Basic plotting

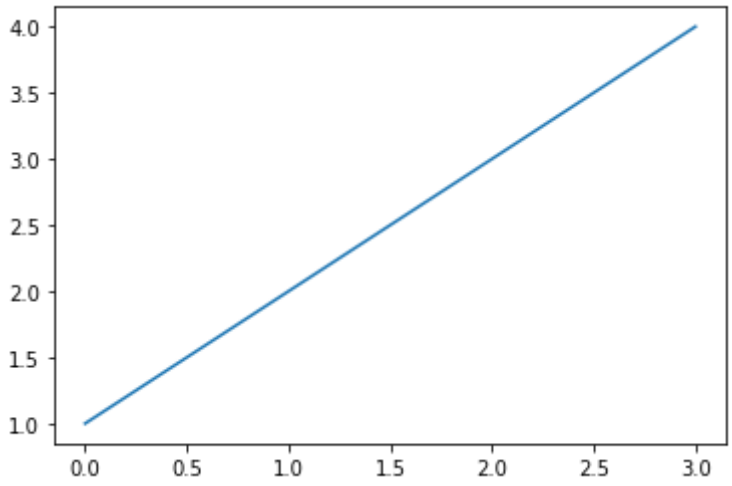
Ref - https://matplotlib.org/tutorials/introductory/pyplot.html

```
1 import matplotlib.pyplot as plt
```

Plot a list (or array)

```
1 plt.plot([1,2,3,4])
```

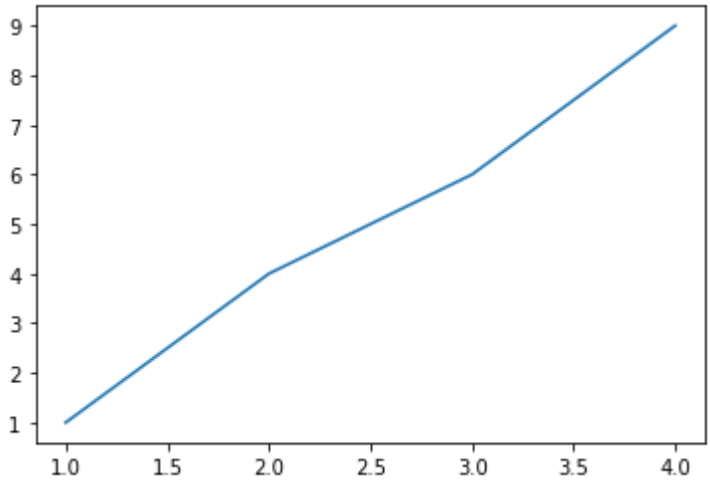
```
[<matplotlib.lines.Line2D at 0x7fc06c5b10f0>]
```



Plot one list vs other list

```
1 plt.plot([1,2,3,4],[1,4,6,9])
```

```
[<matplotlib.lines.Line2D at 0x7fc06c0e31d0>]
```

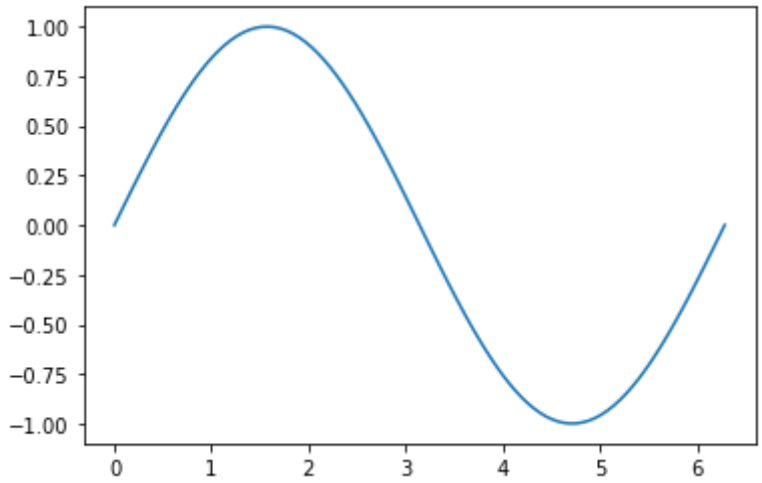


Plot an array (numpy)

```
1 import numpy as np
```

```
1 x = np.linspace(0,2*np.pi,100)
2
3 y = np.sin(x)
4
5 plt.plot(x,y)
```

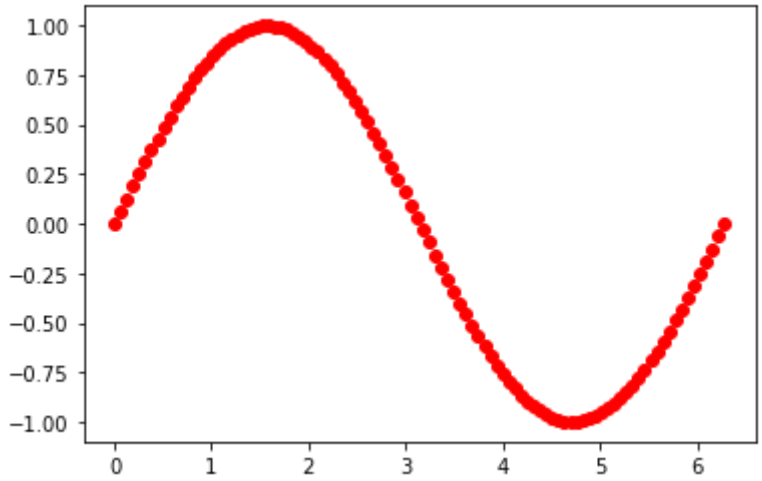
```
[<matplotlib.lines.Line2D at 0x7fc06c0727b8>]
```



