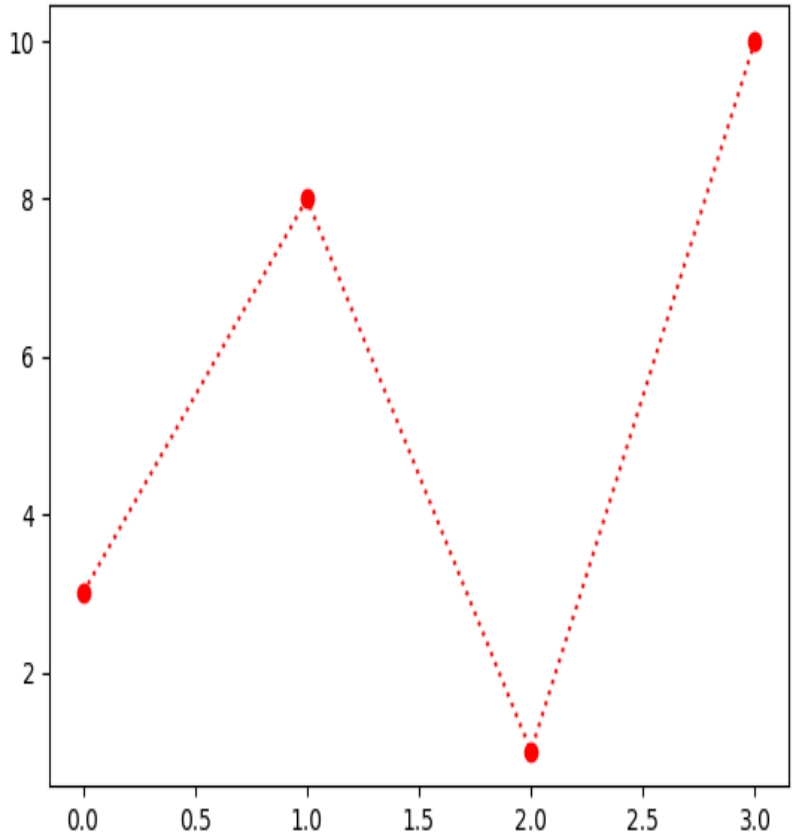
- This parameter is also called fmt, and is written with this syntax:

  marker|*line*|*color*

- import matplotlib.pyplot a
  s plt
  import numpy as np

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

  plt.plot(ypoints, 'o:r')
  plt.show()



Line Reference

Line Syntax	Description
'-'	Solid line
':'	Dotted line
'--'	Dashed line
'-.'	Dashed/dotted line

Color Reference

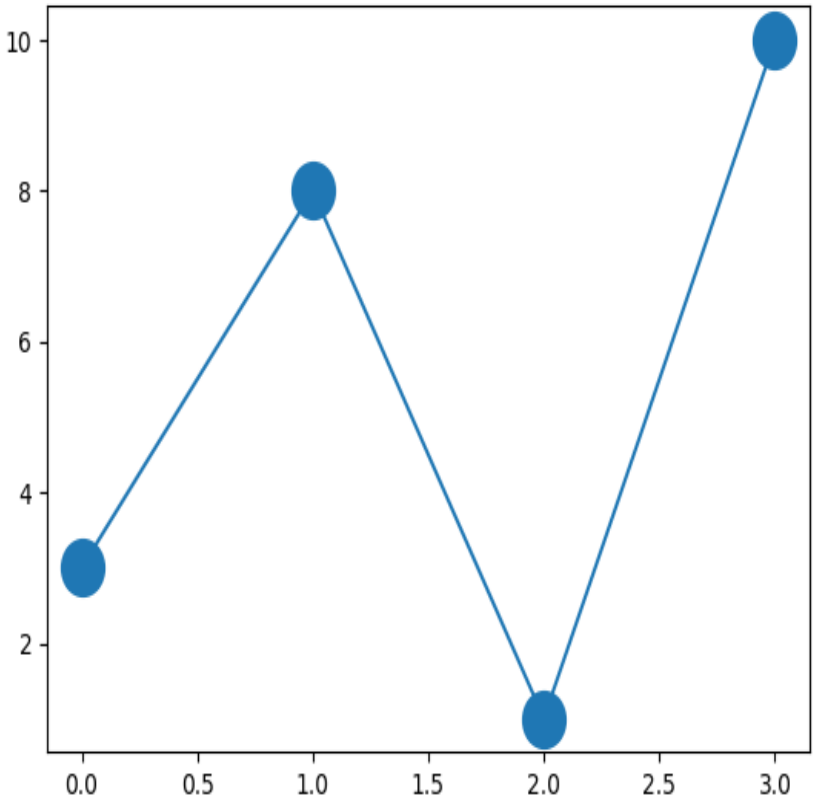
Color Syntax	Description
'r'	Red
'g'	Green
'b'	Blue
'c'	Cyan
'm'	Magenta
'y'	Yellow
'k'	Black
'w'	White

Marker Size

- import matplotlib.pyplot a
  s plt
  import numpy as np

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

  plt.plot(ypoints, marker
  = 'o', ms = 20)
  plt.show()

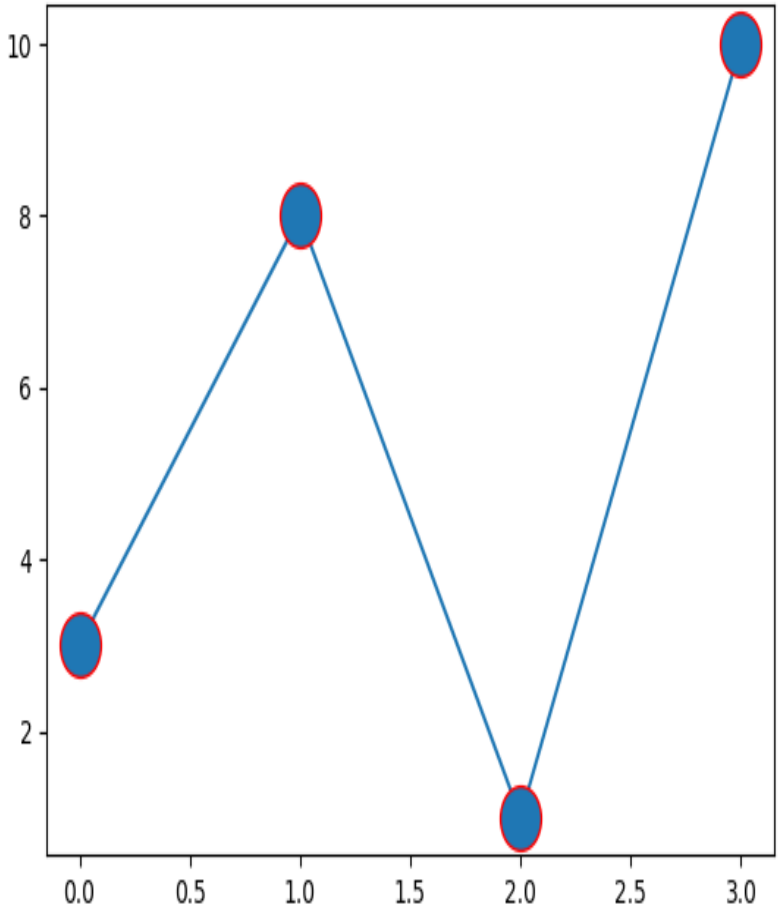


Marker Color

- 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')
  plt.show()

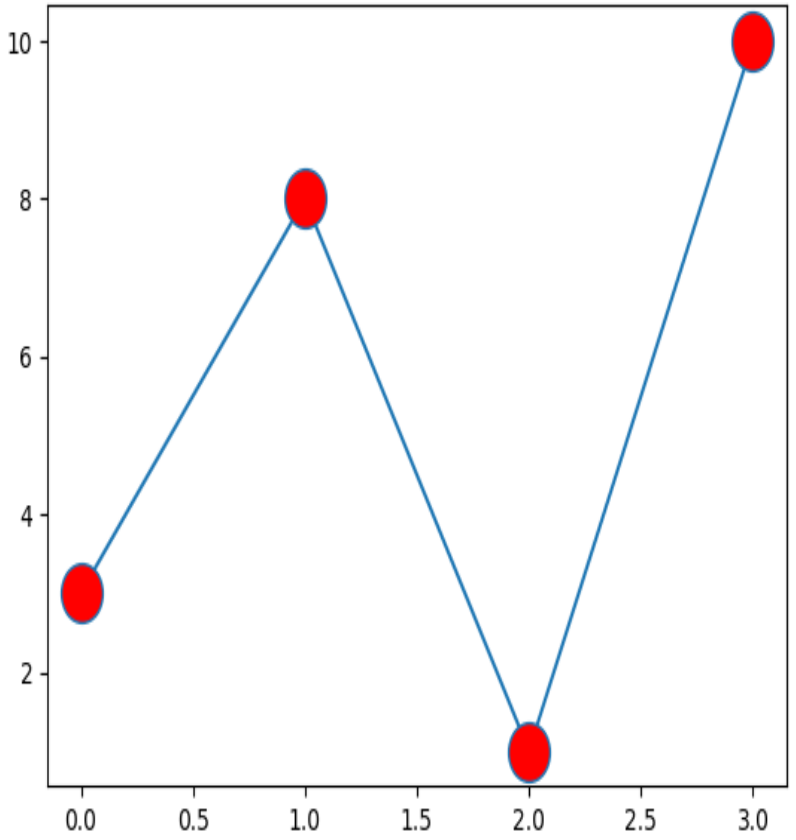


Set the FACE color to red:

- import matplotlib.pyplot as plt
  import numpy as np

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

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

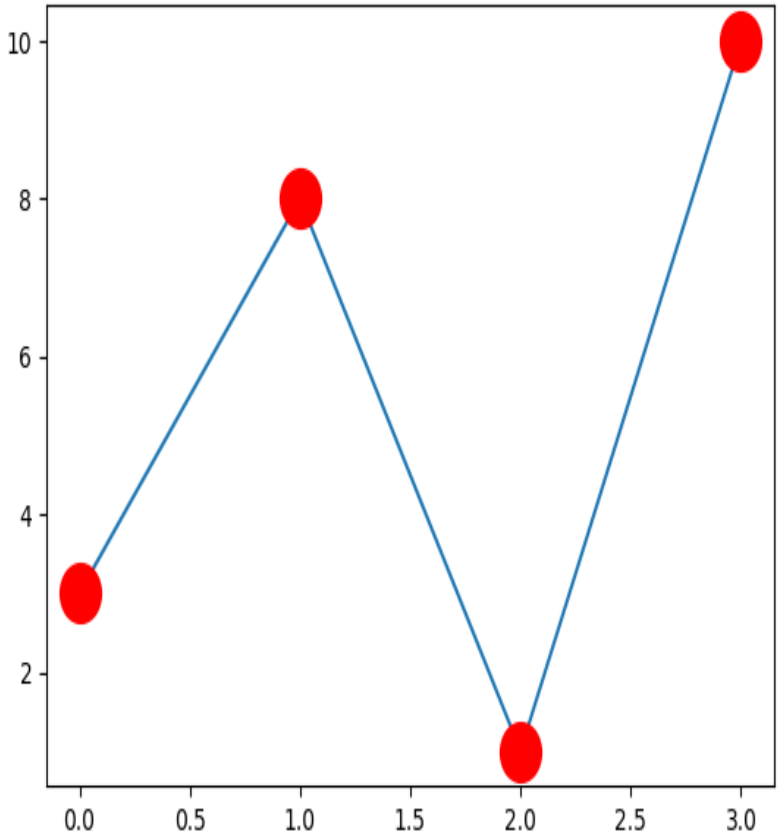


Set the color of both the *edge* and the *face* to red:

- 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()

