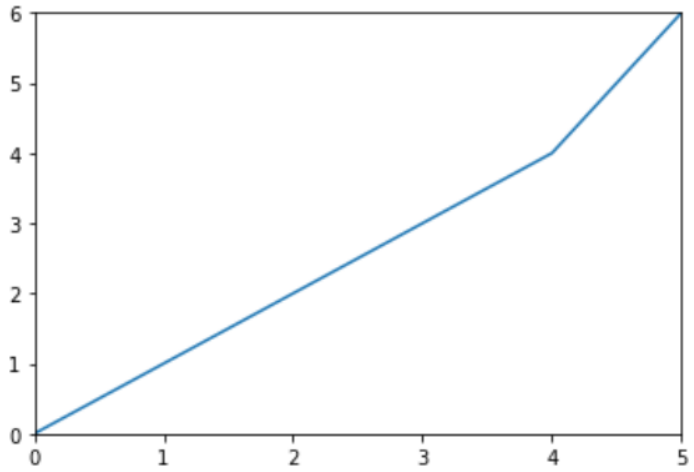
Line Graphs and Graphing Functions in Python

We'll use Anaconda, Python 3, and Jupyter Notebook for the examples. To get started, we need to import a math package and a plotting package.

```
import math
from matplotlib import pyplot
```

To create a line graph in Python, we can use pyplot's plot command. We need to list inside square brackets the points to plot. List the x-coordinates in the first set of square brackets, and the y-coordinates in the second set in the same order. So, each plotted point is $(x[i], y[i])$ or (x_i, y_i) where i is the corresponding position in the vector. Each of the points is connected with a straight line from the previous point.

```
pyplot.plot([0,1,2,3,4,5],[0,1,2,3,4,6])
pyplot.axis([0,5,0,6])
pyplot.show()
```

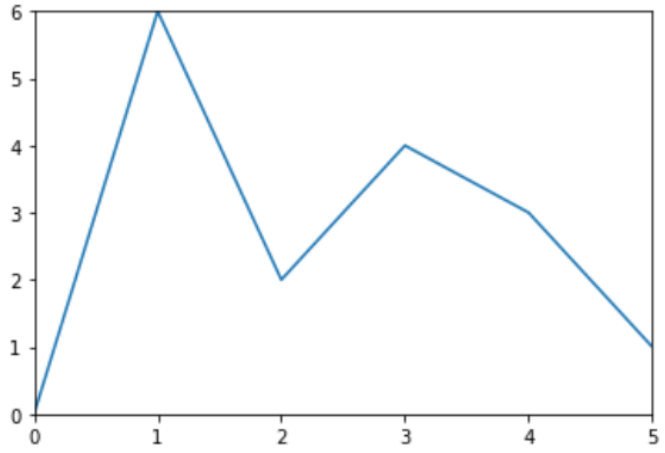


The second line under the set of points sets the axis range. The order of the input, again in vector form enclosed in square brackets is $[xmin, xmax, ymin, ymax]$.

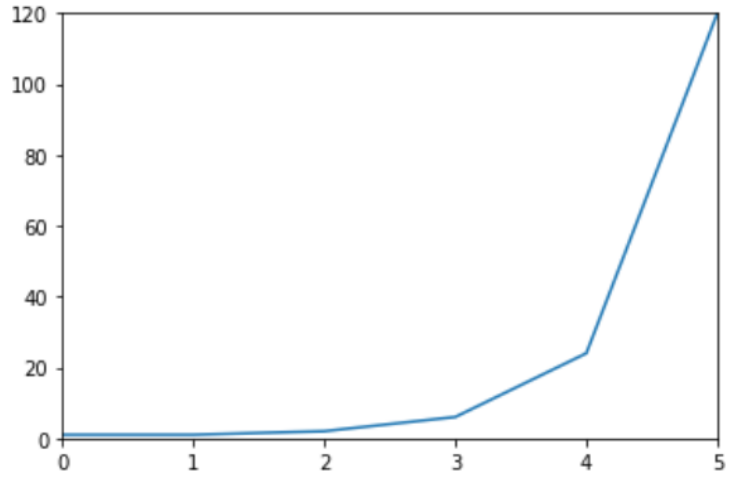
The last line is to display the graph (otherwise, Python will calculate it, but not display it).