Plot using dots

```
1 plt.plot(x,y,'ro')
```

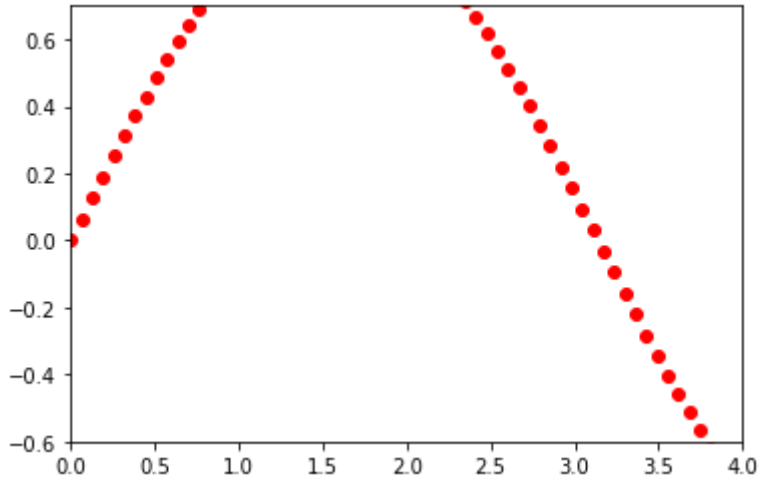
```
[<matplotlib.lines.Line2D at 0x7fc06bfd8dd8>]
```



Define axis lengths

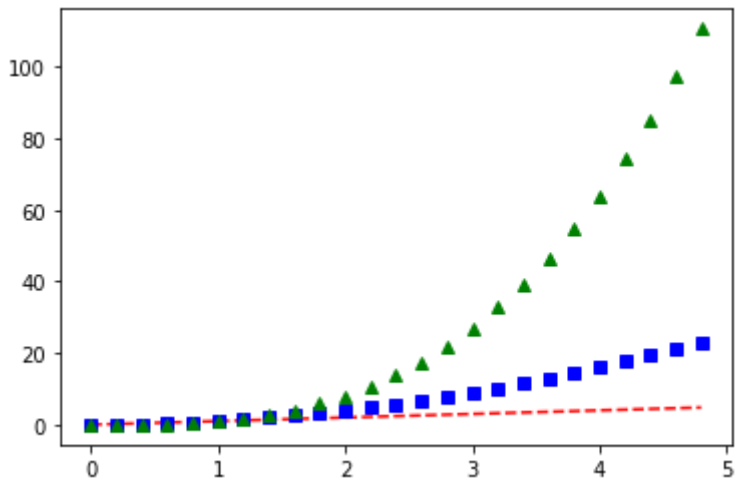
```
1 plt.plot(x,y,'ro')
2 plt.axis([0,4,-0.6,0.7])
```

```
(0.0, 4.0, -0.6, 0.7)
```



Plot multiple curves simultaneously

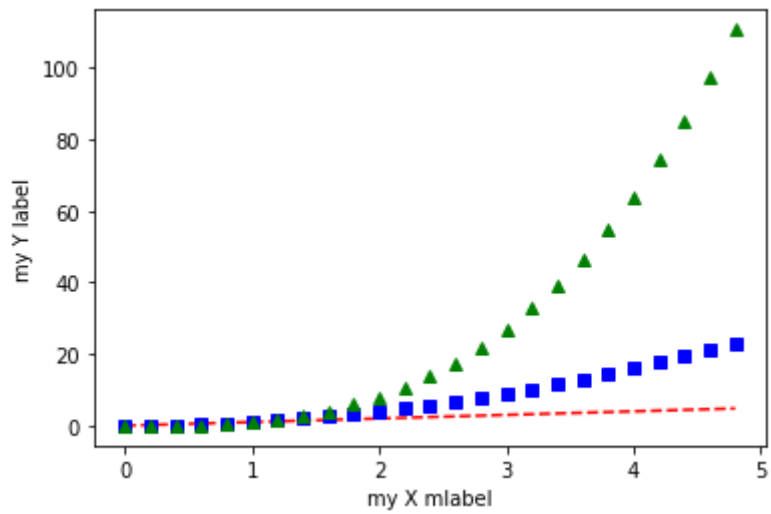
```
1 import numpy as np
2
3 # evenly sampled time at 200ms intervals
4 t = np.arange(0., 5., 0.2)
5
6 # red dashes, blue squares and green triangles
7 plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
8 plt.show()
```