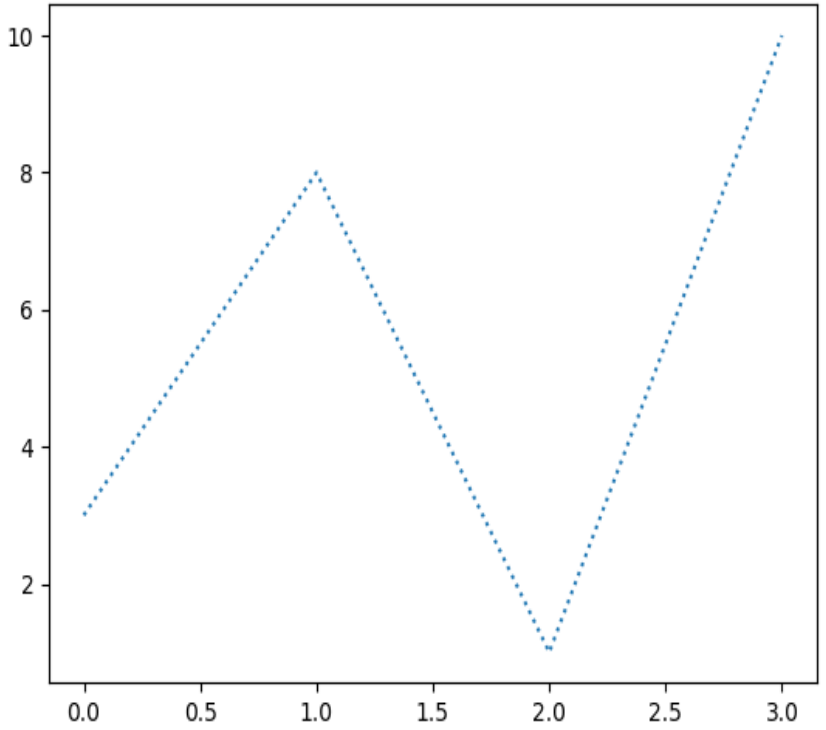


Matplotlib Line

- import matplotlib.pyplot as plt
  import numpy as np

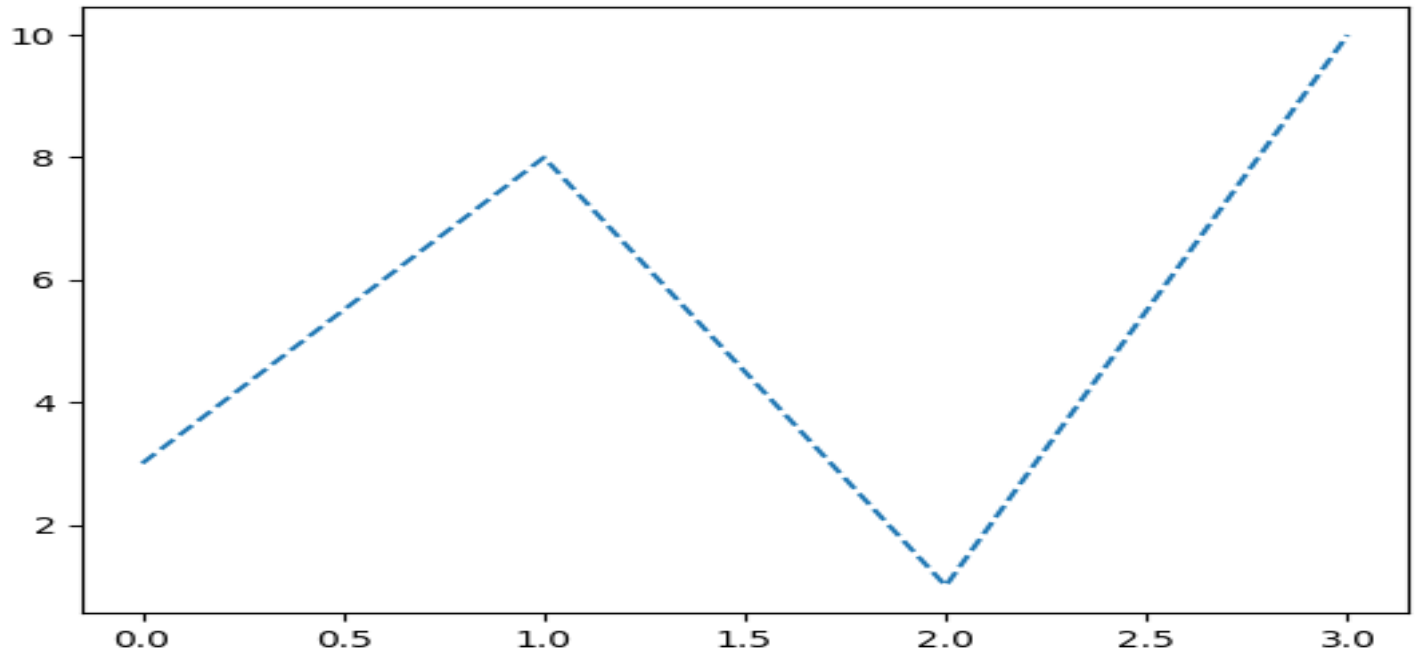
  ypoints = np.array([3, 8, 1, 10])

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



Line Style

- plt.plot(ypoints, linestyle = 'dashed')

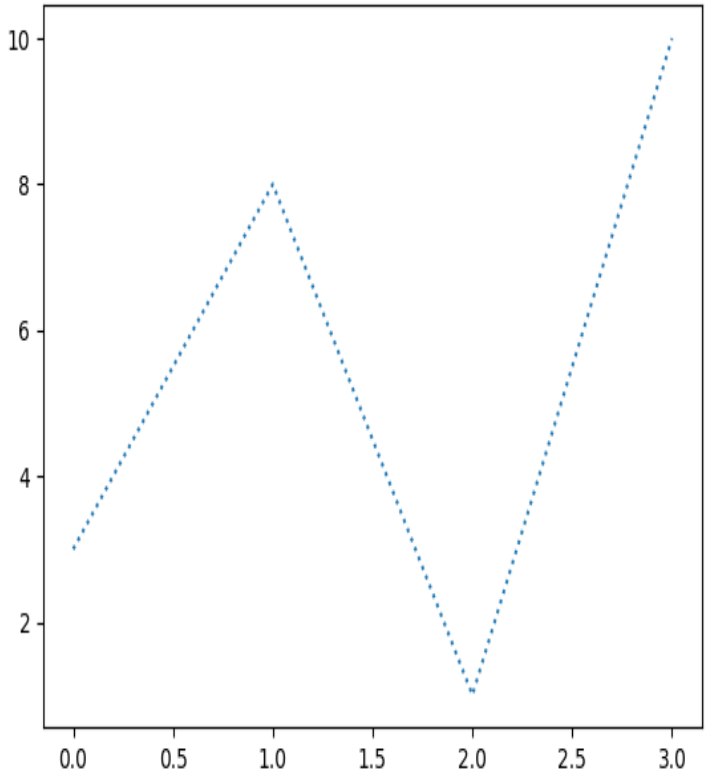


Line Style (contd.)

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 --.

- plt.plot(ypoints, ls = ':')



Line Style (contd.)

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

Line Color

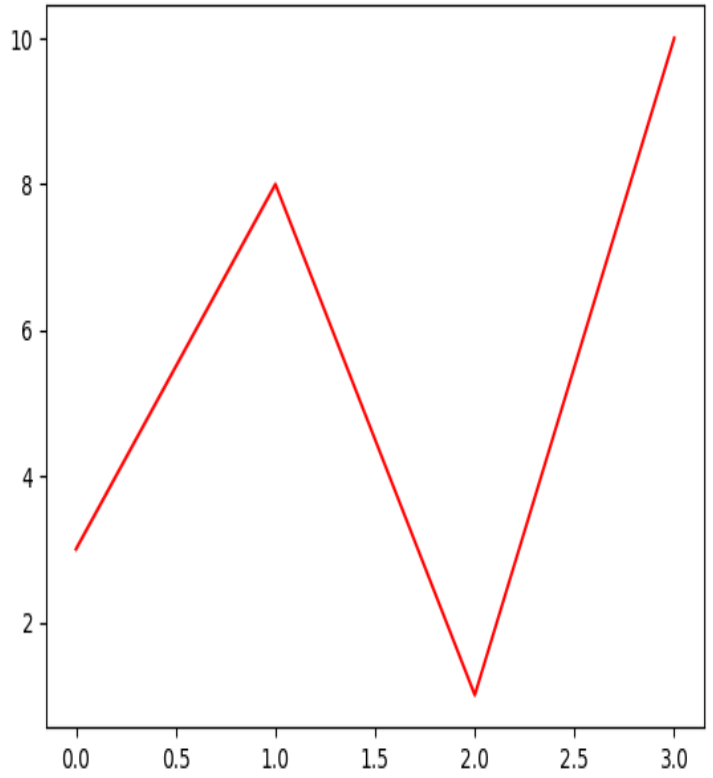
- Example

- import matplotlib.pyplot as plt
  import numpy as np

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

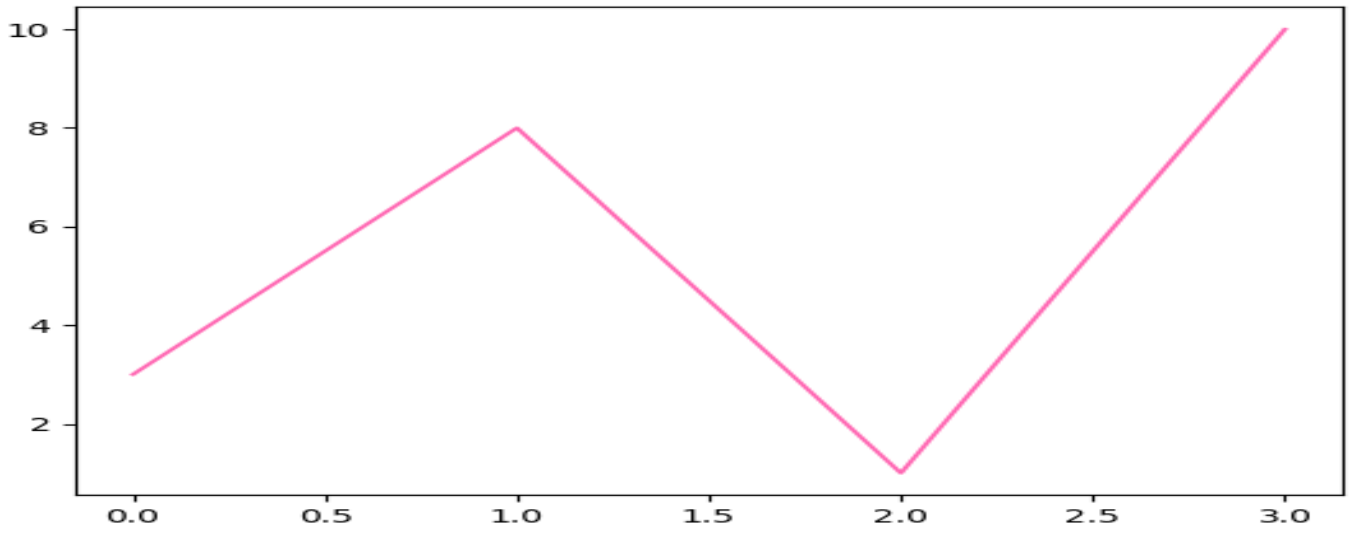
  plt.plot(ypoints, color = 'r')
  plt.show()

- Another example
- ...
  plt.plot(ypoints, c = '#4CAF50')
  ...
- Graph will be greenish in color



Line Color (Contd.)

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



Line Width

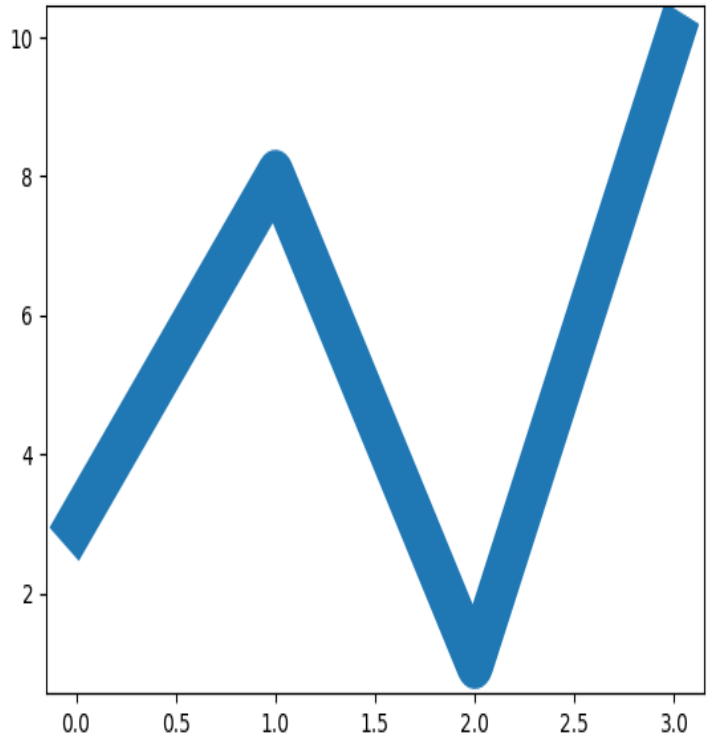
- You can use the keyword argument linewidth or the shorter lw to change the width of the line.
- The value is a floating number, in points:

