Matplotlib

Introduction

Matplotlib is a library for python that was created to support matlab-style plotting. To use matplotlib, download the matplotlib package and add the following to your code…

```
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline  # only required if you're using jupyter notebook + you want to output images
```

If you're using jupyter notebook, your plots should show up automatically. If you're using standalone python, you can use the show() method to open up a window to show your plot.

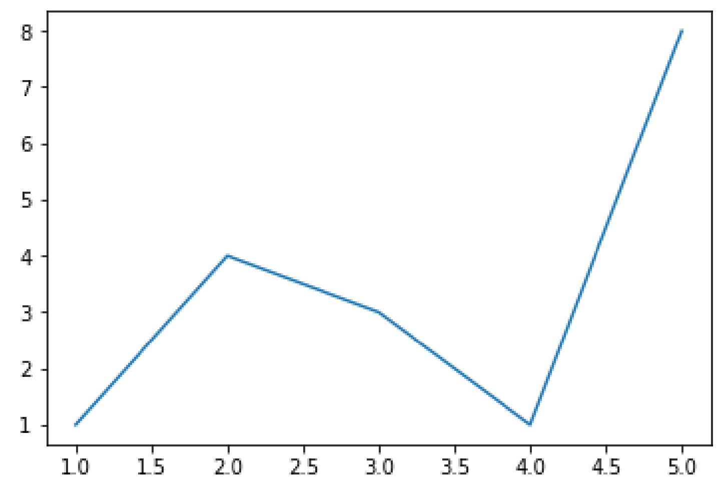
```
plt.plot([1,2,3,4,5], [1,4,3,1,8])
plt.show()
```

Note that once you show(), the data being plotted will be cleared. If you want to do it again, you need to re-plot your data.

Plot Basics

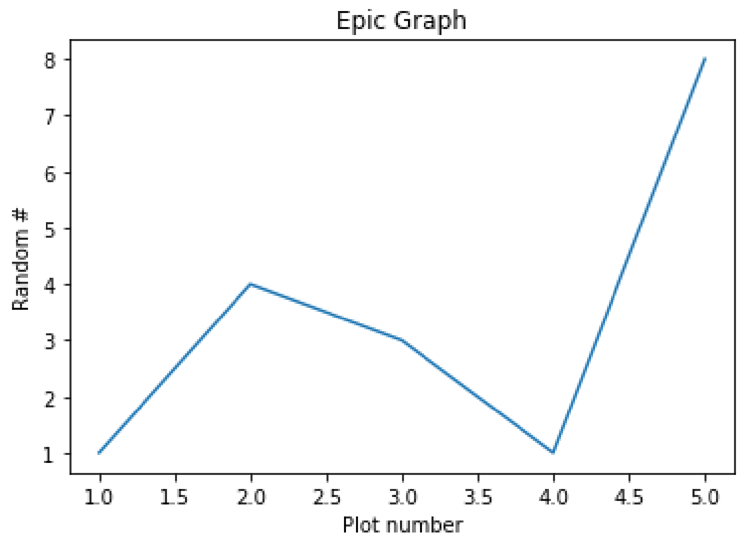
Simple

```
x = [1,2,3,4,5]
y = [1,4,3,1,8]
plt.plot(x, y)
plt
```



Title and Axis Labels

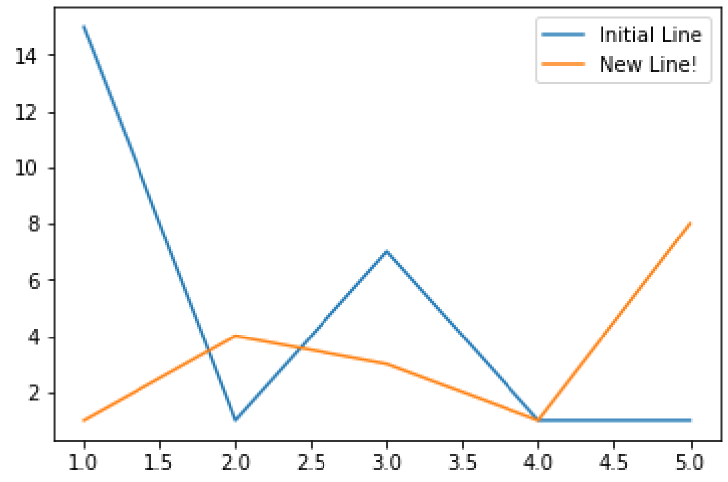
```
x = [1,2,3,4,5]
y = [1,4,3,1,8]
plt.plot(x, y)
plt.title('Epic Graph')
plt.xlabel('Plot number')
plt.ylabel('Random #')
plt
```



Legends

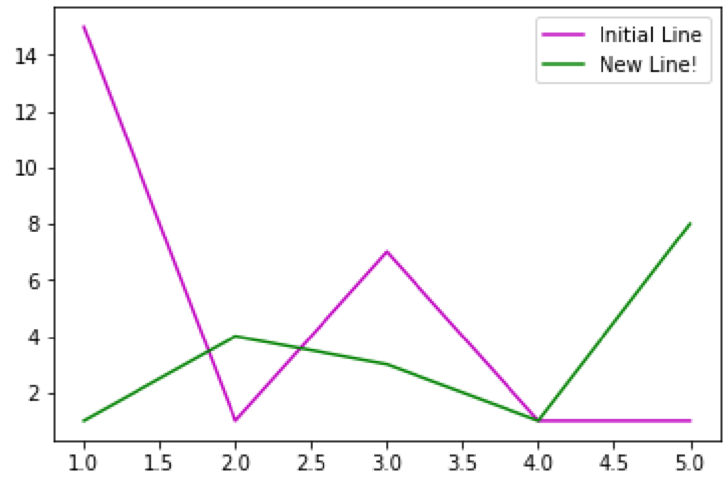
```
x = [1,2,3,4,5]
y = [15,1,7,1,1]
```

```
y2 = [1,4,3,1,8]
plt.plot(x, y, label='Initial Line')
plt.plot(x, y2, label='New Line!')
plt.legend()
plt
```



Colors

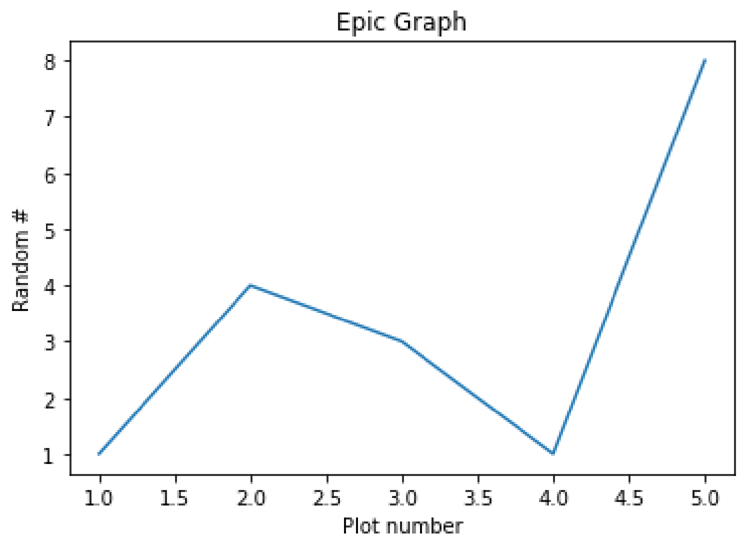
```
x = [1,2,3,4,5]
y = [15,1,7,1,1]
y2 = [1,4,3,1,8]
plt.plot(x, y, label='Initial Line', color='m')
plt.plot(x, y2, label='New Line!', color='g')
plt.legend()
plt
```