Labels for axes

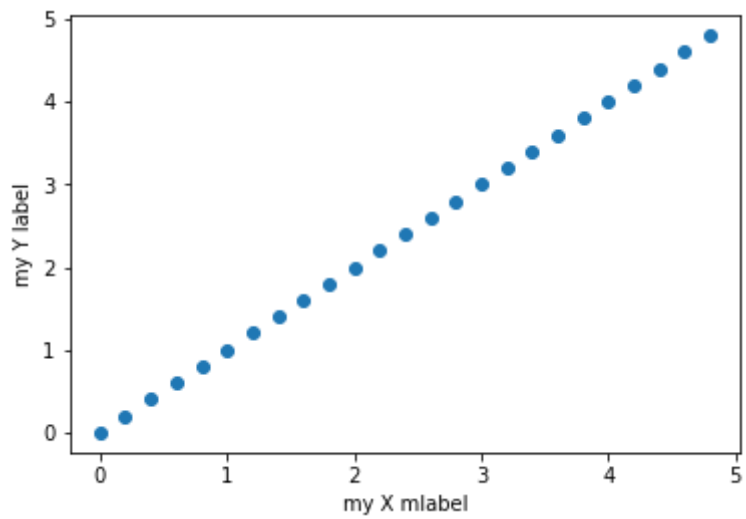
```
1 import numpy as np
2
3 # evenly sampled time at 200ms intervals
4 t = np.arange(0., 5., 0.2)
5
6 # red dashes, blue squares and green triangles
7 plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
8
```

```
 9 plt.xlabel('my X mlabel')
10 plt.ylabel('my Y label')
11
12 plt.show()
```



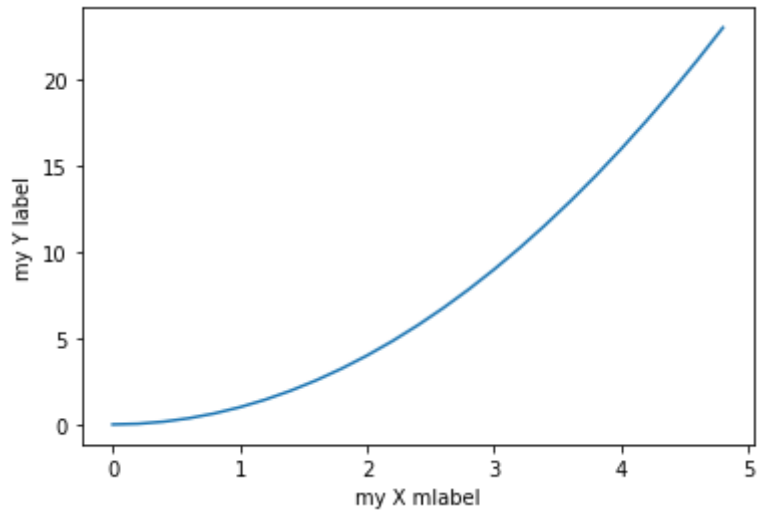
Scatter plot

```
 1 import numpy as np
 2
 3 # evenly sampled time at 200ms intervals
 4 t = np.arange(0., 5., 0.2)
 5
 6 # red dashes, blue squares and green triangles
 7 plt.scatter(t, t)
 8
 9 plt.xlabel('my X mlabel')
10 plt.ylabel('my Y label')
11
12 plt.show()
```

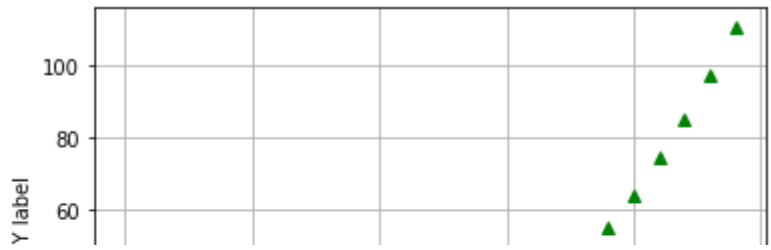


Plotting with dictionary of labelled arrays

```
import numpy as np

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

mylabelled_data = {'my_x':t, 'my_y':t**2}

plt.plot('my_x','my_y', data=mylabelled_data)

plt.xlabel('my X mlabel')
plt.ylabel('my Y label')

plt.show()
```




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

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)

# red dashes, blue squares and green triangles
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')

plt.xlabel('my X mlabel')
plt.ylabel('my Y label')

plt.grid(True)

plt.show()
```

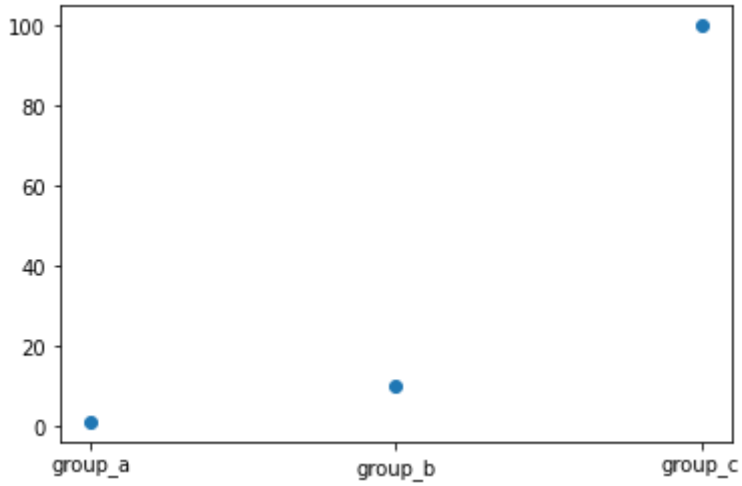


Advanced plotting

Categorical attributes