Line Width example

- import matplotlib.pyplot as plt
  import numpy as np

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

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



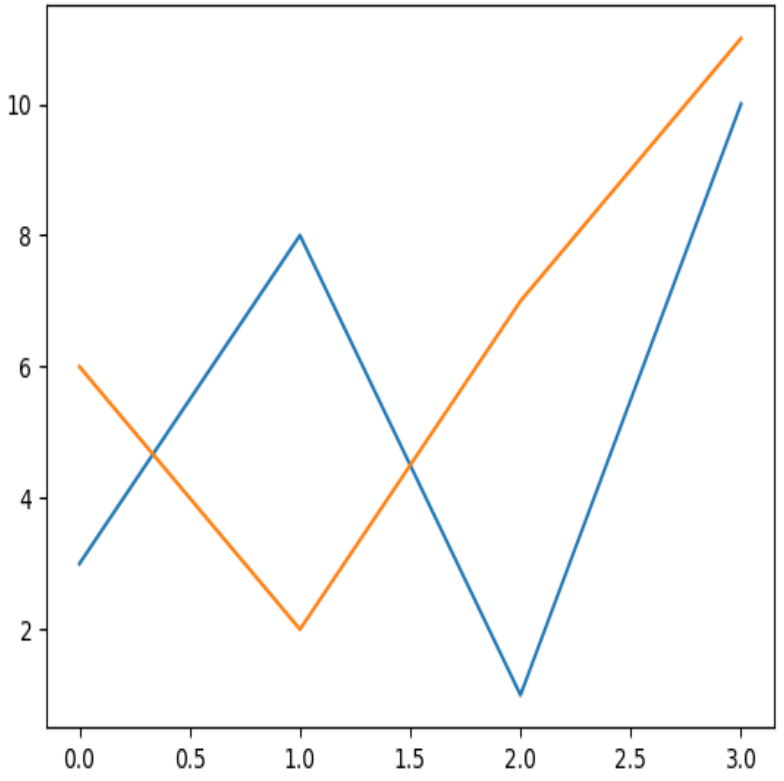
Multiple Lines

- import matplotlib.pyplot as plt
  import numpy as np

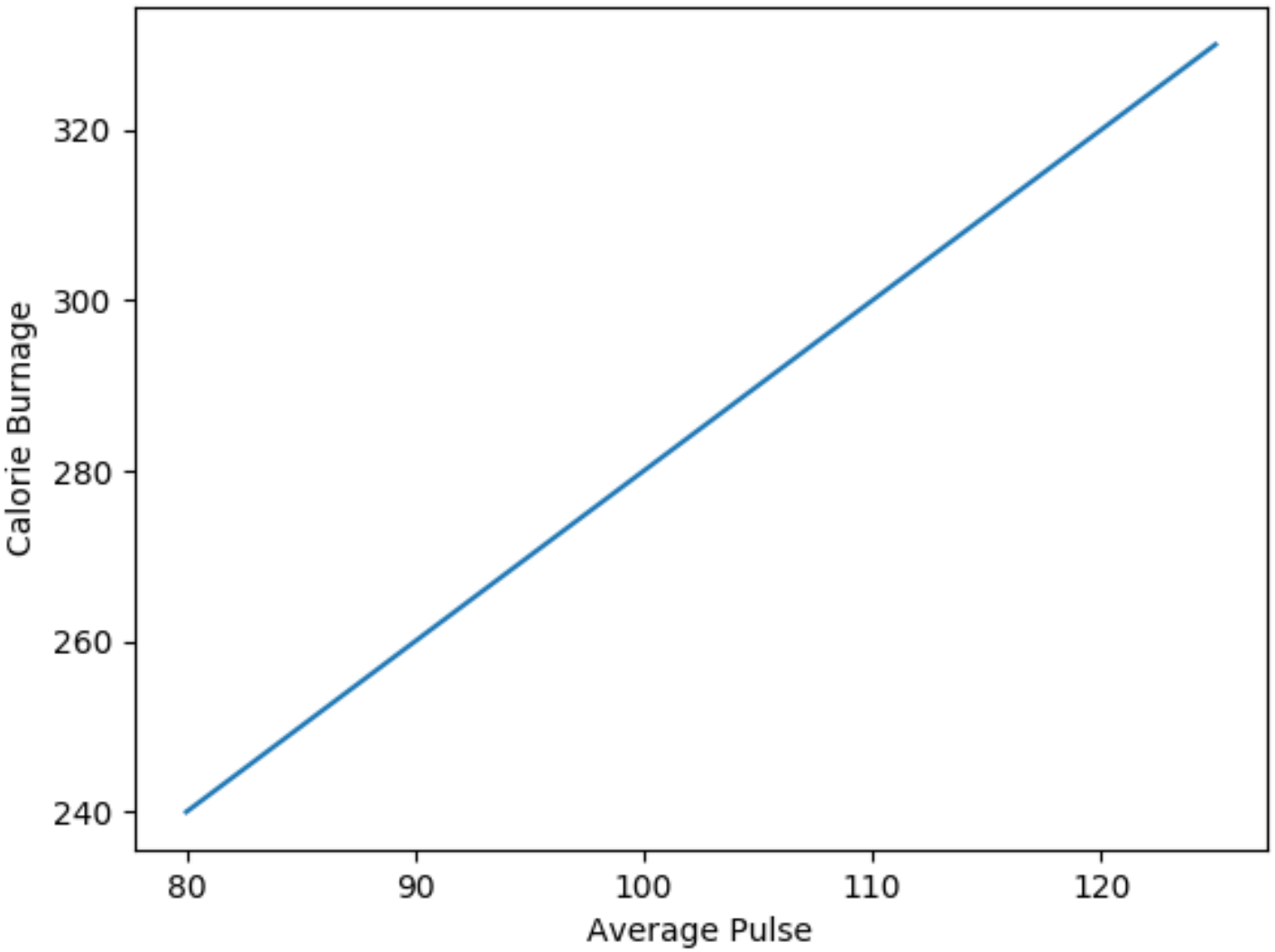
  y1 = np.array([3, 8, 1, 10])
  y2 = np.array([6, 2, 7, 11])

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

  plt.show()



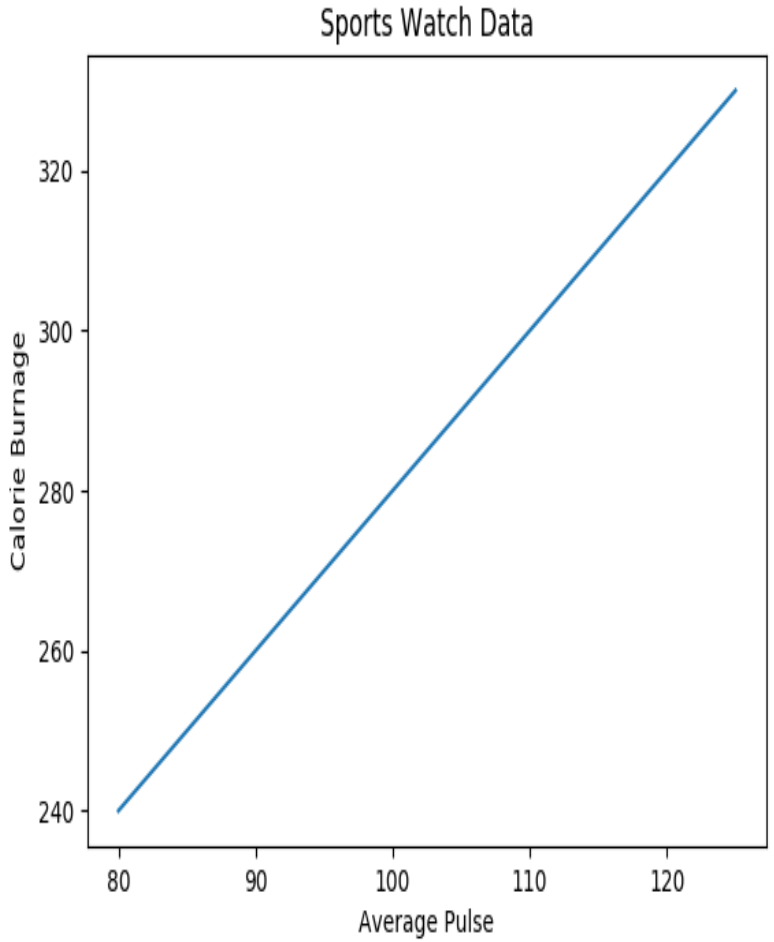
Matplotlib Labels and Title

- ```
  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, 3
  30])

  plt.plot(x, y)

  plt.xlabel("Average Pulse")
  plt.ylabel("Calorie Burnage")

  plt.show()
  ```
  ```


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

Titles and Labels

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

  x =
  np.array([80, 85, 90, 95, 10
  0, 105, 110, 115, 120, 125])
  y =
  np.array([240, 250, 260, 27
  0, 280, 290, 300, 310, 320,
  330])

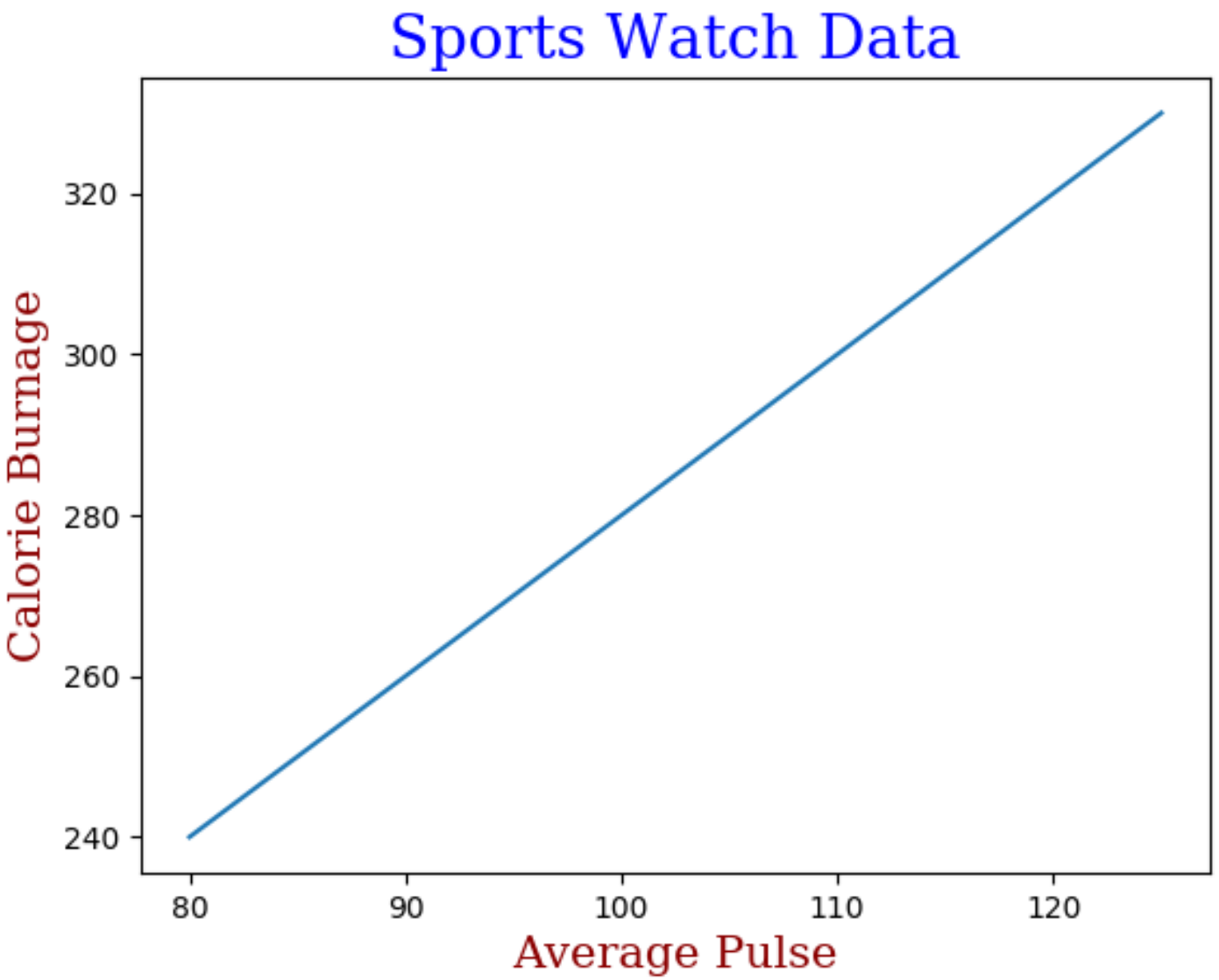
  plt.plot(x, y)

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

  plt.show()
  ```


  ```

Set Font Properties for Title and Labels

```
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()
```


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

Position the Title

- ```
  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()
  ```
  ```

Matplotlib Adding Grid Lines

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

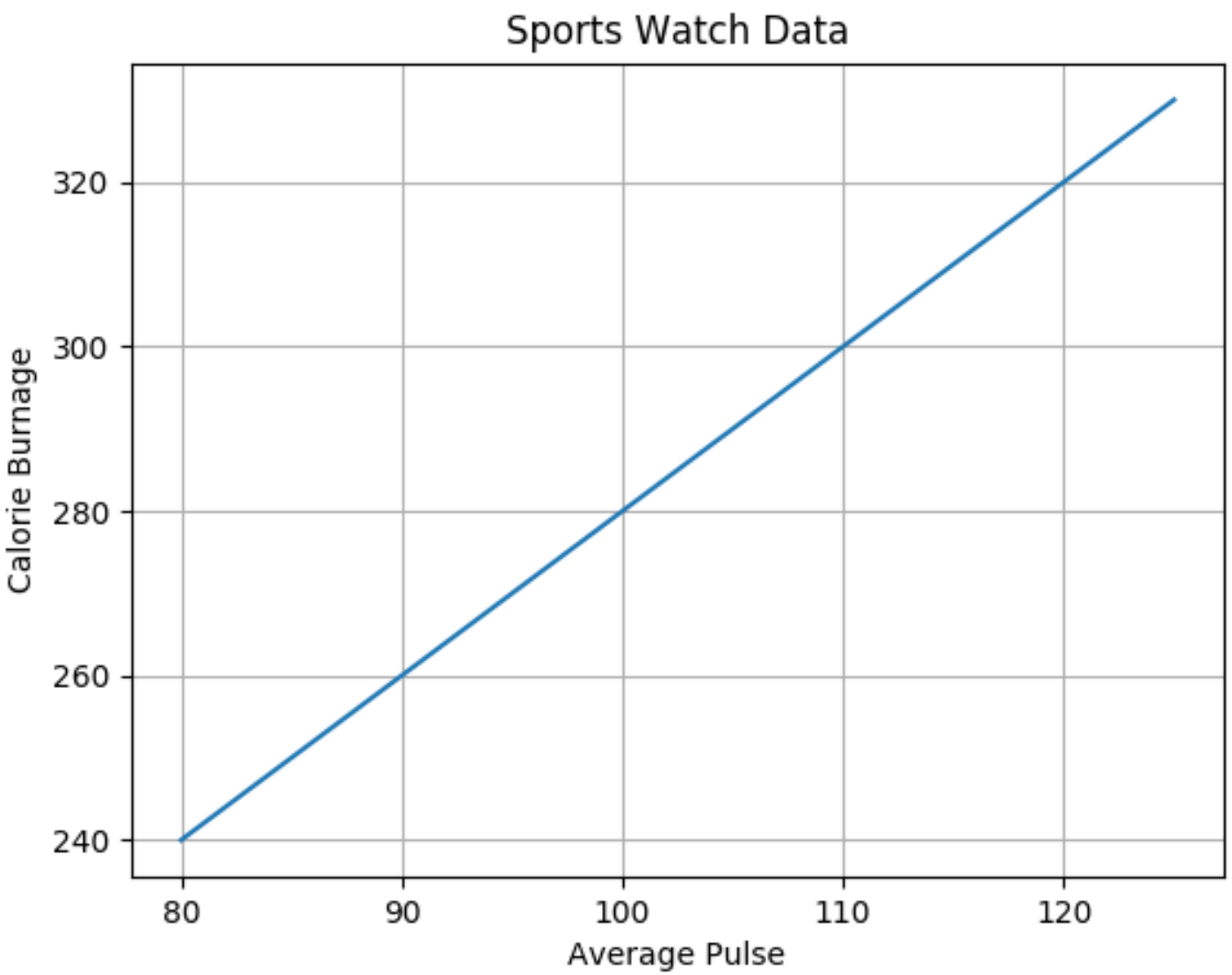
  x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
  y =
  np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

