(http://www.pieriandata.com)

Matplotlib Figure Object

Import the `matplotlib.pyplot` module under the name `plt` (the tidy way):

In [1]:
```
# COMMON MISTAKE!
# DON'T FORGET THE .PYPLOT part

import matplotlib.pyplot as plt
```

NOTE: For users running .py scripts in an IDE like PyCharm or Sublime Text Editor. You will not see the plots in a notebook, instead if you are using another editor, you'll use: *plt.show()* at the end of all your plotting commands to have the figure pop up in another window.

Matplotlib Object Oriented Method

Now that we've seen the basics, let's break it all down with a more formal introduction of Matplotlib's Object Oriented API. This means we will instantiate figure objects and then call methods or attributes from that object.

The Data

In [12]:
```
import numpy as np
```

In [50]:
```
a = np.linspace(0,10,11)
b = a ** 4
```

In [51]:
```
a
```

Out[51]:
```
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10.])
```

In [52]:
```
b
```

Out[52]:
```
array([0.000e+00, 1.000e+00, 1.600e+01, 8.100e+01, 2.560e+02, 6.250e+02,
       1.296e+03, 2.401e+03, 4.096e+03, 6.561e+03, 1.000e+04])
```

In [53]:
```
x = np.arange(0,10)
y = 2 * x
```

In [54]:
```
x
```

Out[54]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [55]:
```
y
```

Out[55]:
```
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])
```

Creating a Figure

The main idea in using the more formal Object Oriented method is to create figure objects and then just call methods or attributes off of that object. This approach is nicer when dealing with a canvas that has multiple plots on it.

In [73]:
```
# Creates blank canvas
fig = plt.figure()
```

```
<Figure size 432x288 with 0 Axes>
```

NOTE: ALL THE COMMANDS NEED TO GO IN THE SAME CELL!

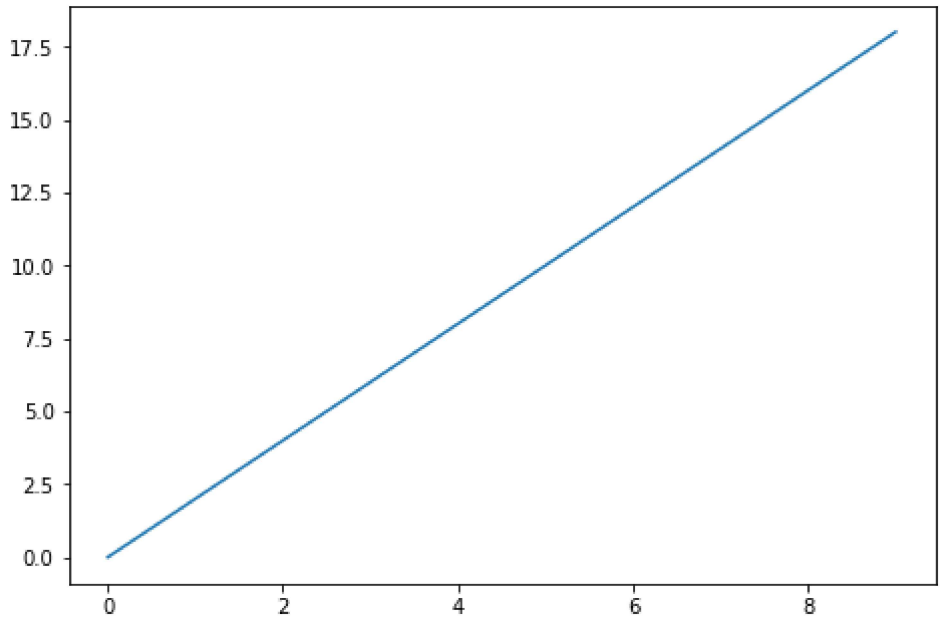
To begin we create a figure instance. Then we can add axes to that figure:

In [56]:
```
# Create Figure (empty canvas)
fig = plt.figure()

# Add set of axes to figure
axes = fig.add_axes([0, 0, 1, 1]) # left, bottom, width, height (range 0 to

# Plot on that set of axes
axes.plot(x, y)

plt.show()
```