```
1 import matplotlib.pyplot as plt
2
3 names = ['group_a', 'group_b', 'group_c']
4 values = [1, 10, 100]
5
6 plt.scatter(names,values)
```

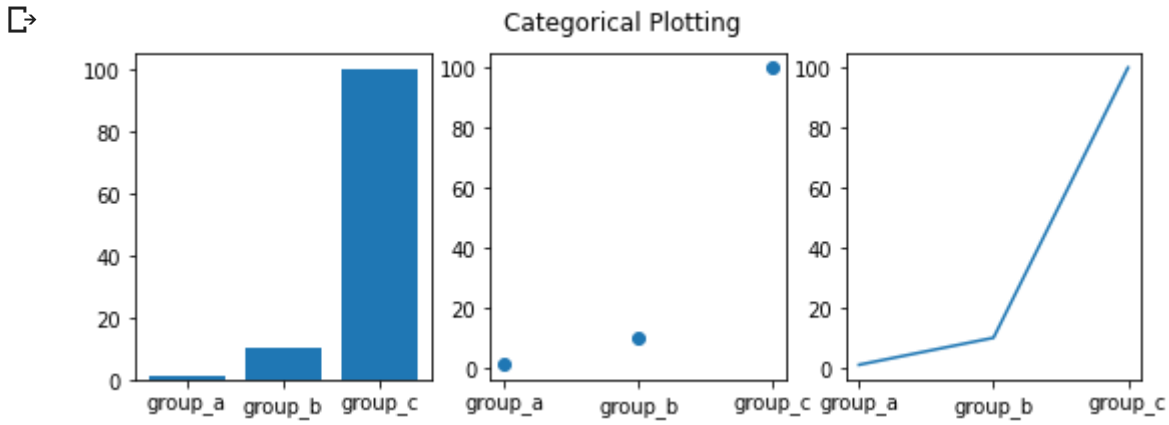
```
<matplotlib.collections.PathCollection at 0x7fc06bd399e8>
```



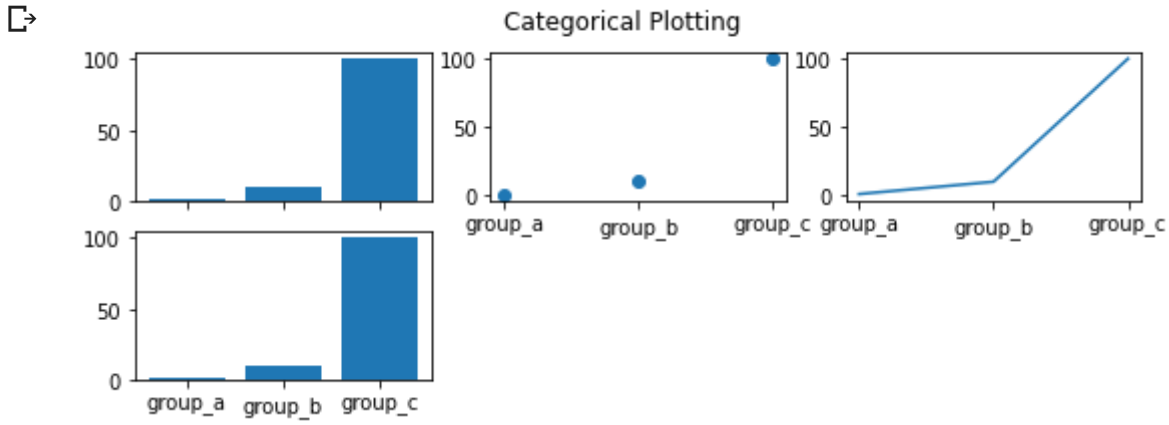
Sub plots

```
1 import matplotlib.pyplot as plt
2
3 names = ['group_a', 'group_b', 'group_c']
4 values = [1, 10, 100]
5
6 plt.figure(figsize=(9, 3))
7
8 plt.subplot(131)
9 plt.bar(names, values)
10 plt.subplot(132)
11 plt.scatter(names, values)
12 plt.subplot(133)
13 plt.plot(names, values)
```