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

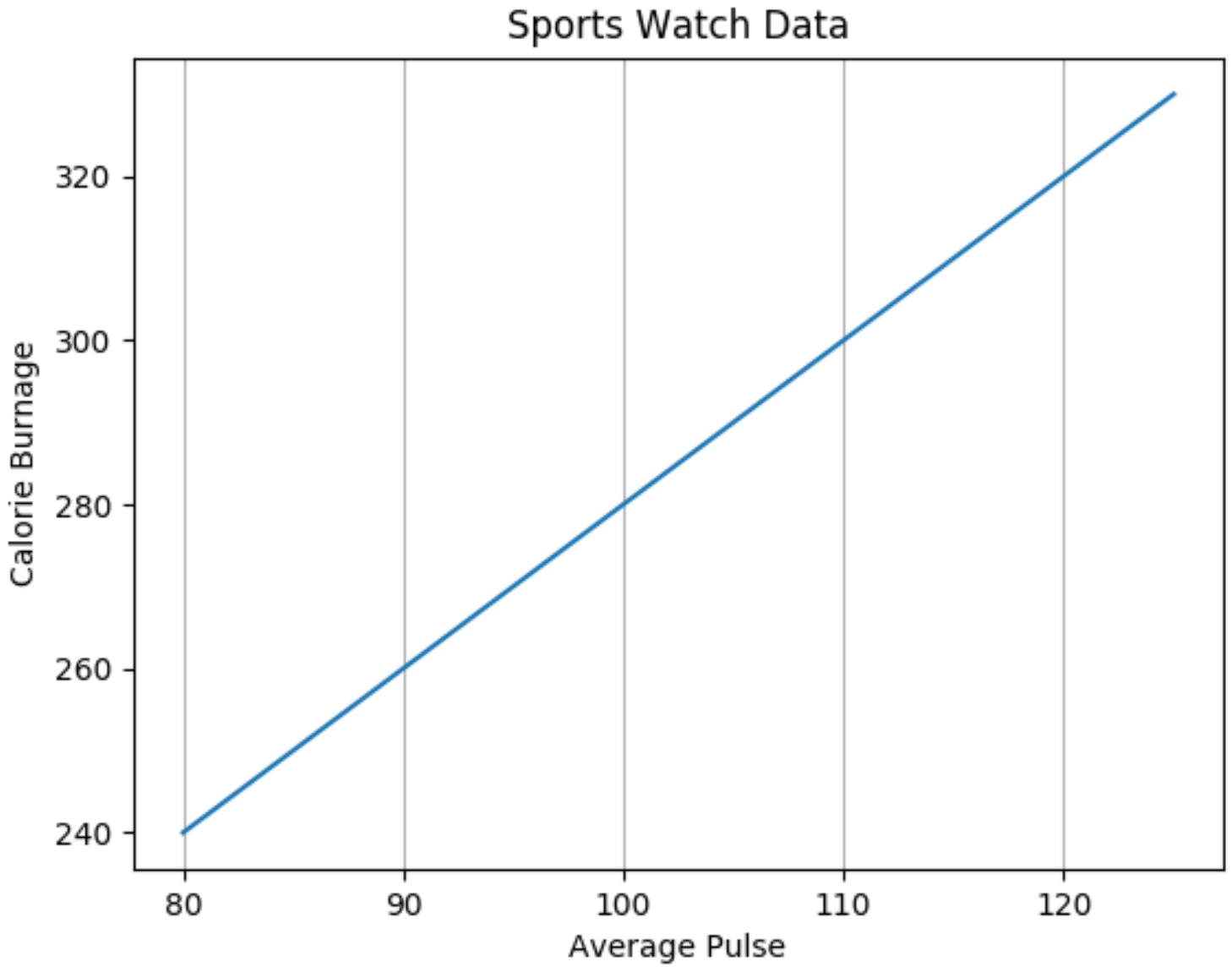
  plt.plot(x, y)

  plt.grid()

  plt.show()
  ```
  ```


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

Specify Which Grid Lines to Display

- ```
  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()
  ```
  ```


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

Set Line Properties for the Grid

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

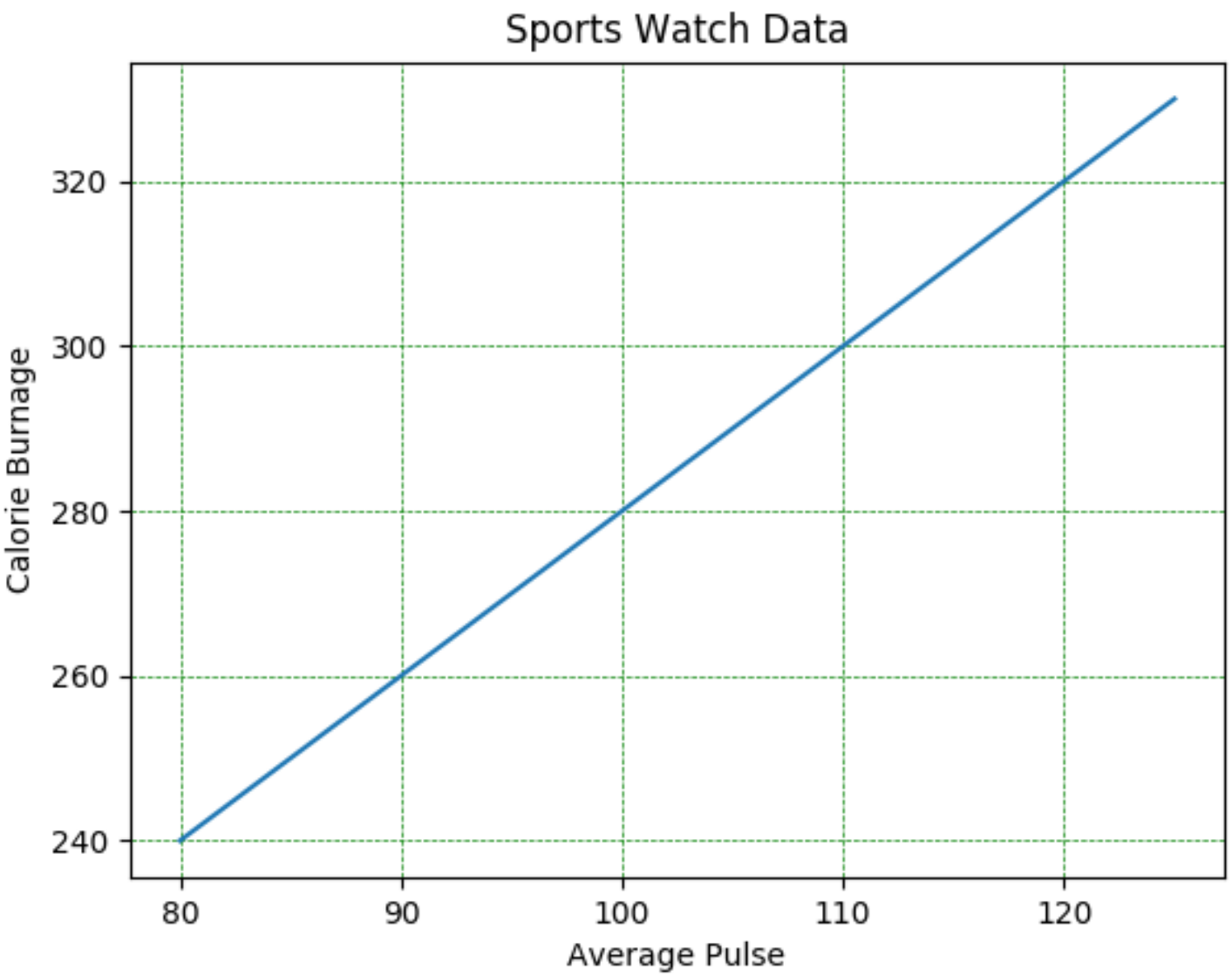
  x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
  y =
  np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])

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

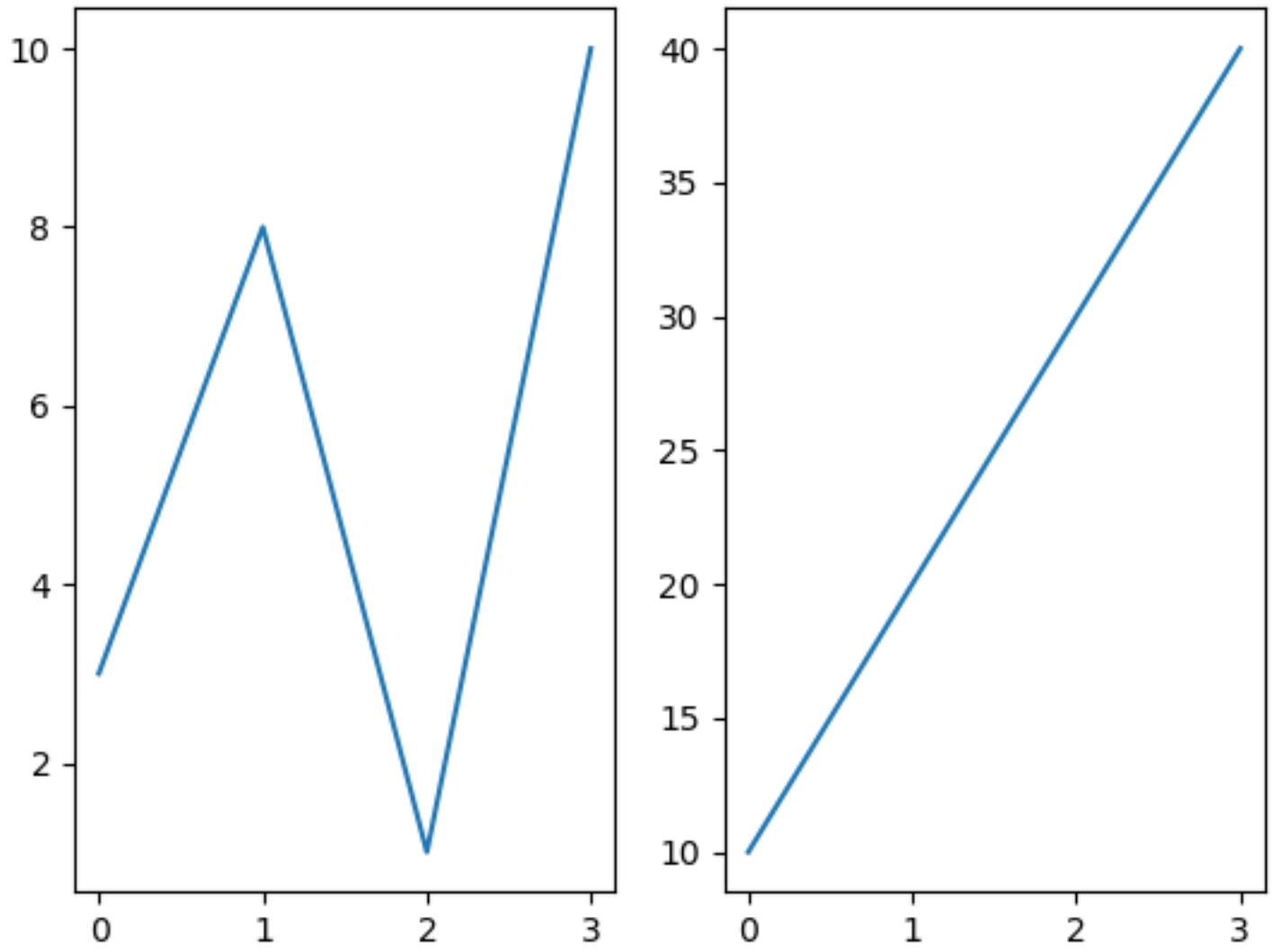
  plt.plot(x, y)

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

  plt.show()
  ```
  ```


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

Matplotlib Subplot

- ```
  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()
  ```
  ```


10
8
6
4
2
0
1
2
3
40
35
30
25
20
15
10
0
1
2
3

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.