Plot Types

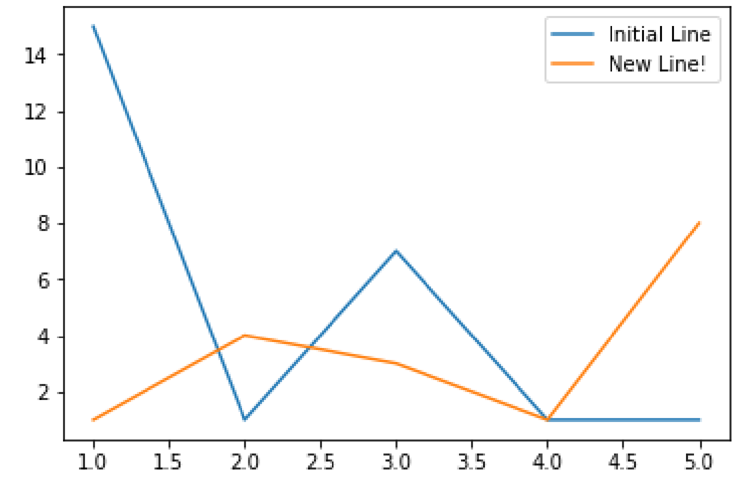
Line

```
x = [1,2,3,4,5]
y = [1,4,3,1,8]
plt.plot(x, y)
plt.title('Epic Graph')
plt.xlabel('Plot number')
plt.ylabel('Random #')
plt
```



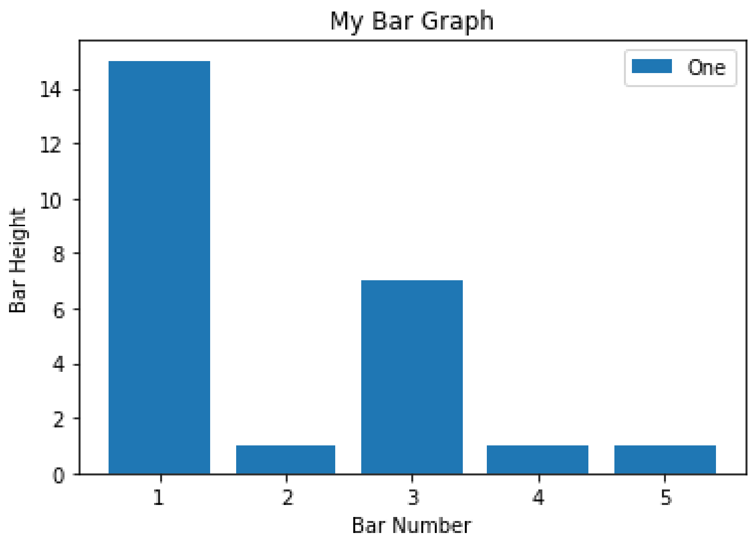
Multi

```
x = [1,2,3,4,5]
y = [15,1,7,1,1]
y2 = [1,4,3,1,8]
plt.plot(x, y, label='Initial Line')
plt.plot(x, y2, label='New Line!')
plt.legend()
plt
```



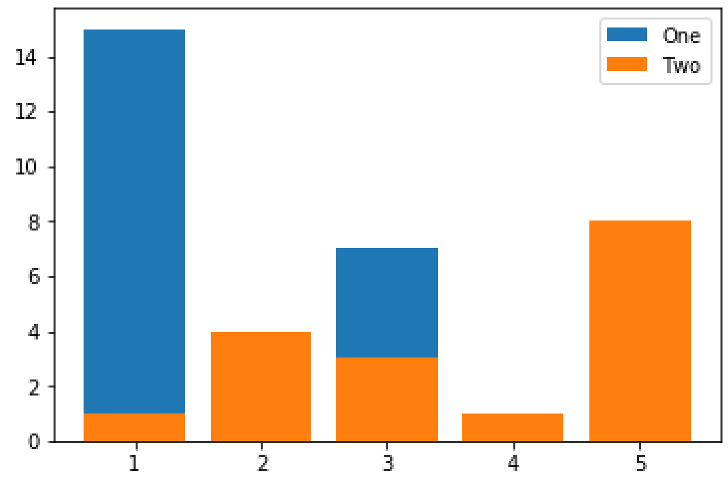
Bar

```
x = [1,2,3,4,5]
y = [15,1,7,1,1]

plt.bar(x, y, label="One")

plt.title('My Bar Graph')
plt.xlabel('Bar Number')
plt.ylabel('Bar Height')
plt.legend()
plt
```



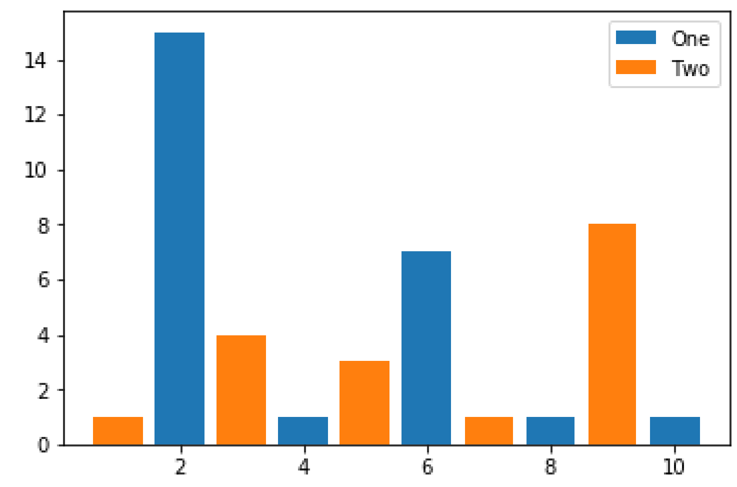
Multi Stacked

```
x = [1,2,3,4,5]
y = [15,1,7,1,1]
y2 = [1,4,3,1,8]

plt.bar(x, y, label="One")
plt.bar(x, y2, label="Two")

plt.legend()
plt
```



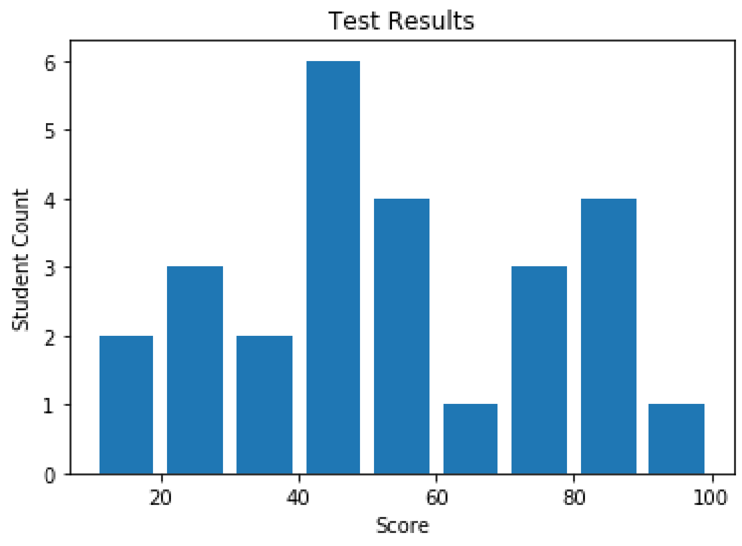
Multi Spaced

```
x = [2,4,6,8,10]
y = [15,1,7,1,1]

x2 = [1,3,5,7,9]
y2 = [1,4,3,1,8]

plt.bar(x, y, label="One")
plt.bar(x2, y2, label="Two")

plt.legend()
plt
```