```
14 plt.suptitle('Categorical Plotting')
15
16
17 plt.show()
```



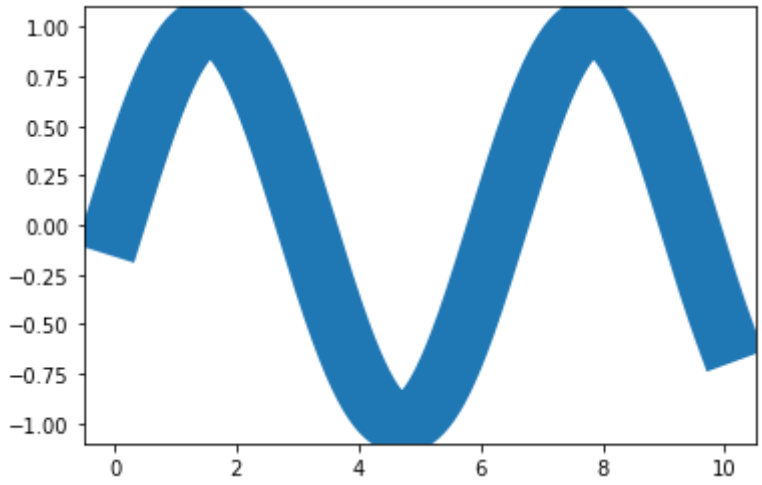

```
 1 import matplotlib.pyplot as plt
 2
 3 names = ['group_a', 'group_b', 'group_c']
 4 values = [1, 10, 100]
 5
 6 plt.figure(figsize=(9, 3))
 7
 8 ax1 = plt.subplot(231)
 9 plt.bar(names, values)
10 plt.subplot(232)
11 plt.scatter(names, values)
12 plt.subplot(233)
13 plt.plot(names, values)
14 plt.suptitle('Categorical Plotting')
15
16 plt.subplot(234, sharex=ax1) #to get same labels as the other plot
17 plt.bar(names, values)
18 plt.setp(ax1.get_xticklabels(), visible=False) #to turn off labels
19
20 plt.show()
21
```



Simple properties

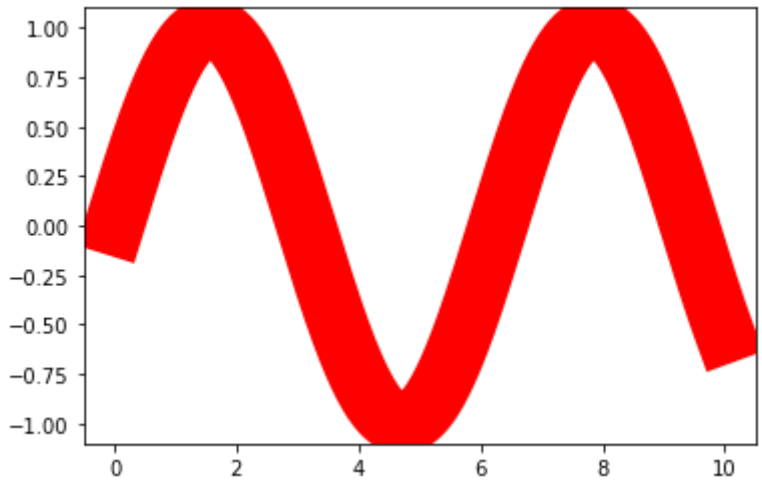
```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x = np.linspace(0,10,1000)
5 y = np.sin(x)
6
7 plt.plot(x,y,linewidth=30.0)
8
```

```
[<matplotlib.lines.Line2D at 0x7fc06be5c5c0>]
```




```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x = np.linspace(0,10,1000)
5 y = np.sin(x)
6
7 plt.plot(x,y,linewidth=30.0,color='red')
```

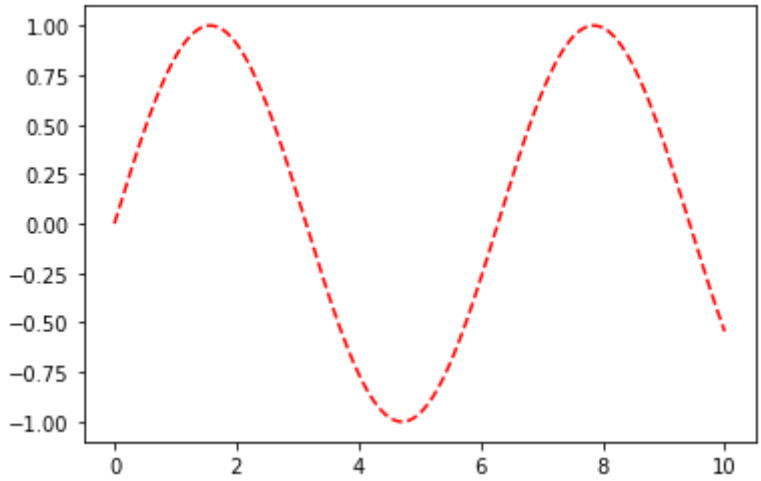
```
[<matplotlib.lines.Line2D at 0x7fc06bcf0cf8>]
```




```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x = np.linspace(0,10,1000)
5 y = np.sin(x)
```

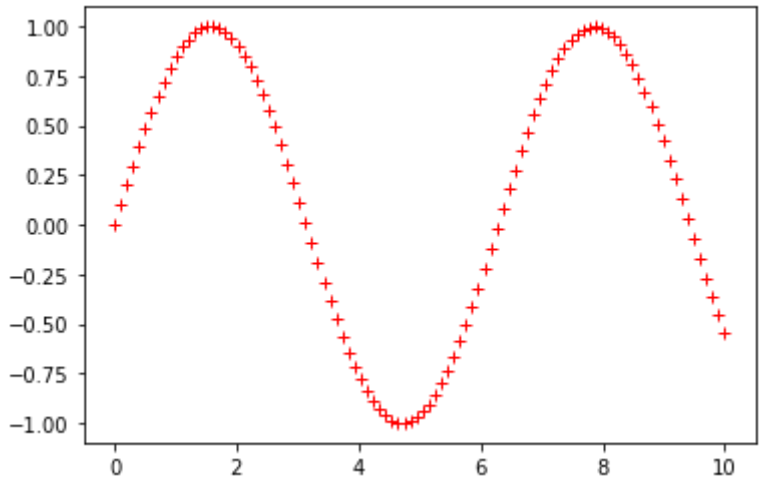
```
6
7 plt.plot(x,y,'r--')
```