Draw 2 plots on top of each other:

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

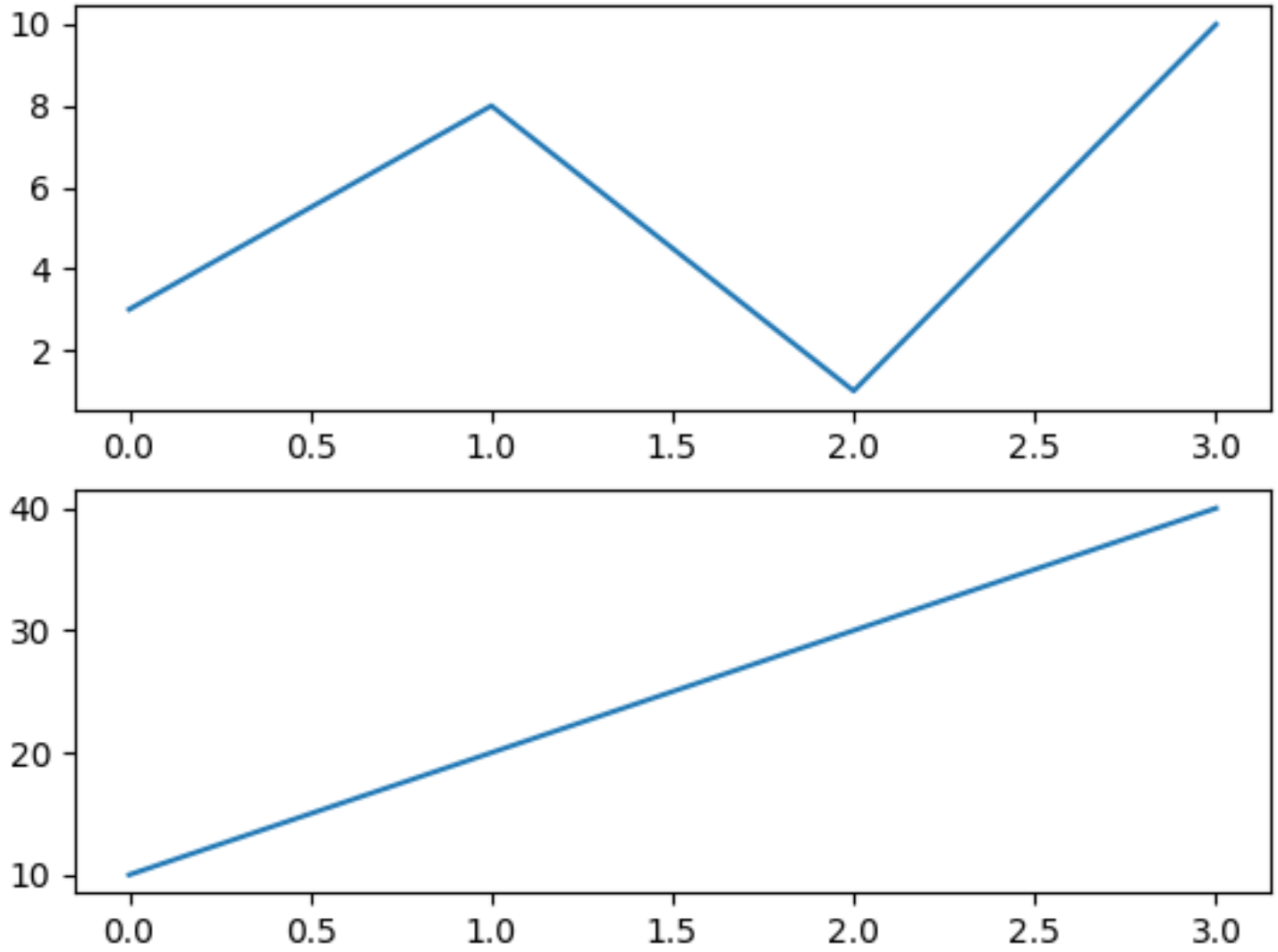
  #plot 1:
  x = np.array([0, 1, 2, 3])
  y = np.array([3, 8, 1, 10])

  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])

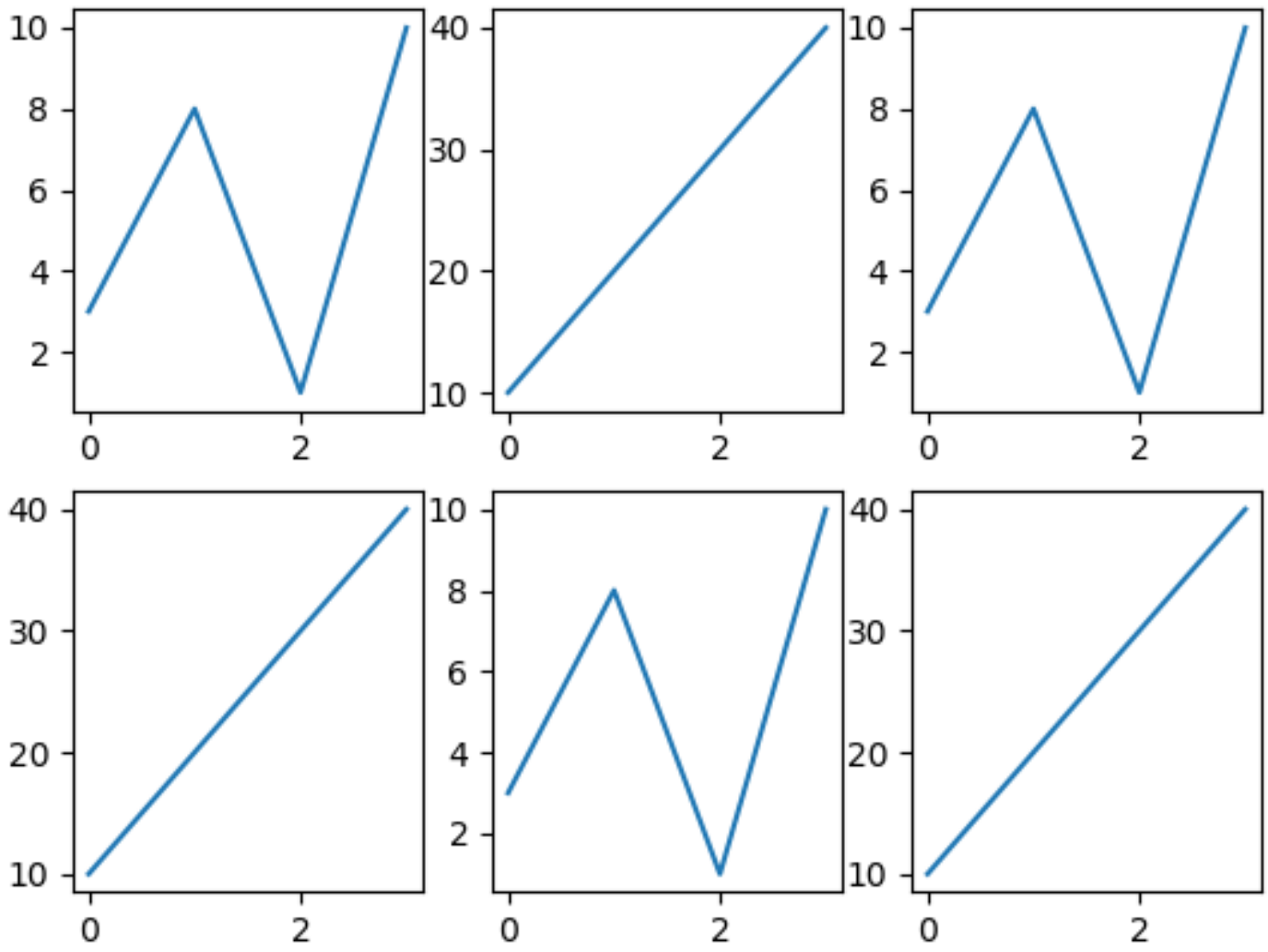
  plt.subplot(2, 1, 2)
  plt.plot(x,y)

  plt.show()
  ```
  ```

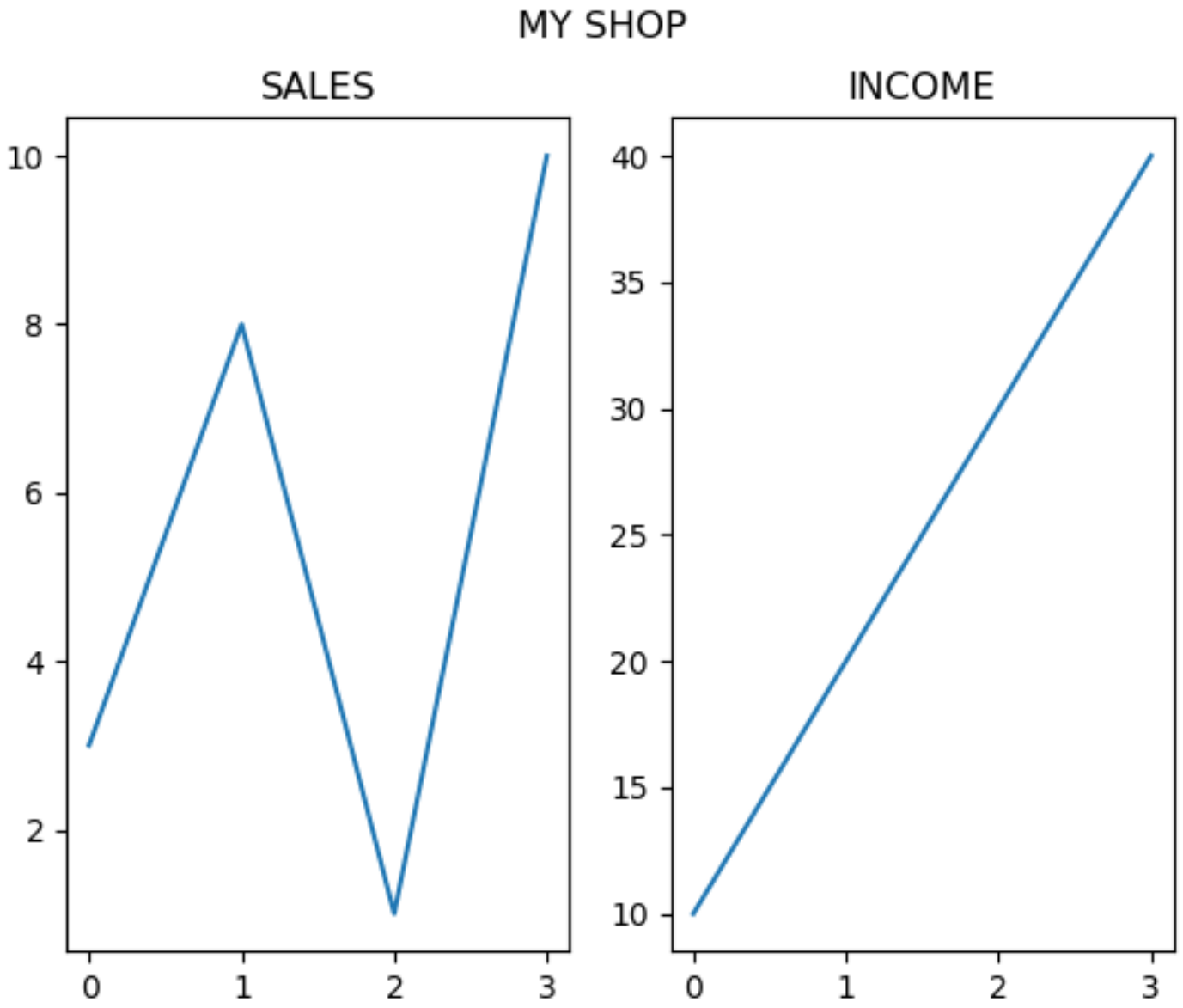
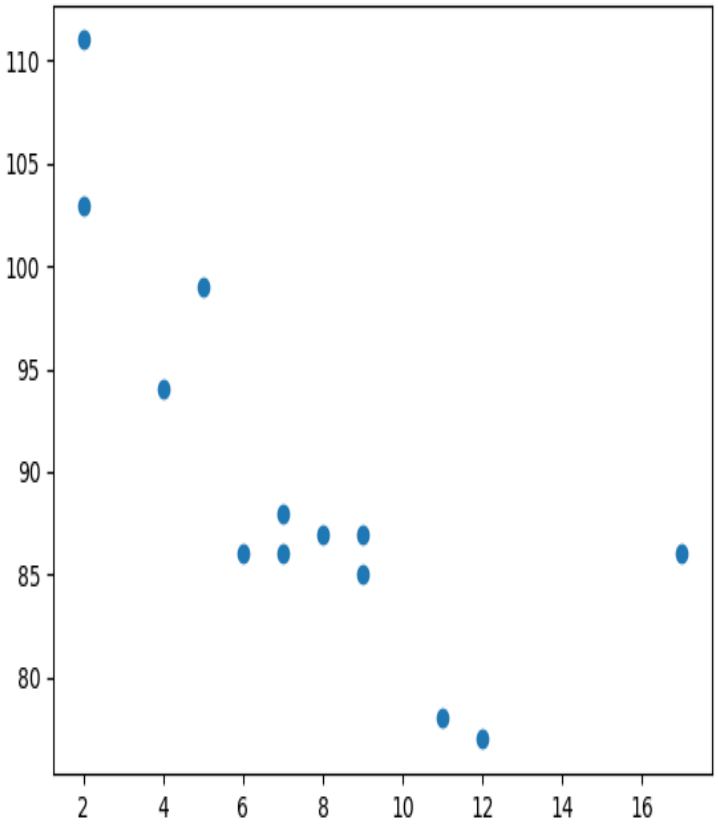