If we change the coordinate points of y into a random order, we can see more clearly what is happening.

```
pyplot.plot([0,1,2,3,4,5],[0,6,2,4,3,1])
pyplot.axis([0,5,0,6])
pyplot.show()
```



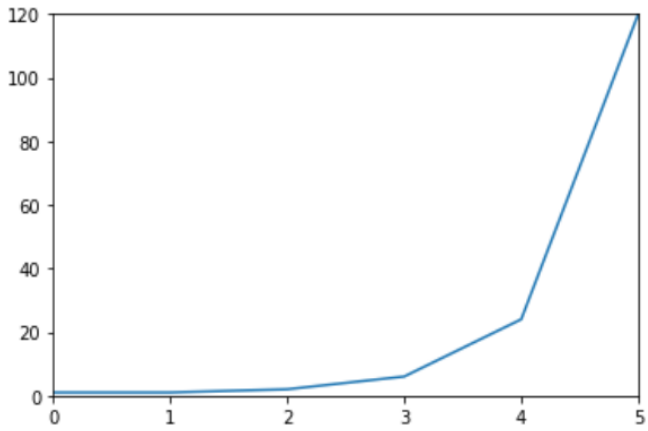

```
pyplot.plot([0,1,2,3,4,5],[1,1,2,6,24,120])
pyplot.axis([0,5,0,120])
pyplot.show()
```



The graph above shows the factorial function. That function is discrete points, not a continuous curve. But what if we wanted to get a smoother curve? Then we need to plot more points so that the straight edges between points is less noticeable. Rather than write out all the points by hand, we can set up a loop to build the vectors for us.

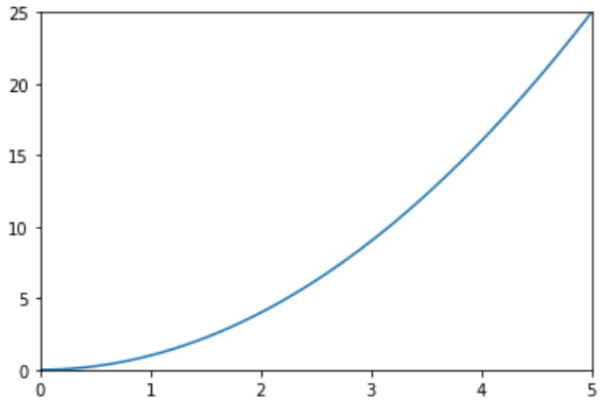
Let's start with a simple case, like the factorial graph above, and then work on the smoothness. Since the x-coordinates are in sequential order. The range function counts in discrete values starting with the first number and stopping (not including) the last number. Thus, the $range(0,6)$ produces the list $[0,1,2,3,4,5]$. The for loop generates the y list based on the factorial formula, building the components of the y vector value by value. And then we plot the two lists as before.

```
xlist=range(0,6)
ylist=[]
for i in xlist:
    ylist.append(math.factorial(i))
pyplot.plot(xlist,ylist)
pyplot.axis([0,5,0,120])
pyplot.show()
```



This is still too few points to make the graph look smooth, so we'll adapt this to obtain the $y = x^2$ graph.

```
ilist=range(0,51)
xlist=[]
ylist=[]
for i in ilist:
    xlist.append(i/10)
    ylist.append((i*i)/100)
pyplot.plot(xlist,ylist)
pyplot.axis([0,5,0,25])
pyplot.show()
```

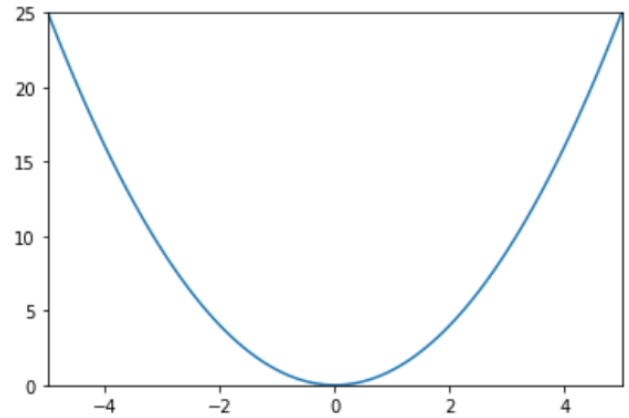


Here, we generate the x list based on the i values, generating a list that goes by $\frac{1}{10}$ units. Then the $x = \frac{i}{10}$ are used to calculate y: $y = x^2$ becomes $y = \left(\frac{i}{10}\right)^2 = \frac{i^2}{100}$. The graph looks nice and smooth now.

Examples below are shown for various functions and ranges. Note that if your graph doesn't start at 0, you have to shift your x values to account for it.

Example 2. $y = x^2$ from $[-5,5]$.

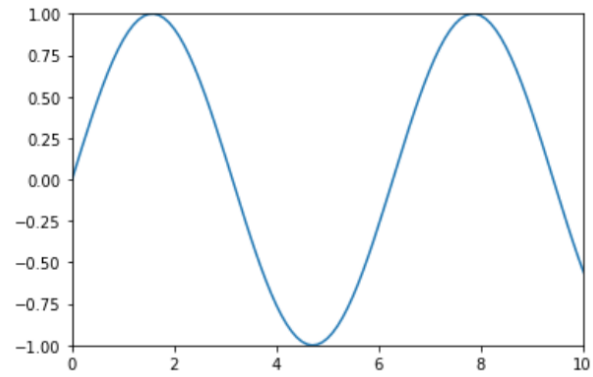
```
ilist=range(0,101)
xlist=[]
ylist=[]
for i in ilist:
    xlist.append(-5+i/10)
    ylist.append((i*i)/100-i+25)
pyplot.plot(xlist,ylist)
pyplot.axis([-5,5,0,25])
pyplot.show()
```



x^2 based on i is $\left(\frac{i}{10} - 5\right)^2 = \frac{i^2}{100} - i + 25$.