```
[<matplotlib.lines.Line2D at 0x7fc06becd8d0>]
```




```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x = np.linspace(0,10,100)
5 y = np.sin(x)
6
7 plt.plot(x,y,'r+')
```

```
[<matplotlib.lines.Line2D at 0x7fc06be8ccf8>]
```

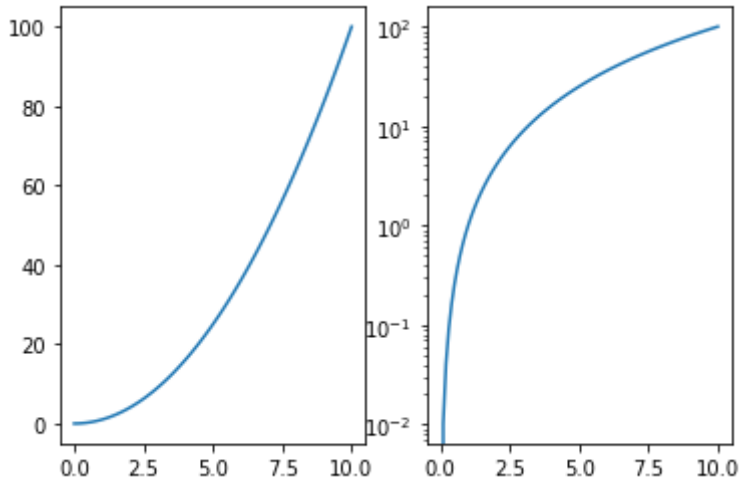


Axes scales

```
1 import numpy as np
2 import matplotlib.pyplot as plot
3
4
5 x = np.linspace(0,10,100)
6 y = x**2
7
8 plt.subplot(121)
9 plt.yscale('linear')
```

```
plt.plot(x,y)

plt.subplot(122)
plt.yscale('log')
plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x7fc06bf28898>]
```



Contours

REF - https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.contour.html

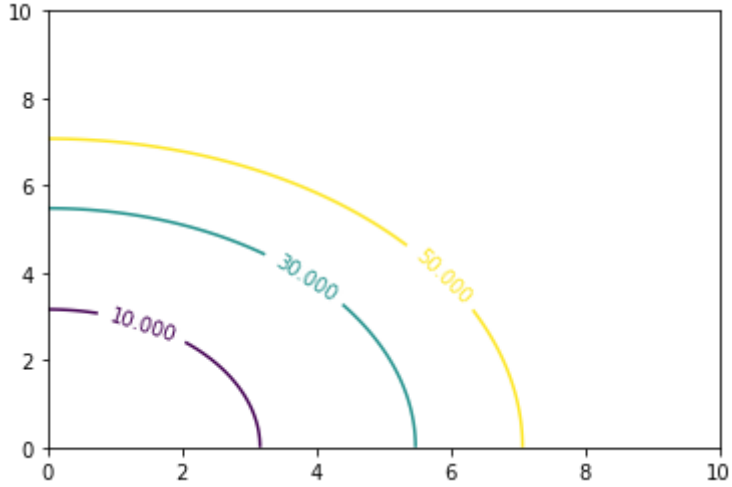
```
a = np.linspace(0,10,100)

X,Y = np.meshgrid(a,a)

Z = X**2 + Y**2

plt.contour(X,Y,Z, levels=[10,30,50])
```

```
<matplotlib.contour.QuadContourSet at 0x7fc06bf85668>
```


```
x = np.linspace(0,10,100)

X,Y = np.meshgrid(x,x)

```

```
Z = X**2 + Y**2

CS = plt.contour(X,Y,Z, levels=[10,30,50])