10
8
6
4
2
0.0
0.5
1.0
1.5
2.0
2.5
3.0
40
30
20
10
0.0
0.5
1.0
1.5
2.0
2.5
3.0

Drawing 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()
```


10
8
6
4
2
0
2
40
30
20
10
0
2
10
8
6
4
2
0
2
40
30
20
10
0
2
10
8
6
4
2
0
2
40
30
20
10
0
2

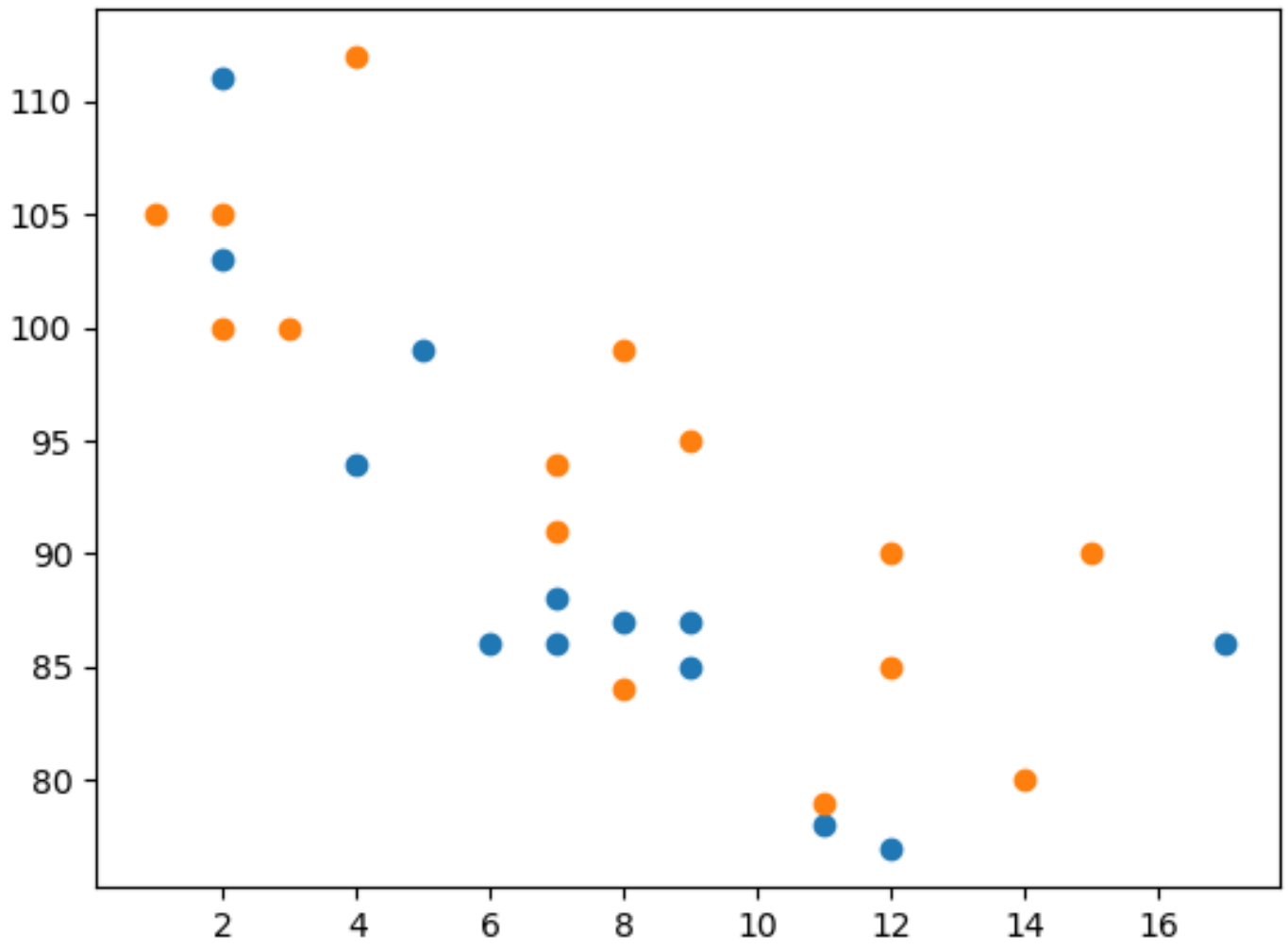
Add a title for the entire figure:

- ```
  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()
  ```
  ```


MY SHOP
SALES
INCOME
10
8
6
4
2
0
1
2
3
40
35
30
25
20
15
10
0
1
2
3

Matplotlib Scatter

- import matplotlib.pyplot a
  s 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
  ,86,103,87,94,78,77,85,8
  6])

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



Draw two plots on the same figure:

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

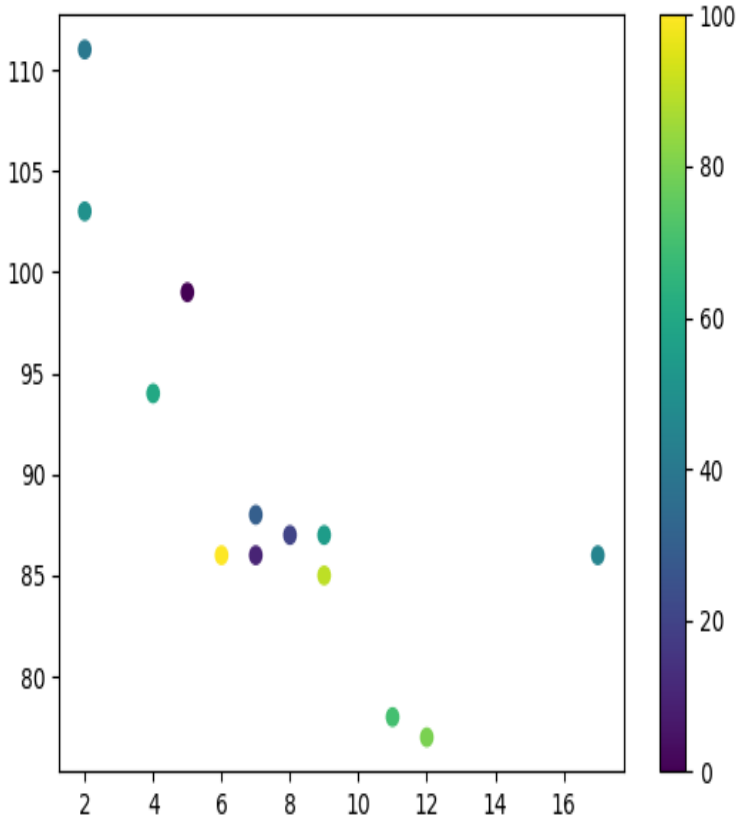
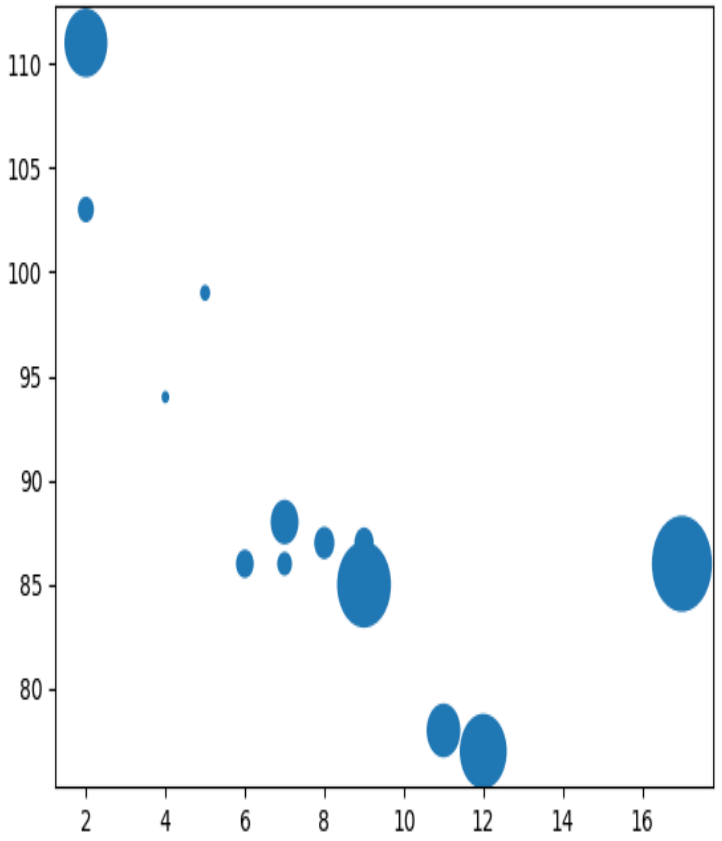
  #day one, the age and speed of 13 cars:
  x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
  y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
  plt.scatter(x, y)

  #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([100,105,84,105,90,99,90,95,94,100,79,112,91,80,8
  5])
  plt.scatter(x, y)