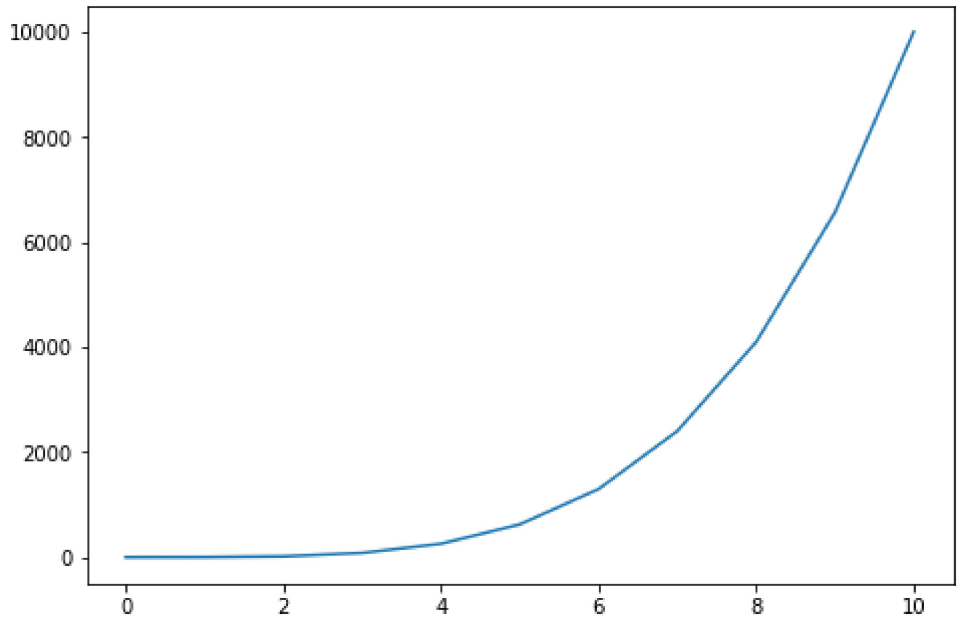


In [57]:
```
# Create Figure (empty canvas)
fig = plt.figure()

# Add set of axes to figure
axes = fig.add_axes([0, 0, 1, 1]) # left, bottom, width, height (range 0 to

# Plot on that set of axes
axes.plot(a, b)

plt.show()
```



Adding another set of axes to the Figure

So far we've only seen one set of axes on this figure object, but we can keep adding new axes on to it at any location and size we want. We can then plot on that new set of axes.

In [58]:
```
type(fig)
```

Out[58]:
```
matplotlib.figure.Figure
```

Code is a little more complicated, but the advantage is that we now have full control of where the plot axes are placed, and we can easily add more than one axis to the figure. Note how we're plotting a,b twice here

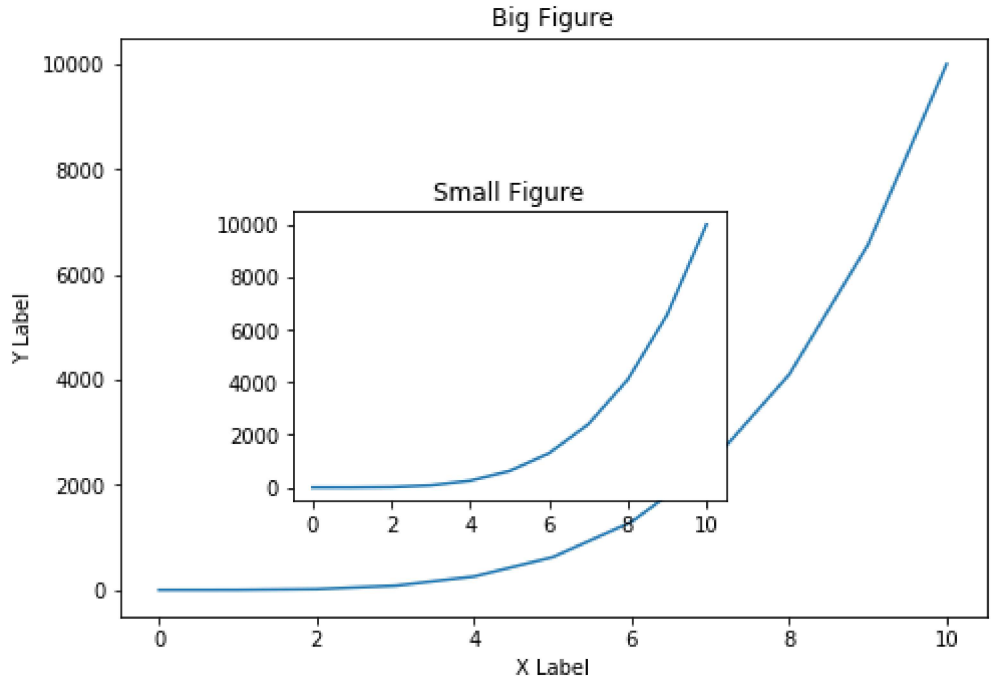
In [62]:
```
# Creates blank canvas
fig = plt.figure()

axes1 = fig.add_axes([0, 0, 1, 1]) # Large figure
axes2 = fig.add_axes([0.2, 0.2, 0.5, 0.5]) # Smaller figure

# Larger Figure Axes 1
axes1.plot(a, b)

# Use set_ to add to the axes figure
axes1.set_xlabel('X Label')
axes1.set_ylabel('Y Label')
axes1.set_title('Big Figure')

# Insert Figure Axes 2
axes2.plot(a,b)
axes2.set_title('Small Figure');
```



Let's move the small figure and edit its parameters.