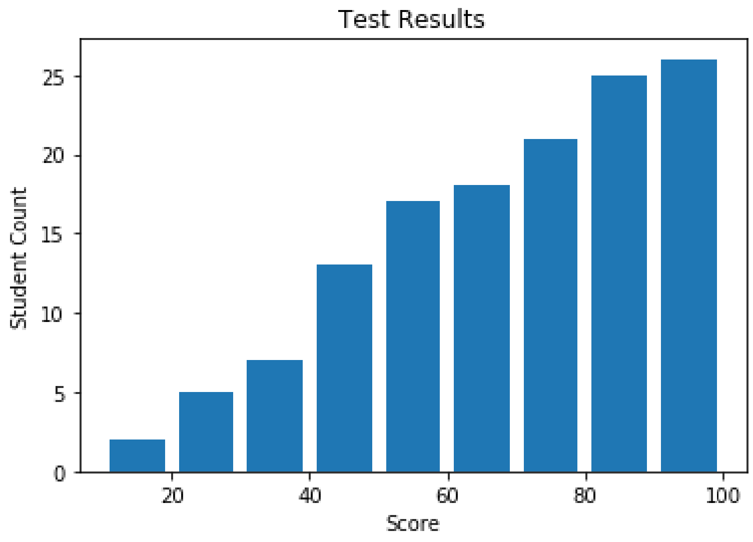
Histogram

```
test_scores = [55, 45, 88, 75, 43, 56, 89, 55, 46, 76,41, 23, 45, 86, 88, 90, 11, 22, 33, 41, 11,
33, 56, 26, 66, 77]

bins = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

plt.title('Test Results')
plt.xlabel('Score')
plt.ylabel('Student Count')

plt.hist(test_scores, bins, histtype='bar', rwidth=0.8)
plt
```



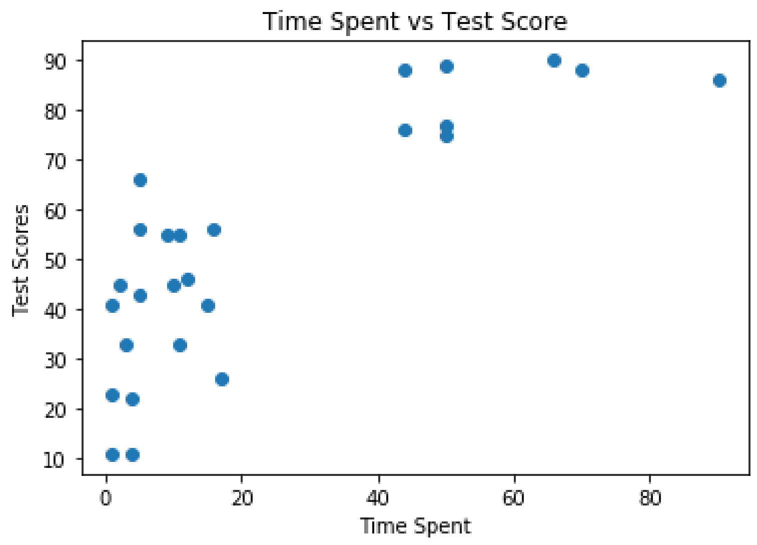
Cumulative (add previous)

```
test_scores = [55, 45, 88, 75, 43, 56, 89, 55, 46, 76,41, 23, 45, 86, 88, 90, 11, 22, 33, 41, 11,
33, 56, 26, 66, 77]
x = [x for x in range(len(test_scores))]

bins = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

plt.title('Test Results')
plt.xlabel('Score')
plt.ylabel('Student Count')

plt.hist(test_scores, bins, histtype='bar', cumulative=True, rwidth=0.8)
plt
```



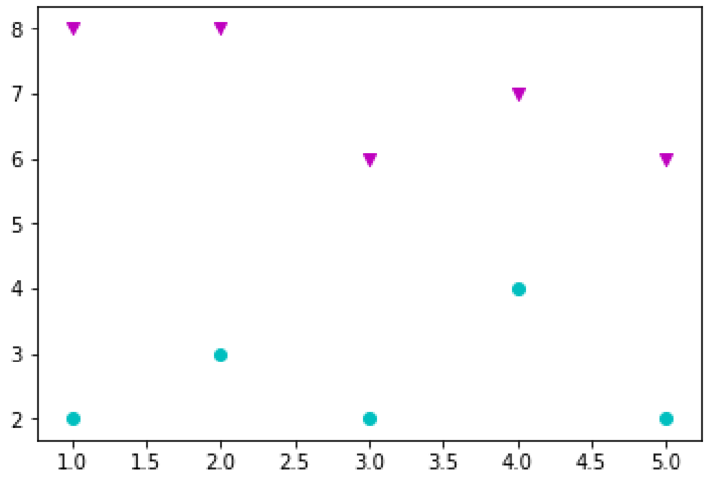
Scatter

```
test_scores = [55, 45, 88, 75, 43, 56, 89, 55, 46, 76, 41, 23, 45, 86, 88, 90, 11, 22, 33, 41, 11,
33, 56, 26, 66, 77]
time_spent  = [11, 10, 44, 50, 5,  5,  50,  9, 12, 44, 15, 1,  2,  90, 70, 66, 4,  4,  11, 1,  1,  3,  16,
17, 5,  50]
x = [x for x in range(len(test_scores))]

bins = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

plt.title('Time Spent vs Test Score')
plt.xlabel('Time Spent')
plt.ylabel('Test Scores')
plt.scatter(time_spent, test_scores)
plt
```



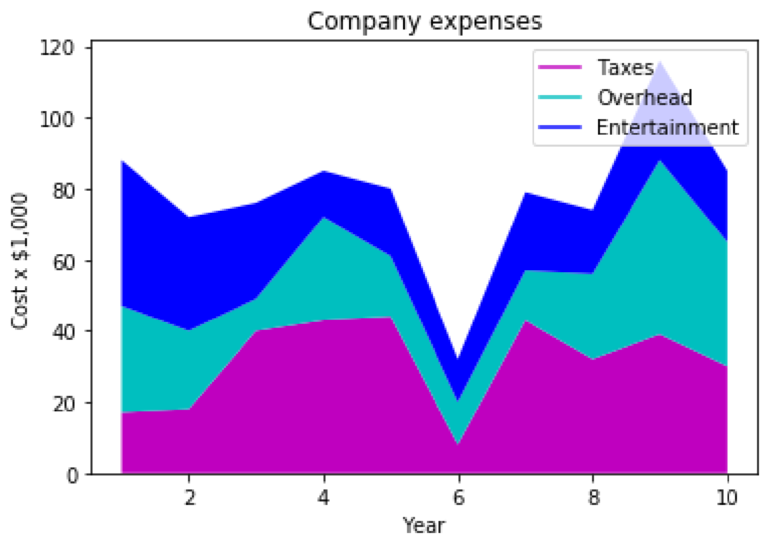
Multi

```
x = [1,2,3,4,5]
y1 = [2,3,2,4,2]
y2 = [8,8,6,7,6]

plt.scatter(x, y1, marker='o', color='c')
plt.scatter(x, y2, marker='v', color='m')

plt
```