plt.clabel(CS)
```

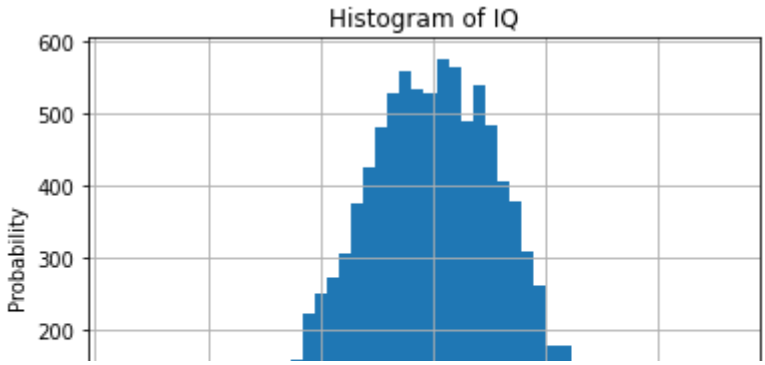
```
<a list of 3 text.Text objects>
```



Text on plots

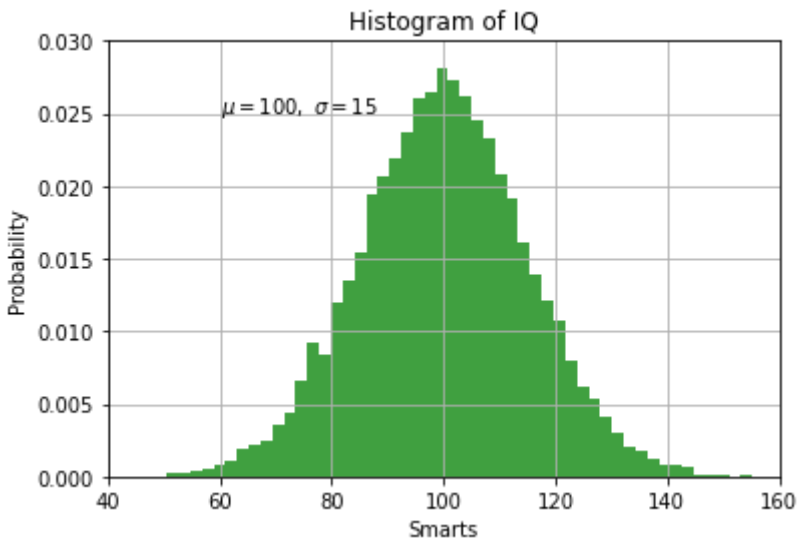
Axes labels and title

```
import numpy as np

mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50)


plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')

#plt.axis([40, 160, 0, 0.03])

plt.grid(True)

plt.show()
```



Text label inside the plot

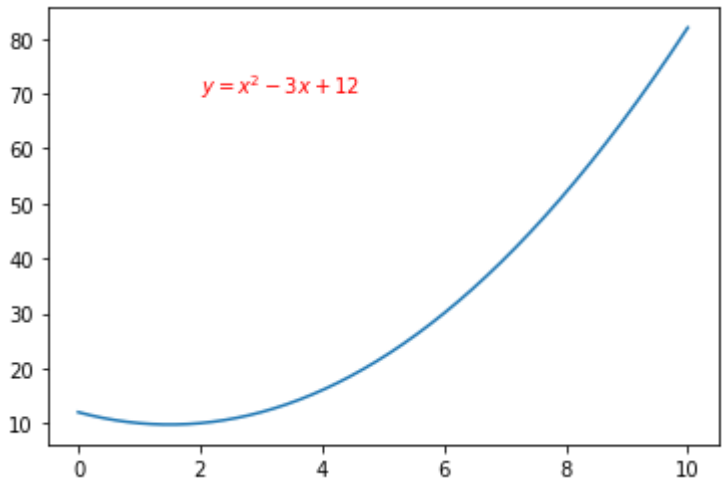

```
import numpy as np

mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50, density=1, facecolor='g', alpha=0.75)

plt.text(60, .025, r'$\mu=100,\ \sigma=15$')

plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title('Histogram of IQ')

plt.axis([40, 160, 0, 0.03])

plt.grid(True)

plt.show()

```



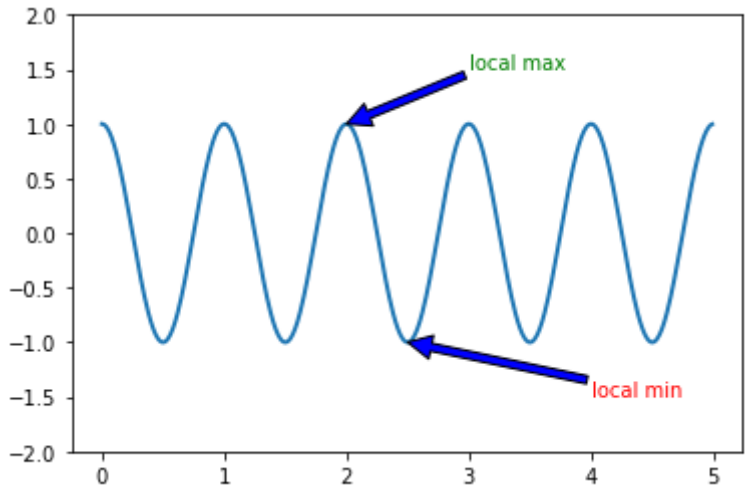

```
import numpy as np
import matplotlib.pyplot as plt

```

```
4
5 x = np.linspace(0,10,1000)
6 y = x**2 - 3*x + 12
7
8 plt.plot(x,y)
9
10 plt.text(2,70,'$y=x^2 - 3 x + 12$',color='red')
11
```