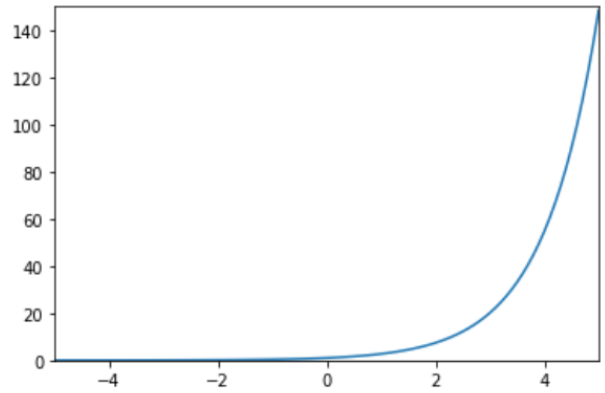
Example 3. Sine function $y = \sin(x)$ in $[0,10]$.

```
ilist=range(0,201)
xlist=[]
ylist=[]
for i in ilist:
    xlist.append(i/10)
    ylist.append(math.sin(i/10))
pyplot.plot(xlist,ylist)
pyplot.axis([0,10,-1,1])
pyplot.show()
```



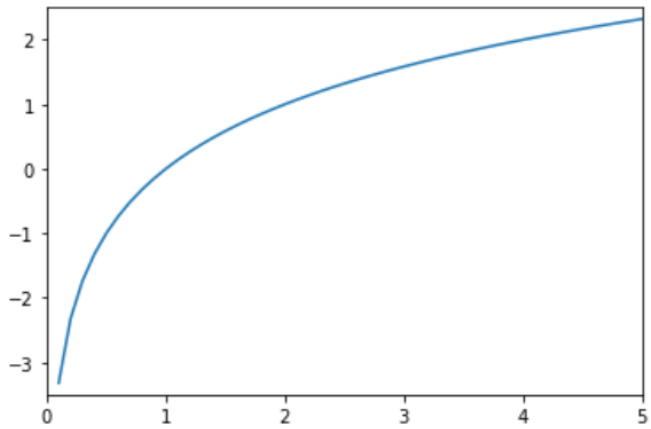
Example 4. The exponential function $y = e^x$ on the interval $[-5,5]$.

```
ilist=range(0,101)
xlist=[]
ylist=[]
for i in ilist:
    xlist.append(-5+i/10)
    ylist.append(math.exp(-5+i/10))
pyplot.plot(xlist,ylist)
pyplot.axis([-5,5,0,150])
pyplot.show()
```



Example 5. Log base two. $y = \log_2 x$ on the interval $[\frac{1}{10}, 5]$.

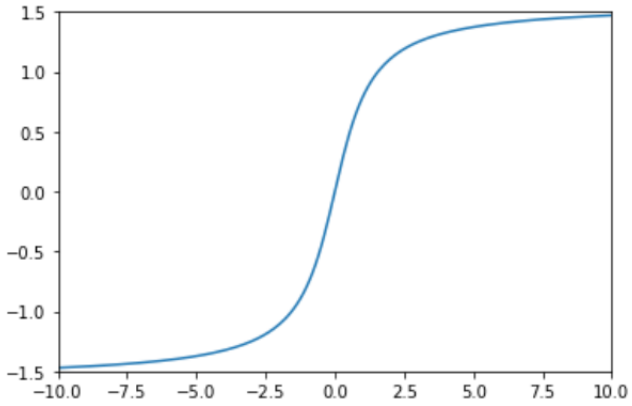
```
ilist=range(1,51)
xlist=[]
ylist=[]
for i in ilist:
    xlist.append(i/10)
    ylist.append(math.log2(i/10))
pyplot.plot(xlist,ylist)
pyplot.axis([0,5,-3.5,2.5])
pyplot.show()
```



If you don't want to worry about the horizontal shifts, you can also adjust the range of i to include negative values.

Example 6. Inverse tangent function. $y = \arctan(x)$ on $[-10,10]$.

```
ilist=range(-100,101)
xlist=[]
ylist=[]
for i in ilist:
    xlist.append(i/10)
    ylist.append(math.atan(i/10))
pyplot.plot(xlist,ylist)
pyplot.axis([-10,10,-1.5,1.5])
pyplot.show()
```



Functions that can be graphed from the math library include:

'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'cos', 'cosh', 'e', 'exp', 'factorial', 'floor', 'gamma', 'log', 'log10', 'log2', 'sin', 'sinh', 'sqrt', 'tan', 'tanh'.

I've used $\frac{i}{10}$ as the ratio for a standard graph. If your domain is quite large, then i by itself will probably be fine. If you need to zoom in, then make the ratio smaller if the graph has too many sharp edges.