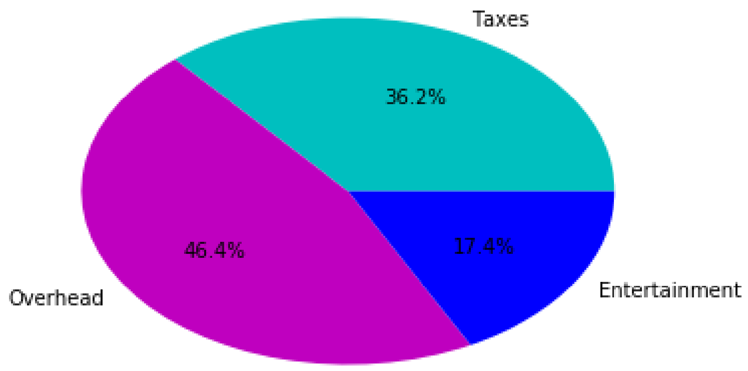
Stack

```
year = [1,2,3,4,5,6,7,8,9,10]

# in thousands of dollars
taxes = [17,18,40,43,44,8,43,32,39,30]
overhead = [30,22,9,29,17,12,14,24,49,35]
entertainment = [41,32,27,13,19,12,22,18,28,20]

# there isn't an easy way to put a legend in a stack plot...
# we can fake it by passing in empty line plots
#
# anything that fills in polygons is not really legend-able
plt.plot([], [], color='m', label='Taxes')
plt.plot([], [], color='c', label='Overhead')
plt.plot([], [], color='b', label='Entertainment')

# notice this is different than previous graph types...
# the first arg is x, rest of them are y (before you start using keyword args)
# notice the colors being passed in are the same as the fake line plots
plt.stackplot(year, taxes, overhead, entertainment, colors=['m','c','b'])
plt.legend()
plt.title('Company expenses')
plt.xlabel('Year')
plt.ylabel('Cost x $1,000')
plt
```



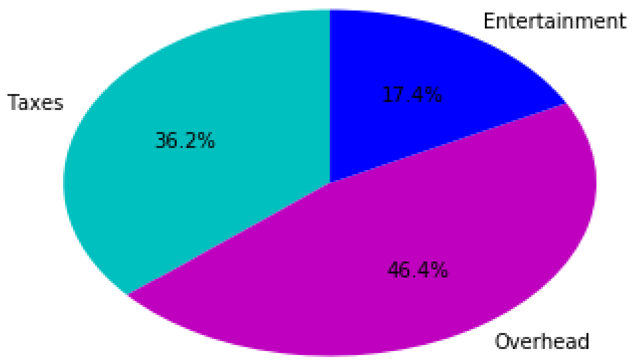
Pie Charts

```
labels = ('Taxes', 'Overhead', 'Entertainment')

sizes = [25, 32, 12]
colors = ['c', 'm', 'b']

plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%')
plt
```



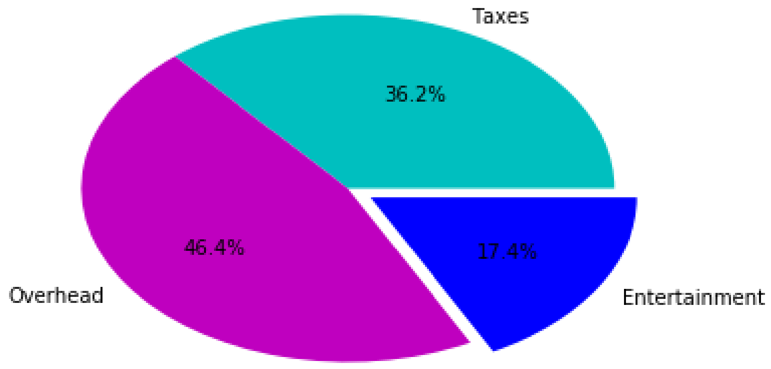
Axis-aligned

```
labels = ('Taxes', 'Overhead', 'Entertainment')

sizes = [25, 32, 12]
colors = ['c', 'm', 'b']

plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=90)
plt
```



Explode

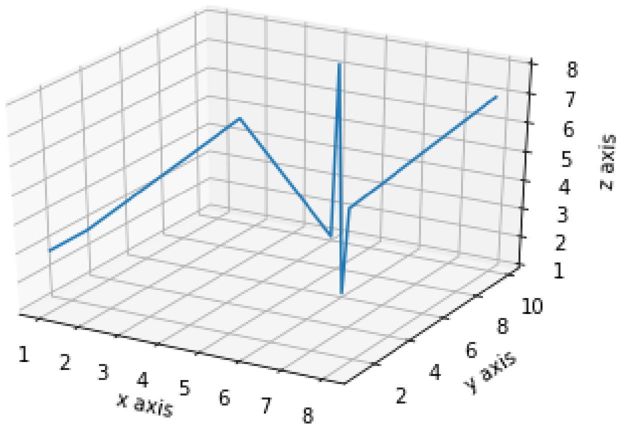
```
labels = ('Taxes', 'Overhead', 'Entertainment')

sizes = [25, 32, 12]
colors = ['c', 'm', 'b']

plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', explode=(0,0,0.1))
plt
```



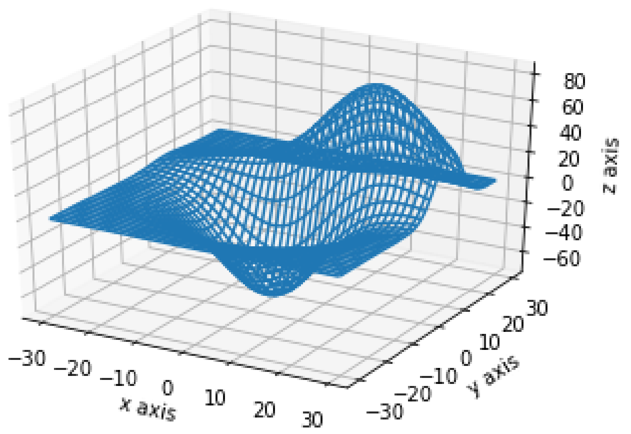
3D Line

```
from mpl_toolkits.mplot3d import axes3d

fig = plt.figure()
ax1 = fig.add_subplot(111, projection = '3d')
```

```
x = [1,2,3,4,5,6,7,8]
y = [1,1,7,10,8,6,4,10]
z = [3,4,6,1,8,1,5,7]

ax1.plot_wireframe(x,y,z)
ax1.set_xlabel('x axis')
ax1.set_ylabel('y axis')
ax1.set_zlabel('z axis')

plt
```

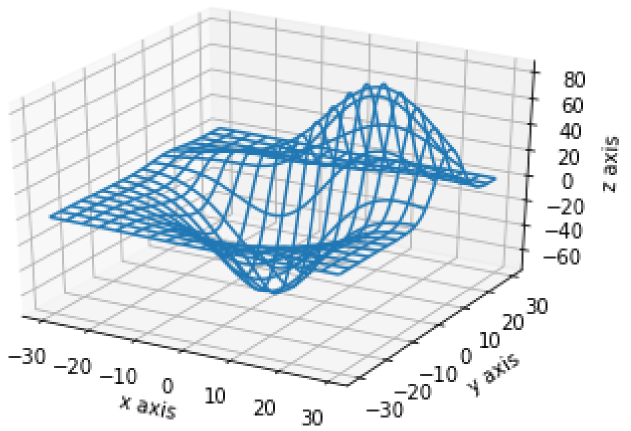


```
from mpl_toolkits.mplot3d import axes3d

fig = plt.figure()
ax1 = fig.add_subplot(111, projection = '3d')

x, y, z = axes3d.get_test_data() # generated test data

ax1.plot_wireframe(x, y, z)

ax1.set_xlabel('x axis')
ax1.set_ylabel('y axis')
ax1.set_zlabel('z axis')
plt.legend()
plt
```