  plt.show()
  ```
  ```


110
105
100
95
90
85
80
2
4
6
8
10
12
14
16

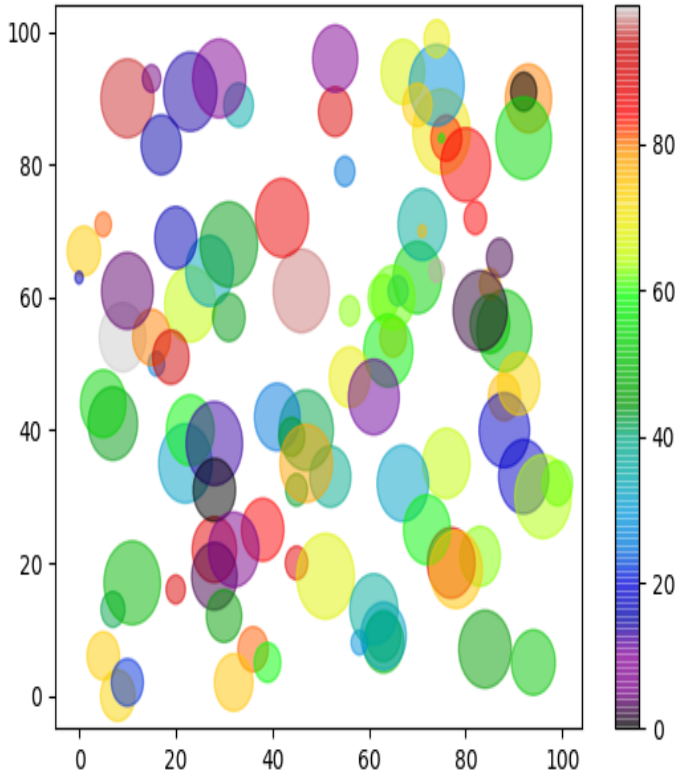
- ```
  import matplotlib.pyplot as plt
  import numpy as np

  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])

  plt.scatter(x, y, c=colors,
  cmap='viridis')

  plt.colorbar()

  plt.show()
  ```


  ```

Set your own size for the markers:

- 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([99,86,87,88,111,86,1
  03,87,94,78,77,85,86])
  sizes
  = np.array([20,50,100,200,500
  ,1000,60,90,10,300,600,800,7
  5])

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

  plt.show()



Combine Color Size and Alpha

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

  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()
  ```


  ```

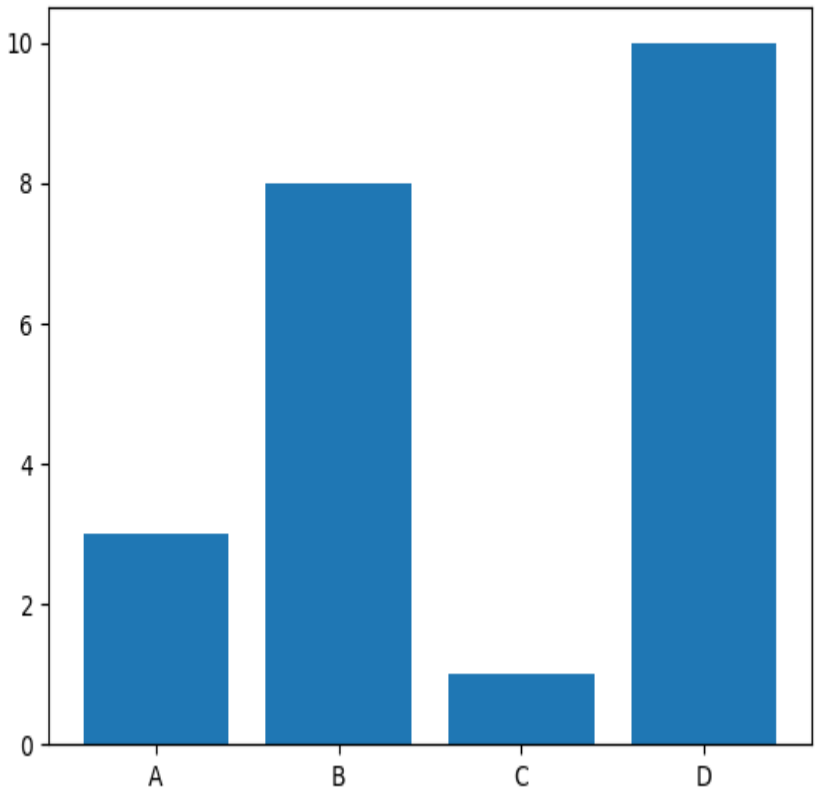
Matplotlib Bars

- import matplotlib.pyplot a
  s plt
  import numpy as np

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

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

Result:

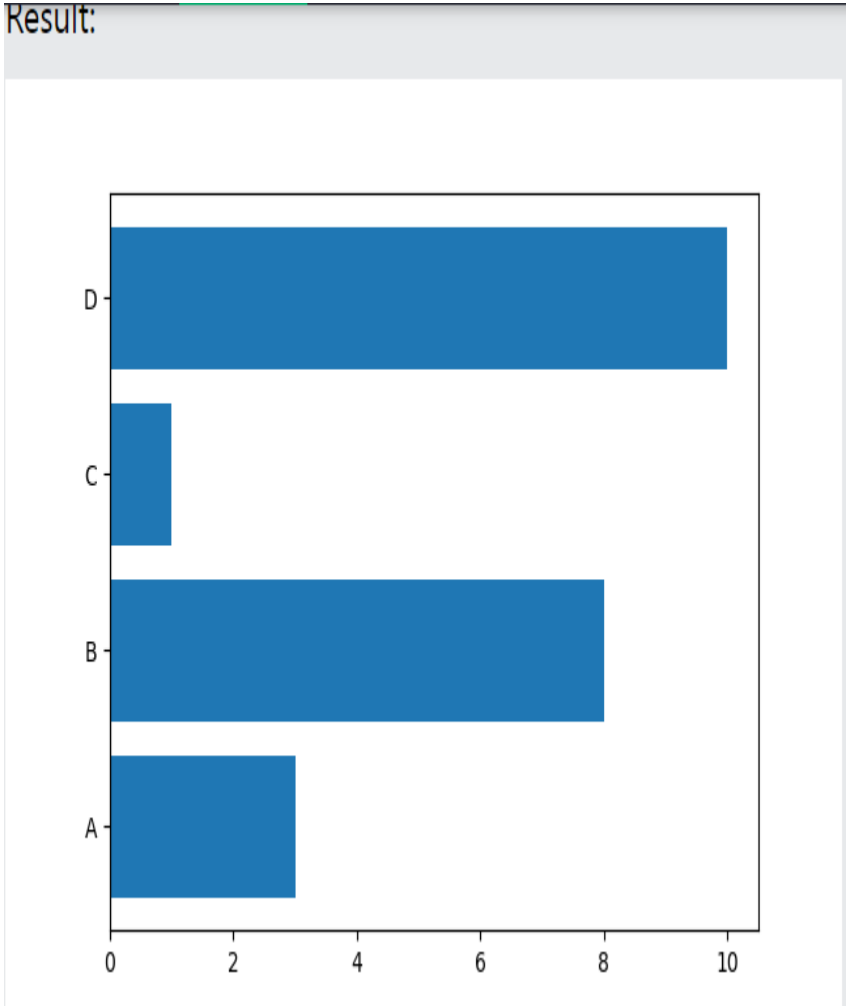


Horizontal Bars

- 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)
  plt.show()



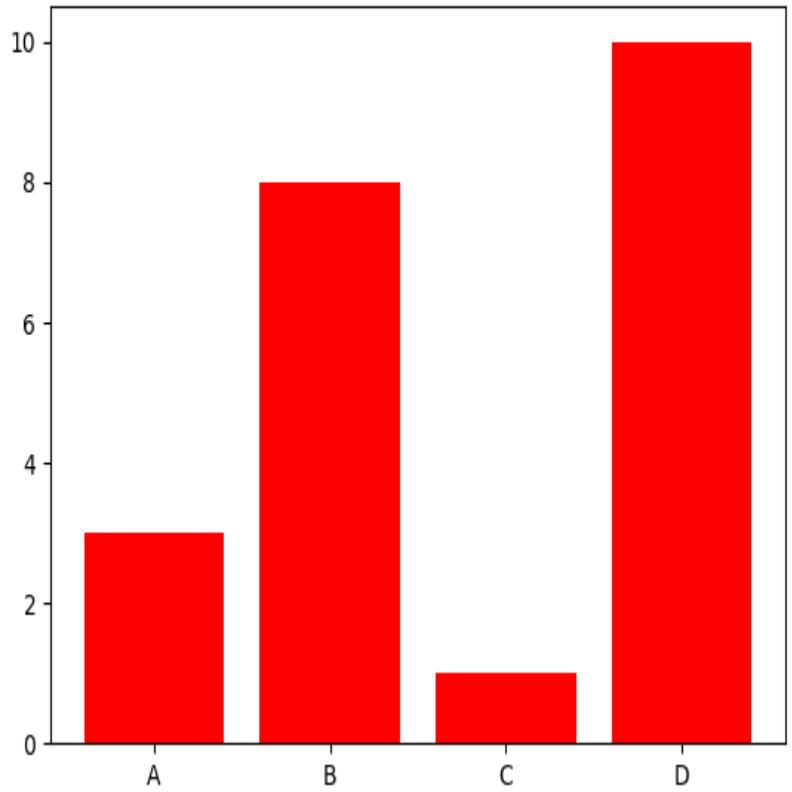
Bar Color

- 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, color = "red")
  plt.show()

Result:

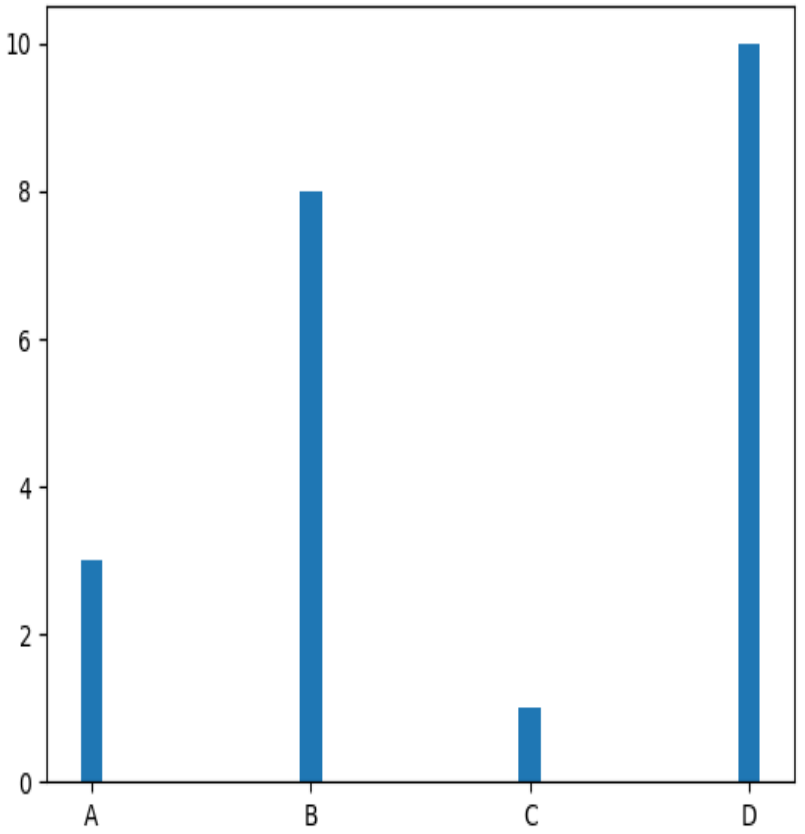


Bar Width

- 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, width = 0.1)
  plt.show()

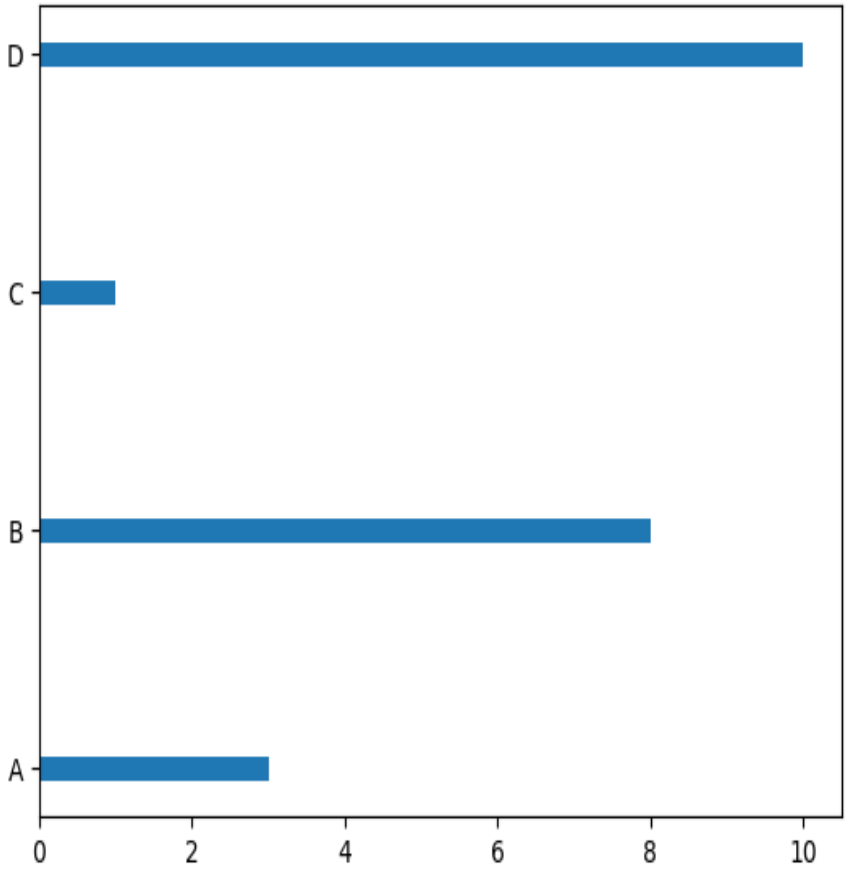


Bar Height & Width

- import matplotlib.pyplot as plt
  import numpy as np

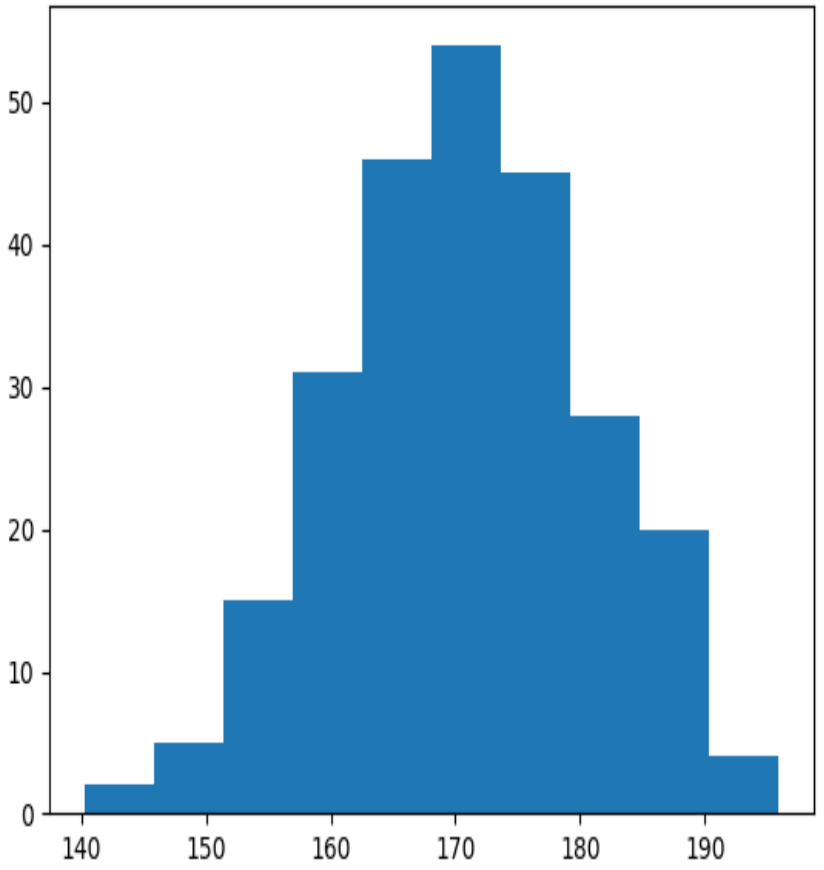
  x = np.array(["A", "B", "C", "D"])
  y = np.array([3, 8, 1, 10])

  plt.barh(x, y, height = 0.1)
  plt.show()



Matplotlib Histograms

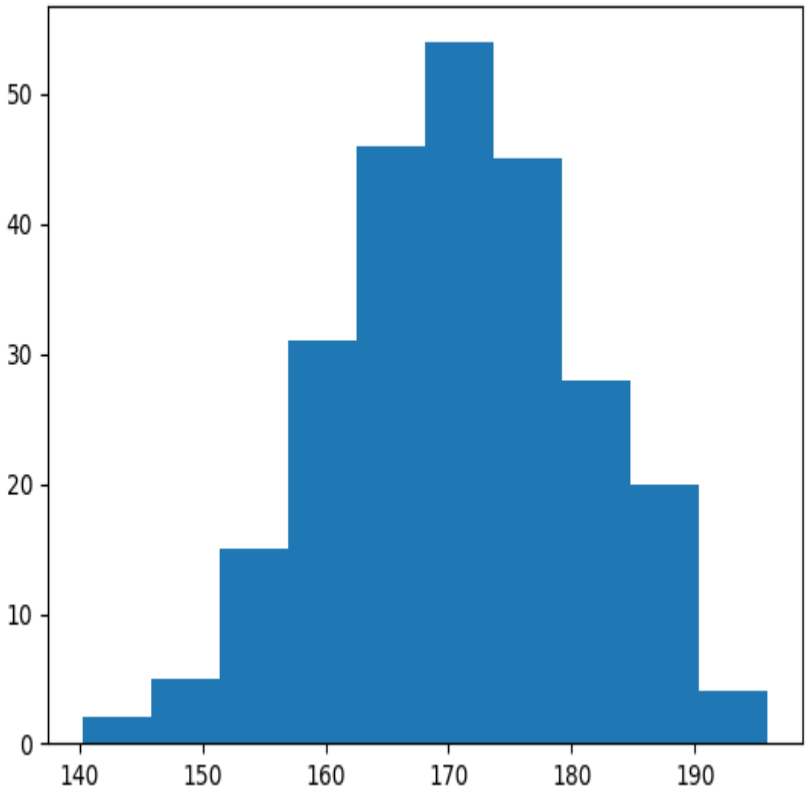
- A histogram is a graph showing *frequency* distributions .
- It is a graph showing the number of observations within each given interval.
- Example: Say you ask for the height of 250 people, you might end up with a histogram like this:



- import matplotlib.pyplot a
  s plt
  import numpy as np

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