```
Text(2, 70, '$y=x^2 - 3 x + 12$')
```




```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 ax = plt.subplot(111)
5
6 t = np.arange(0.0, 5.0, 0.01)
7 s = np.cos(2*np.pi*t)
8 line, = plt.plot(t, s, lw=2)
9
10 plt.annotate('local max', xy=(2, 1), xytext=(3, 1.5), arrowprops={'facecolor':'blue'},c
11
12 plt.annotate('local min', xy=(2.5, -1), xytext=(4, -1.5), arrowprops={'facecolor':'blue
13
14 plt.ylim(-2, 2)
15 plt.show()
```

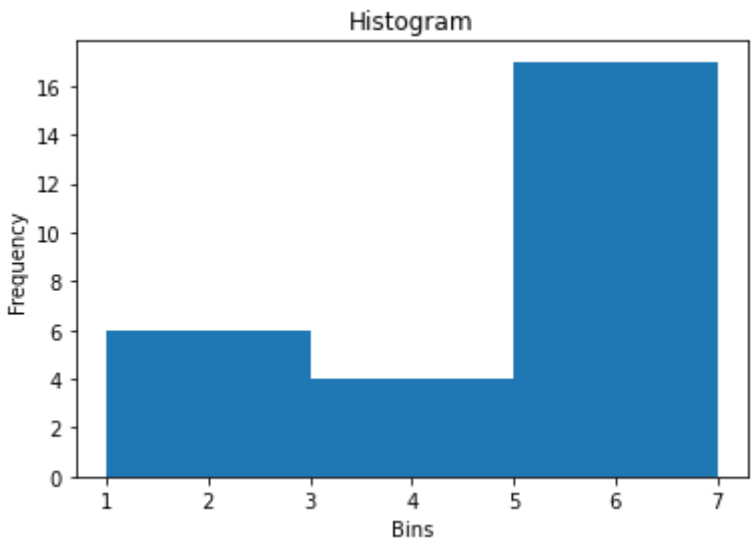


Statistical plots

Complex example REF - https://matplotlib.org/gallery/statistics/histogram_features.html

Histogram

```
1
2 import matplotlib
3 import numpy as np
4 import matplotlib.pyplot as plt
5
6 np.random.seed(42)
7
8 x = [1,1,1,2,2,2,3,3,4,4,5,5,5,5,5,5,6,6,6,6,6,6,6,6,7]
9
10 num_bins = 3
11
12 # the histogram of the data
13 n, bins, patches = plt.hist(x, num_bins)
14
15
16 plt.xlabel('Bins')
17 plt.ylabel('Frequency')
18 plt.title(r'Histogram')
19
20 plt.show()
```

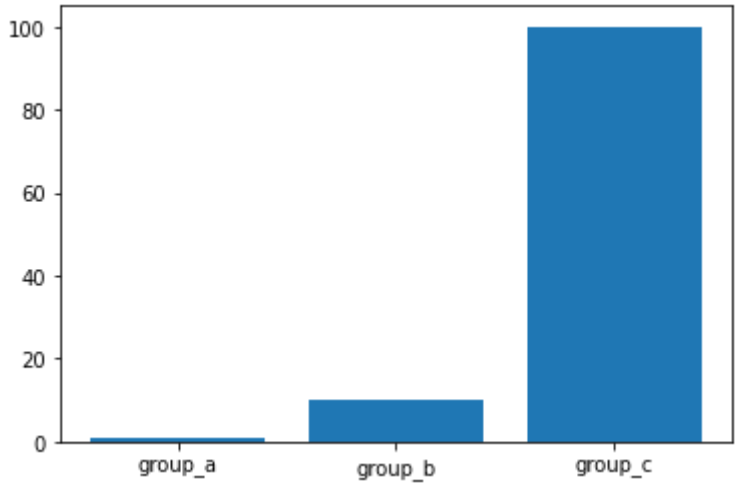


Bar plot

```
1 import matplotlib.pyplot as plt
2
3 names = ['group_a', 'group_b', 'group_c']
```

```
4 values = [1, 10, 100]
5
6 plt.bar(names,values)
```

```
<BarContainer object of 3 artists>
```

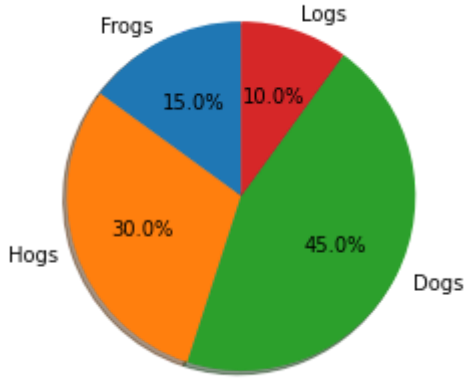


Pie plot

REF - https://matplotlib.org/gallery/pie_and_polar_charts/pie_features.html

Simple pie

```
 1 import matplotlib.pyplot as plt
 2
 3 # Pie chart, where the slices will be ordered and plotted counter-clockwise:
 4 labels1 = 'Frogs', 'Hogs', 'Dogs', 'Logs'
 5 sizes = [15, 30, 45, 10]
 6
 7 plt.pie(sizes, labels=labels1, autopct='%1.1f%%',
 8         shadow=True, startangle=90)
 9
10 plt.show()
11
```



Complex pie

```
import matplotlib.pyplot as plt

# Pie chart, where the slices will be ordered and plotted counter-clockwise:
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0)  # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```