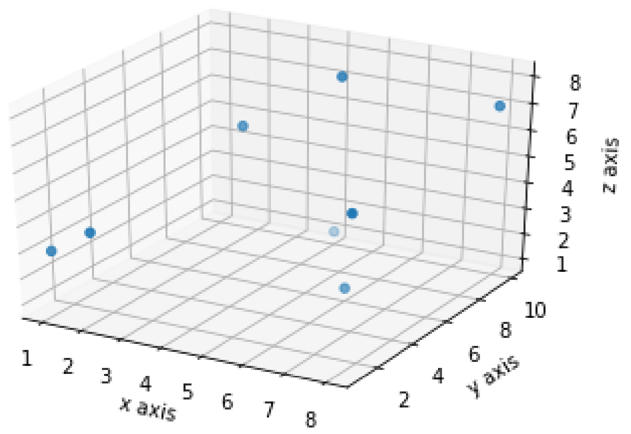
Increased Gaps

```
from mpl_toolkits.mplot3d import axes3d

fig = plt.figure()
ax1 = fig.add_subplot(111, projection = '3d')

x, y, z = axes3d.get_test_data()

ax1.plot_wireframe(x, y, z, rstride=7, cstride=7)

ax1.set_xlabel('x axis')
ax1.set_ylabel('y axis')
ax1.set_zlabel('z axis')
plt.legend()
plt
```



3D Scatter

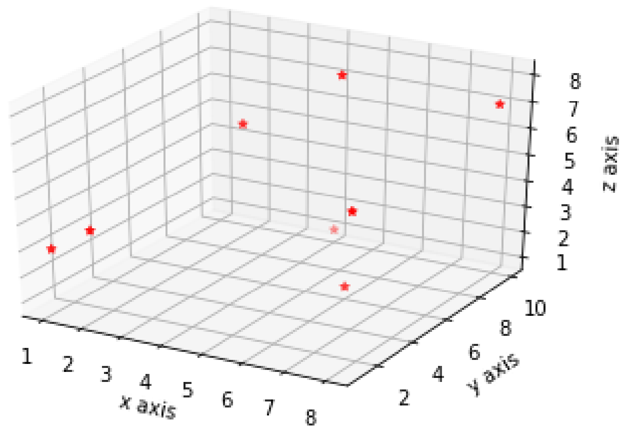
```
from mpl_toolkits.mplot3d import axes3d

fig = plt.figure()
ax1 = fig.add_subplot(111, projection = '3d')

x = [1,2,3,4,5,6,7,8]
y = [1,1,7,10,8,6,4,10]
z = [3,4,6,1,8,1,5,7]

ax1.scatter(x,y,z)
ax1.set_xlabel('x axis')
ax1.set_ylabel('y axis')
ax1.set_zlabel('z axis')

# THE DOTS CLOSEST TO YOU ARE DARKER (more alpha applied farther away)

plt
```



Custom Markers

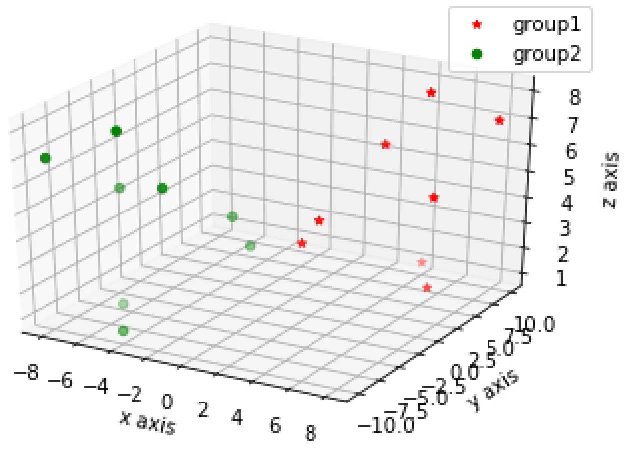
```
from mpl_toolkits.mplot3d import axes3d

fig = plt.figure()
ax1 = fig.add_subplot(111, projection = '3d')

x = [1,2,3,4,5,6,7,8]
y = [1,1,7,10,8,6,4,10]
z = [3,4,6,1,8,1,5,7]
```

```
ax1.scatter(x,y,z,c='r',marker='*')
ax1.set_xlabel('x axis')
ax1.set_ylabel('y axis')
ax1.set_zlabel('z axis')

plt
```



Groups / Clusters

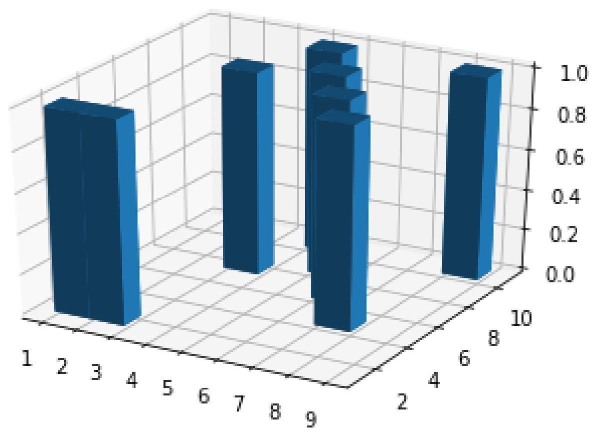
```
from mpl_toolkits.mplot3d import axes3d

fig = plt.figure()
ax1 = fig.add_subplot(111, projection = '3d')

x = [1,2,3,4,5,6,7,8]
y = [1,1,7,10,8,6,4,10]
z = [3,4,6,1,8,1,5,7]

x2 = [-1,-2,-3,-4,-5,-6,-7,-8]
y2 = [-1,-1,-7,-10,-8,-6,-4,-10]
z2 = [3,4,6,1,8,1,5,7]

ax1.scatter(x,y,z,c='r',marker='*',label='group1')
ax1.scatter(x2,y2,z2,c='g',marker='o',label='group2')
ax1.set_xlabel('x axis')
ax1.set_ylabel('y axis')
ax1.set_zlabel('z axis')

plt.legend()
plt
```



3D Bar Chart

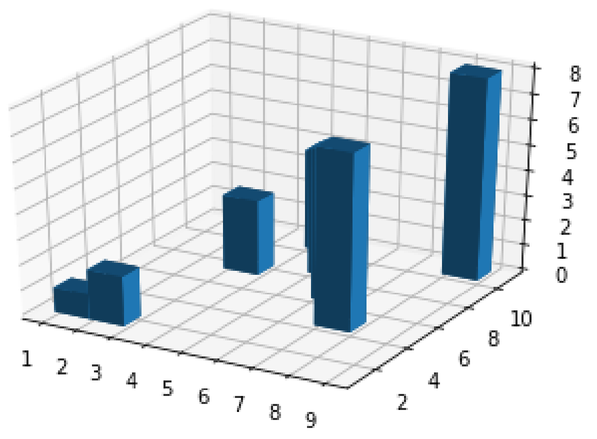
Scattered

```
from mpl_toolkits.mplot3d import axes3d

fig = plt.figure()
ax1 = fig.add_subplot(111, projection = '3d')

x = [1,2,3,4,5,6,7,8]
y = [1,1,7,10,8,6,4,10]
z = [0,0,0,0,0,0,0,0]

dx = 1
dy = 1
dz = 1

ax1.bar3d(x, y, z, dx, dy, dz)

plt
```