  plt.hist(x)
  plt.show()



Matplotlib Pie Charts

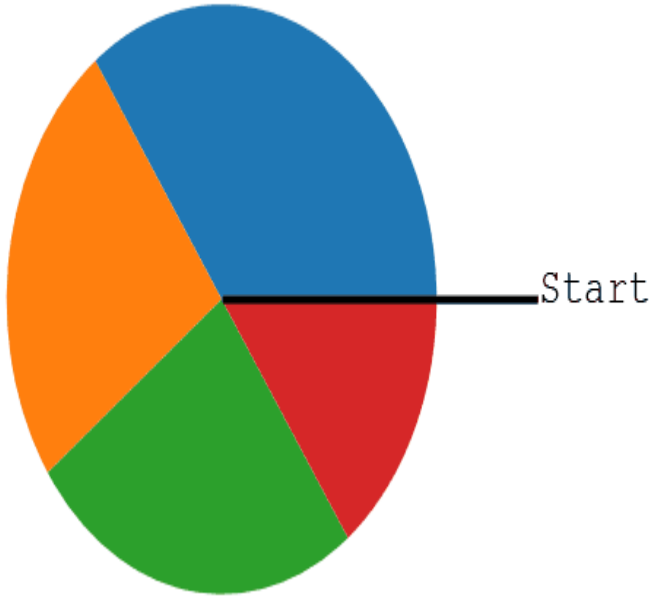
- import matplotlib.pyplot
  as plt
  import numpy as np

  y =
  np.array([35, 25, 25, 15])

  plt.pie(y)
  plt.show()

Creating Pie Charts

- As you can see the pie chart draws one piece (called a wedge) for each value in the array (in this case [35, 25, 25, 15]).
- By default the plotting of the first wedge starts from the x-axis and move *counterclockwise*:
- **Note:** The size of each wedge is determined by comparing the value with all the other values, by using this formula:
- The value divided by the sum of all values: x/sum(x)

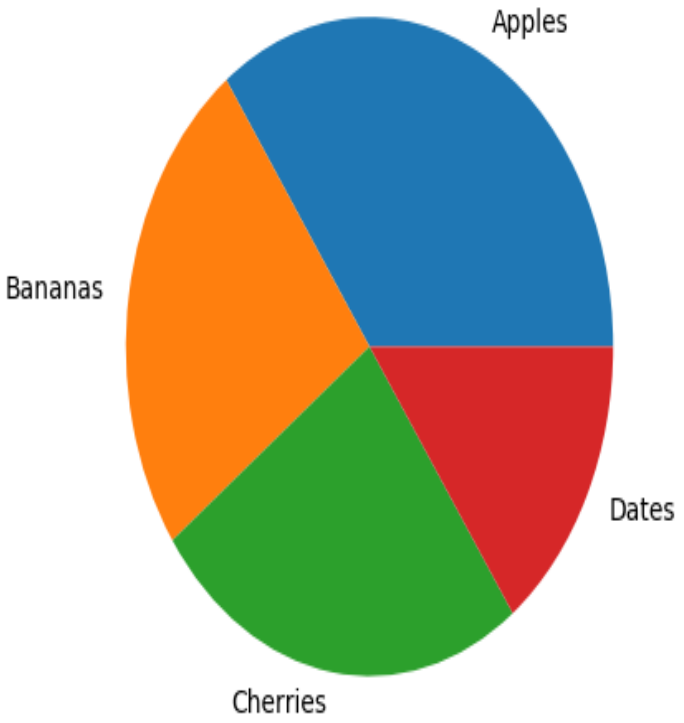


Labels

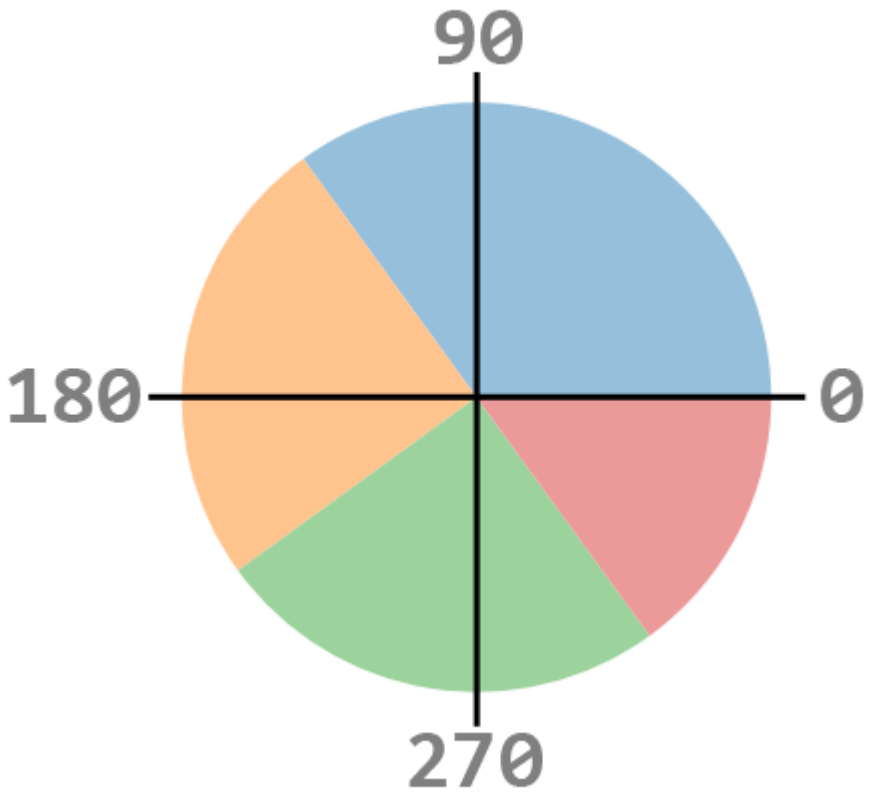
- import matplotlib.pyplot as plt
  import numpy as np

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

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



Start Angle

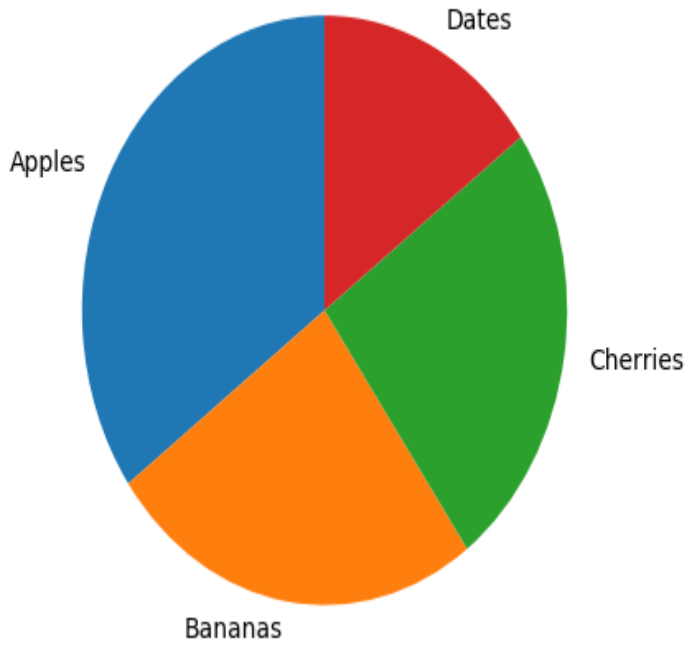


Example

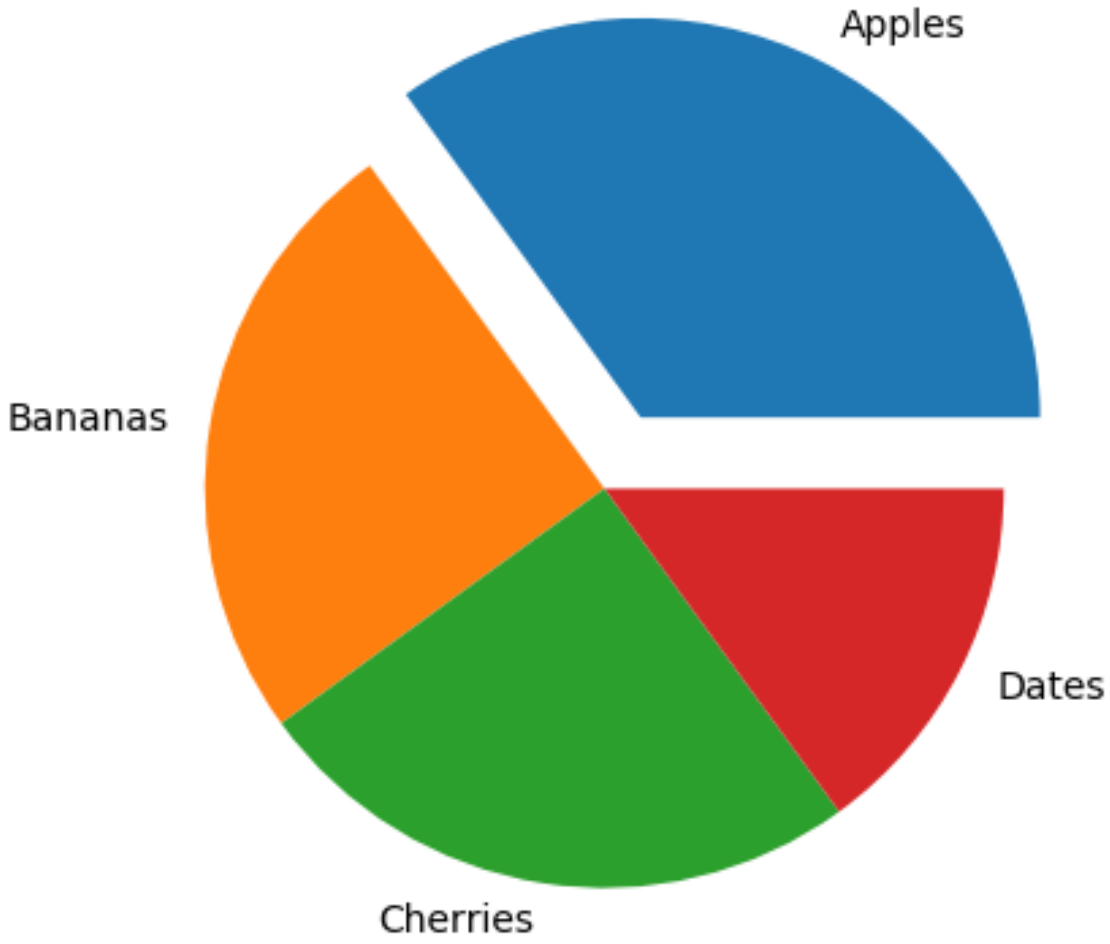
- import matplotlib.pyplot as plt
  import numpy as np

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

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

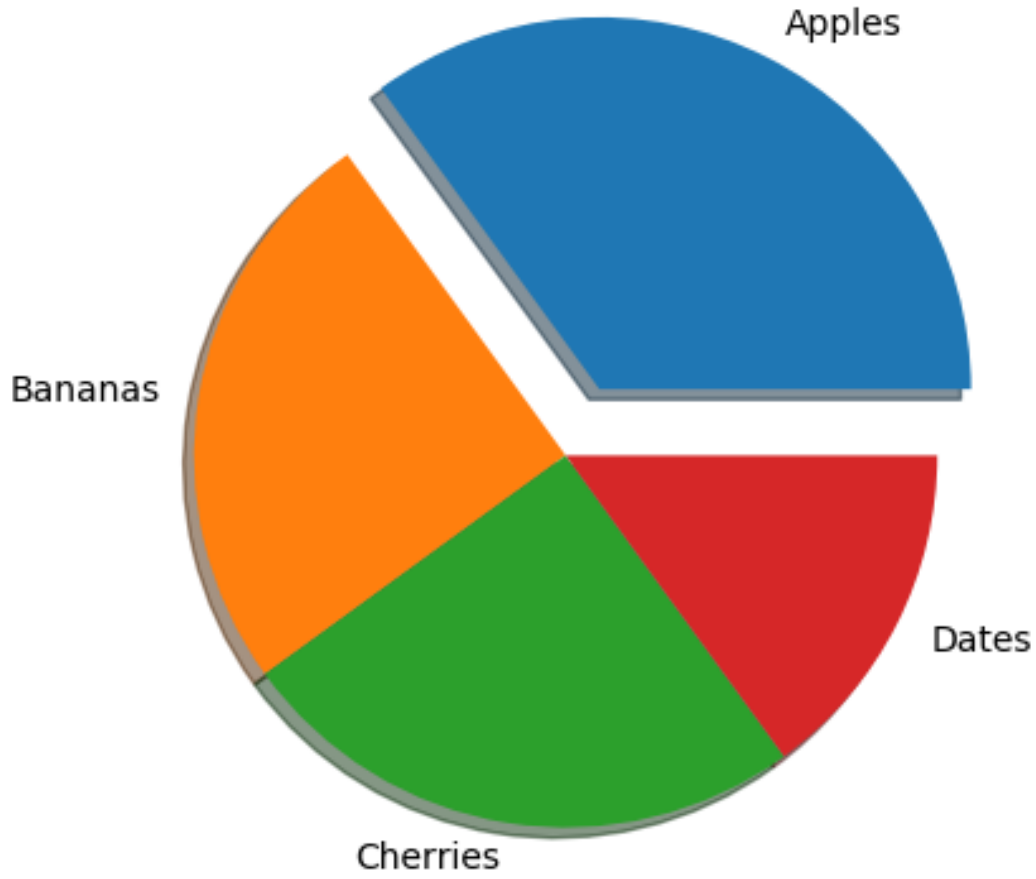


Exploded Pie Charts

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

  y = np.array([35, 25, 25, 15])
  mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
  myexplode = [0.2, 0, 0, 0]

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


Apples
Bananas
Dates
Cherries

Shadow

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

  y = np.array([35, 25, 25, 15])
  mylabels = ["Apples", "Bananas", "Cherries", "Dates"]
  myexplode = [0.2, 0, 0, 0]

  plt.pie(y, labels = mylabels, explode = myexplode,
  shadow = True)
  plt.show()
  ```
  ```


Apples
Bananas
Dates
Cherries

Colors

- import matplotlib.pyplot as plt
  import numpy as np

  y = np.array([35, 25, 25, 15])
  mylabels =
  ["Apples", "Bananas", "Cherri
  es", "Dates"]
  mycolors =
  ["black", "hotpink", "b", "#4CA
  F50"]

  plt.pie(y, labels = mylabels,
  colors = mycolors)
  plt.show()