Scattered with Height

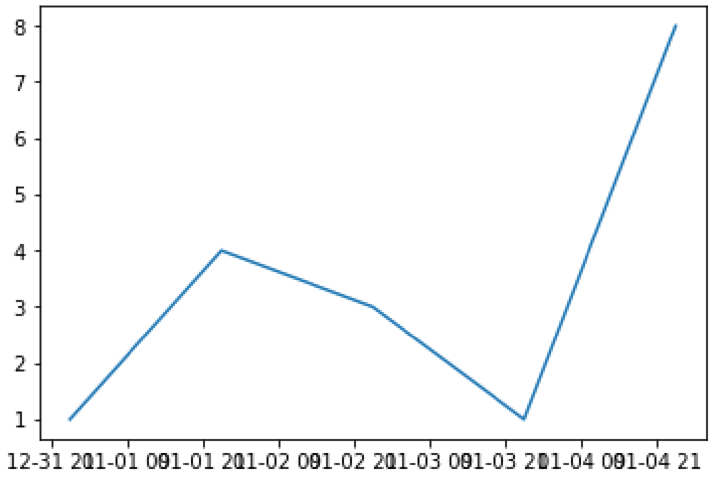
```
from mpl_toolkits.mplot3d import axes3d

fig = plt.figure()
ax1 = fig.add_subplot(111, projection = '3d')

x = [1,2,3,4,5,6,7,8]
y = [1,1,7,10,8,6,4,10]
z = [0,0,0,0,0,0,0,0]

dx = [1,1,1,1,1,1,1,1] #length of bars
dy = [1,1,1,1,1,1,1,1] #width of bars
dz = [1,2,3,4,5,6,7,8] #height of bars

ax1.bar3d(x, y, z, dx, dy, dz)

plt
```



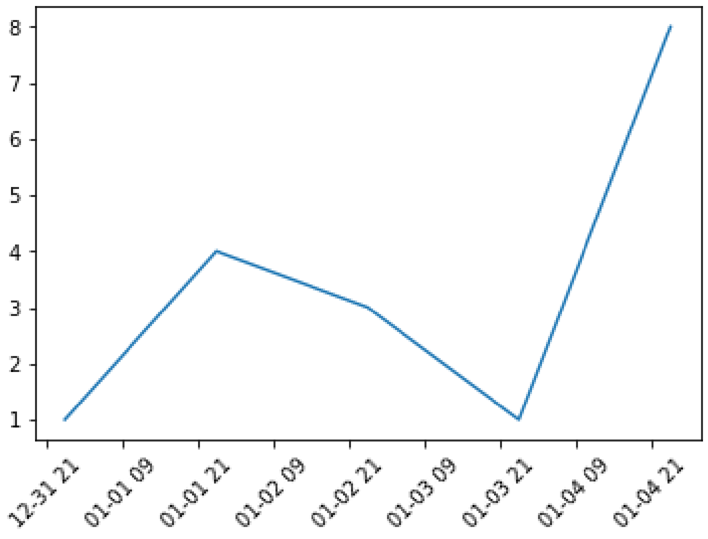
Advanced Techniques

Plotting Dates

```
# You must first convert your timestamps to Python datetime objects (use datetime.strptime).
# Then use date2num to convert the dates to matplotlib format.
#
# Plot the dates and values using plot_date...
# dates = matplotlib.dates.date2num(list_of_datetimes) matplotlib.pyplot.plot_date(dates,
values)

from datetime import datetime

x = ['2001-01-01', '2001-01-02', '2001-01-03', '2001-01-04', '2001-01-05']
x_datetimes = [datetime.strptime(date, '%Y-%m-%d') for date in x]
x_matplotlib_datetimes = matplotlib.dates.date2num(x_datetimes)

y = [1,4,3,1,8]
plt.plot_date(x_matplotlib_datetimes, y, '-')
plt
```

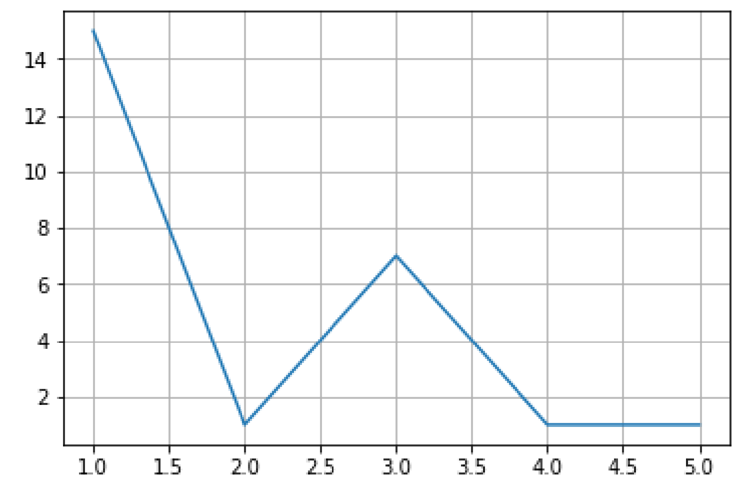


```
from datetime import datetime

x = ['2001-01-01', '2001-01-02', '2001-01-03', '2001-01-04', '2001-01-05']
x_datetimes = [datetime.strptime(date, '%Y-%m-%d') for date in x]
x_matplotlib_datetimes = matplotlib.dates.date2num(x_datetimes)

# For wahtever reason this must be above call to plot_date...
# Otherwise 2 graphs will show, the second one will be empty
# and have slanted x labels, but the first will have the data
# but horizontal labels.
fig = plt.figure()
ax1 = plt.subplot2grid((1,1), (0,0))
for label in ax1.xaxis.get_ticklabels():
    label.set_rotation(45)

y = [1,4,3,1,8]
plt.plot_date(x_matplotlib_datetimes, y, '-')

plt
```

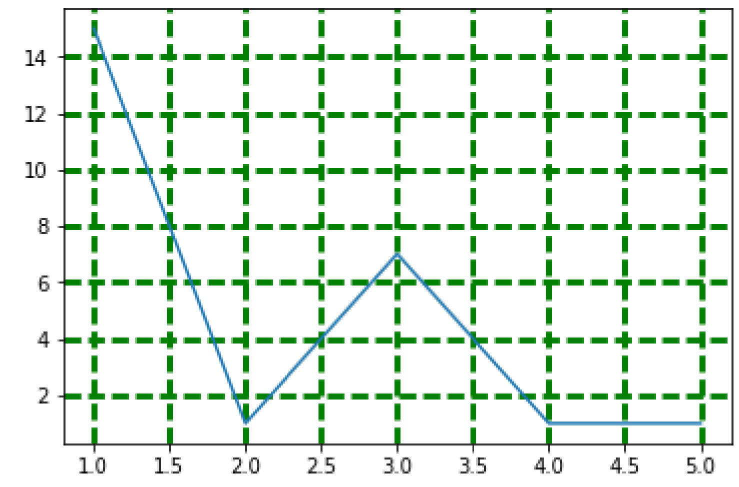


Adding Grids

```
x = [1,2,3,4,5]
y = [15,1,7,1,1]
fig = plt.figure()
ax1 = plt.subplot2grid((1,1), (0,0))
ax1.grid(True)
plt.plot(x, y)
plt
```



x = [1,2,3,4,5]
y = [15,1,7,1,1]
fig = plt.figure()
ax1 = plt.subplot2grid((1,1), (0,0))
ax1.grid(True, color='g', linestyle='--', linewidth=3)
plt.plot(x, y)
plt

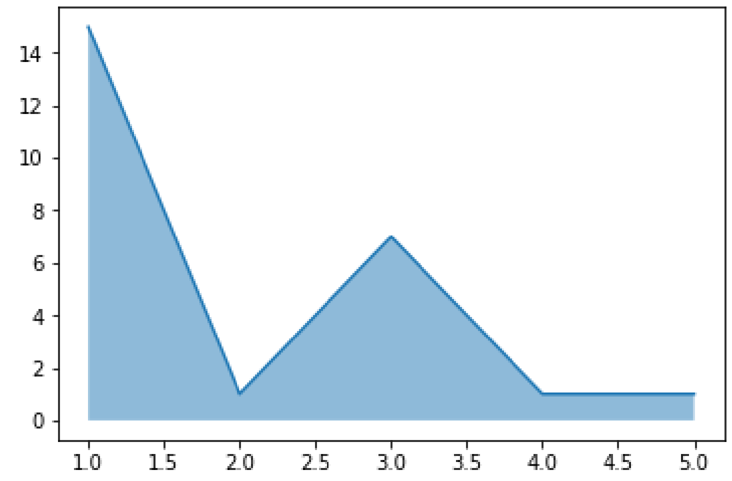


Filling Plots

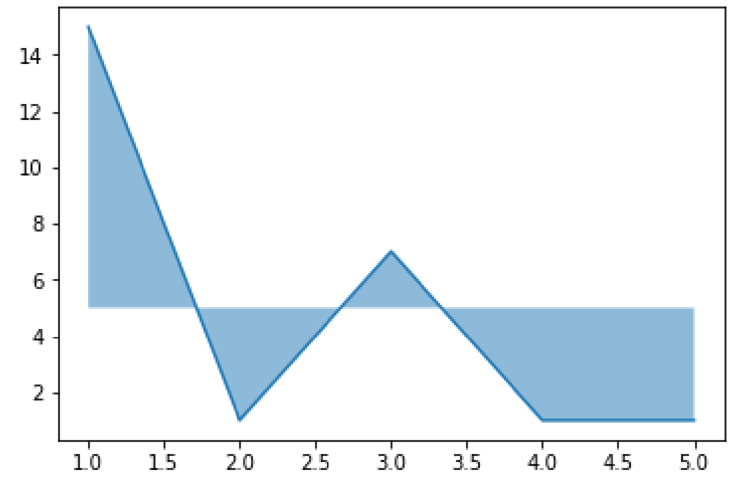
x = [1,2,3,4,5]
y = [15,1,7,1,1]

fig = plt.figure()
ax1 = plt.subplot2grid((1,1), (0,0))
ax1.fill_between(x, y, 0, alpha=0.5)

```
plt.plot(x, y)

plt
```

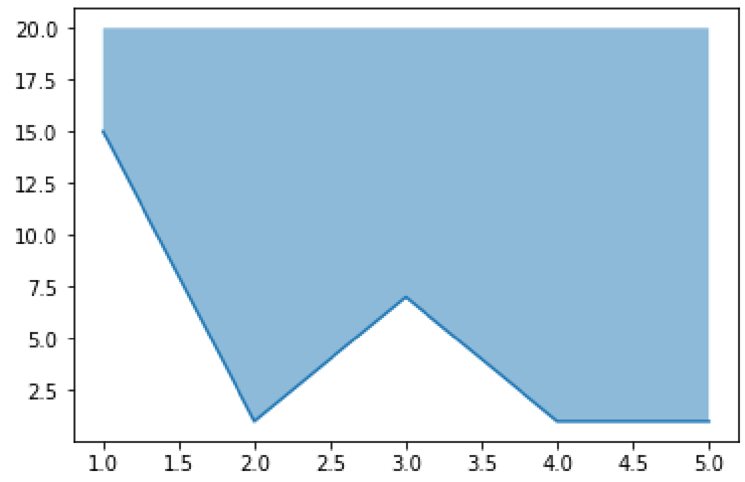


```
x = [1,2,3,4,5]
y = [15,1,7,1,1]

fig = plt.figure()
ax1 = plt.subplot2grid((1,1), (0,0))
ax1.fill_between(x, y, 5, alpha=0.5)

plt.plot(x, y)

plt
```

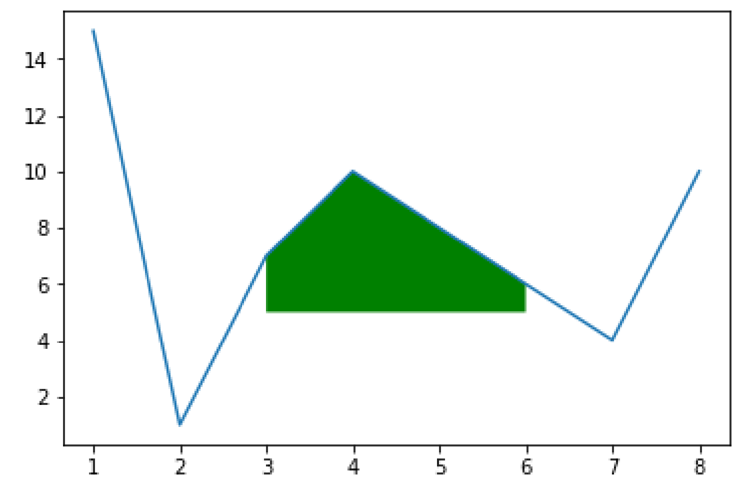


```
x = [1,2,3,4,5]
y = [15,1,7,1,1]

fig = plt.figure()
```

```
ax1 = plt.subplot2grid((1,1), (0,0))
ax1.fill_between(x, y, 20, alpha=0.5)

plt.plot(x, y)

plt
```



```
import numpy as np

x = np.array([1,2,3,4,5,6,7,8])
y = np.array([15,1,7,10,8,6,4,10])

fig = plt.figure()
ax1 = plt.subplot2grid((1,1), (0,0))
plt.plot(x, y)
ax1.fill_between(x, y, 5, where=(y >= 5), facecolor='g')
ax1.fill_between(x, y, 5, where=(y <= 5), facecolor='r')

plt
```



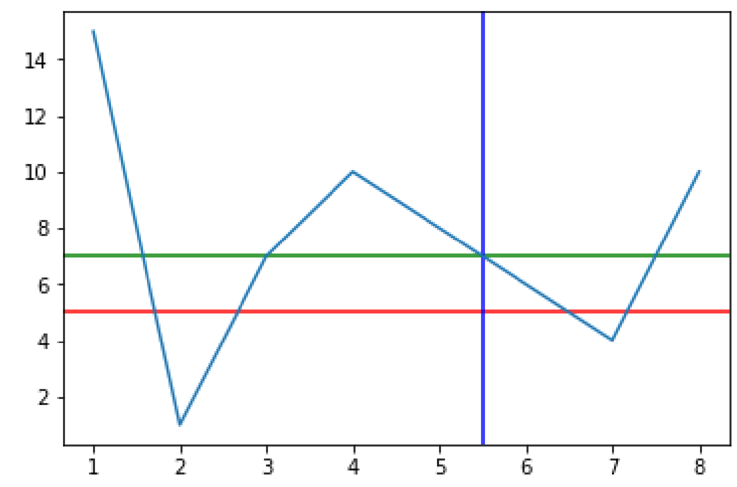
NOTE: This last one is totally broken. Not sure why. Everything above 5 should be green, everything blow should be red.

Drawing Arbitrary Lines

x =[1,2,3,4,5,6,7,8]
y = [15,1,7,10,8,6,4,10]

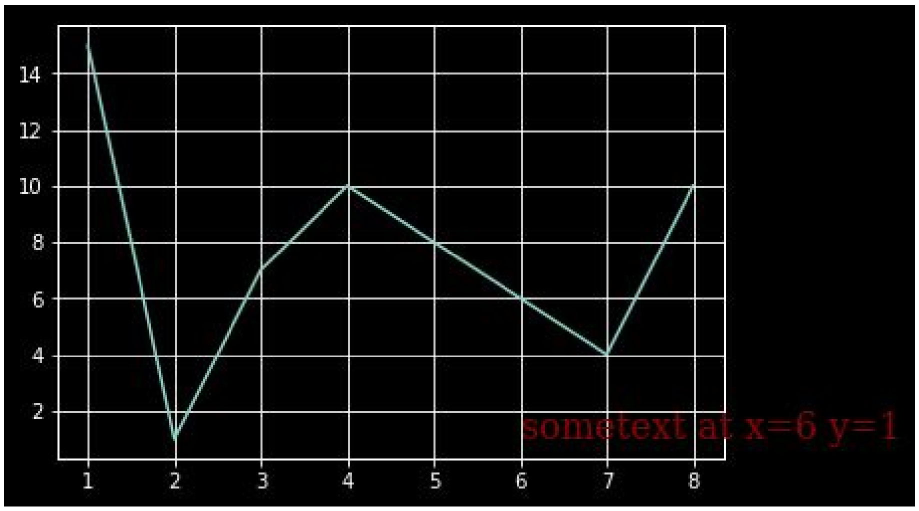
fig = plt.figure()
ax1 = plt.subplot2grid((1,1), (0,0))
ax1.axhline(5, color='r')
ax1.axhline(7, color='g')
ax1.axvline(5.5, color='b')
plt.plot(x, y)

plt



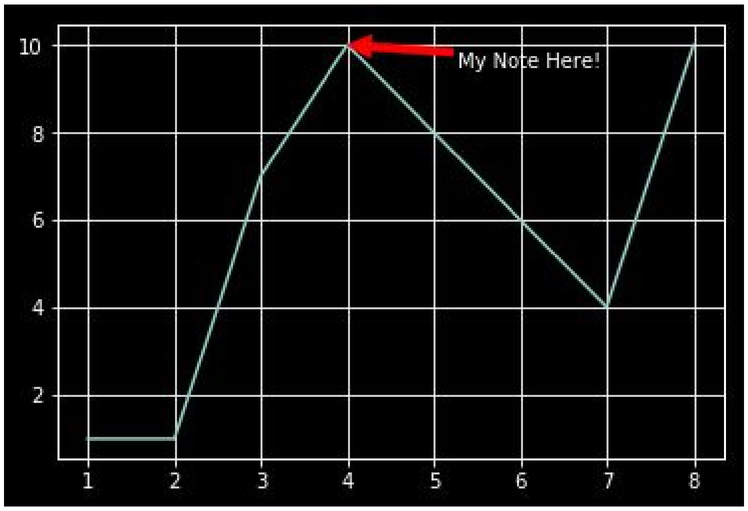
Drawing Arbitrary Text

```
x =[1,2,3,4,5,6,7,8]
y = [15,1,7,10,8,6,4,10]

fig = plt.figure()
ax1 = plt.subplot2grid((1,1), (0,0))
plt.plot(x, y)

font = {'family' : 'serif',
        'color' : 'darkred',
        'size' : 18}
plt.text(6, 1, "sometext at x=6 y=1", fontdict=font)

plt
```



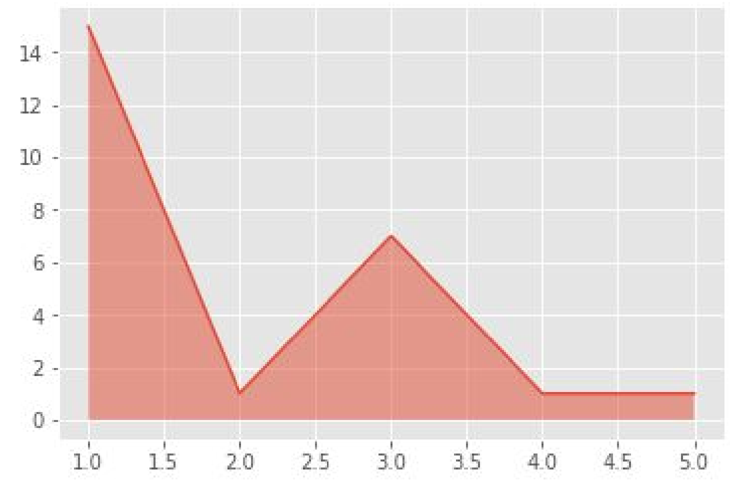
Annotating

```
import numpy as np

x = [1,2,3,4,5,6,7,8]
y = [1,1,7,10,8,6,4,10]

fig = plt.figure()
ax1 = plt.subplot2grid((1,1), (0,0))
ax1.annotate('My Note Here!',
             (x[3], y[3]),
             xytext=(0.6, 0.9),
             textcoords='axes fraction',
            arrowprops=dict(facecolor='r', color='r'))
plt.plot(x, y)

plt
```

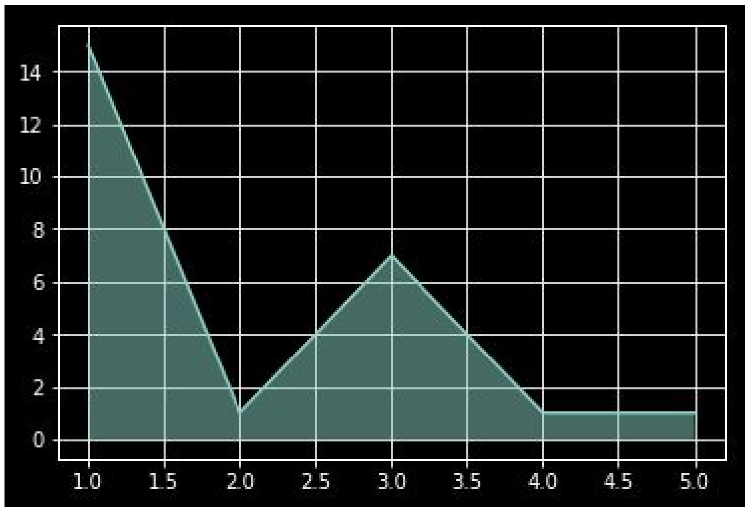


Using Styles

```
from matplotlib import style

style.use('ggplot')

x = [1,2,3,4,5]
y = [15,1,7,1,1]

fig = plt.figure()
ax1 = plt.subplot2grid((1,1), (0,0))
ax1.fill_between(x, y, 0, alpha=0.5)

plt.plot(x, y)

plt
```



```
from matplotlib import style

style.use('dark_background')

x = [1,2,3,4,5]
y = [15,1,7,1,1]

fig = plt.figure()
ax1 = plt.subplot2grid((1,1), (0,0))
ax1.fill_between(x, y, 0, alpha=0.5)

plt.plot(x, y)

plt
```



14
12
10
8
6
4
2
0
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0