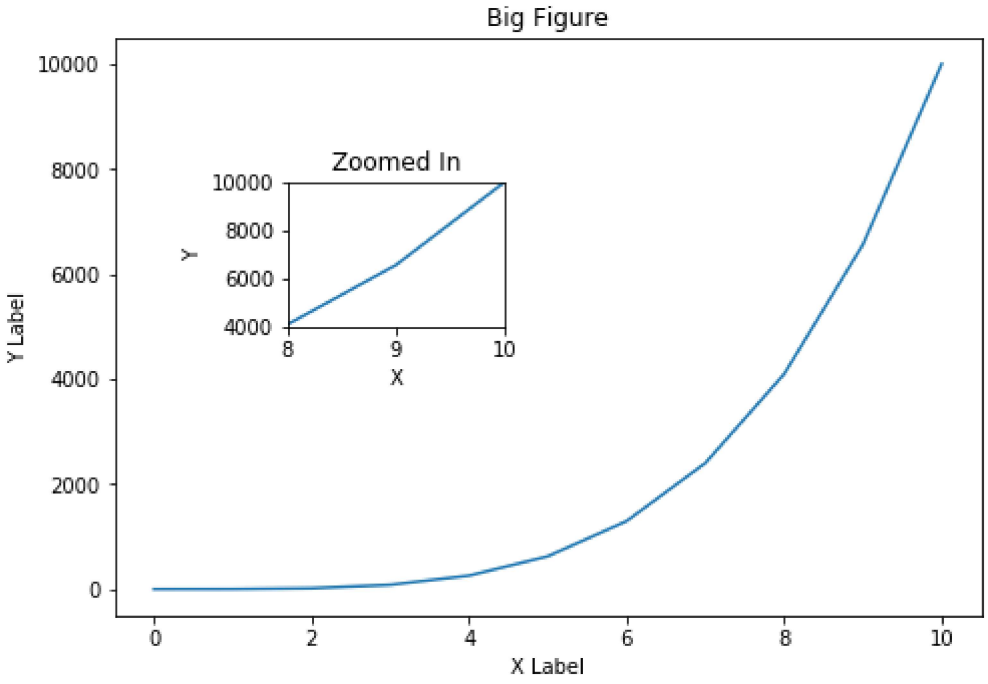
In [69]:
```
# Creates blank canvas
fig = plt.figure()

axes1 = fig.add_axes([0, 0, 1, 1]) # Large figure
axes2 = fig.add_axes([0.2, 0.5, 0.25, 0.25]) # Smaller figure

# Larger Figure Axes 1
axes1.plot(a, b)

# Use set_ to add to the axes figure
axes1.set_xlabel('X Label')
axes1.set_ylabel('Y Label')
axes1.set_title('Big Figure')

# Insert Figure Axes 2
axes2.plot(a,b)
axes2.set_xlim(8,10)
axes2.set_ylim(4000,10000)
axes2.set_xlabel('X')
axes2.set_ylabel('Y')
axes2.set_title('Zoomed In');
```



You can add as many axes on to the same figure as you want, even outside of the main figure if the length and width correspond to this.

In [74]:
```
# Creates blank canvas
fig = plt.figure()

axes1 = fig.add_axes([0, 0, 1, 1]) # Full figure
axes2 = fig.add_axes([0.2, 0.5, 0.25, 0.25]) # Smaller figure
axes3 = fig.add_axes([1, 1, 0.25, 0.25]) # Starts at top right corner!

# Larger Figure Axes 1
axes1.plot(a, b)

# Use set_ to add to the axes figure
axes1.set_xlabel('X Label')
axes1.set_ylabel('Y Label')
axes1.set_title('Big Figure')

# Insert Figure Axes 2
axes2.plot(a,b)
axes2.set_xlim(8,10)
axes2.set_ylim(4000,10000)
axes2.set_xlabel('X')
axes2.set_ylabel('Y')
axes2.set_title('Zoomed In');

# Insert Figure Axes 3
axes3.plot(a,b)
```

Out[74]:
```
[<matplotlib.lines.Line2D at 0x1cd42ad2888>]
```

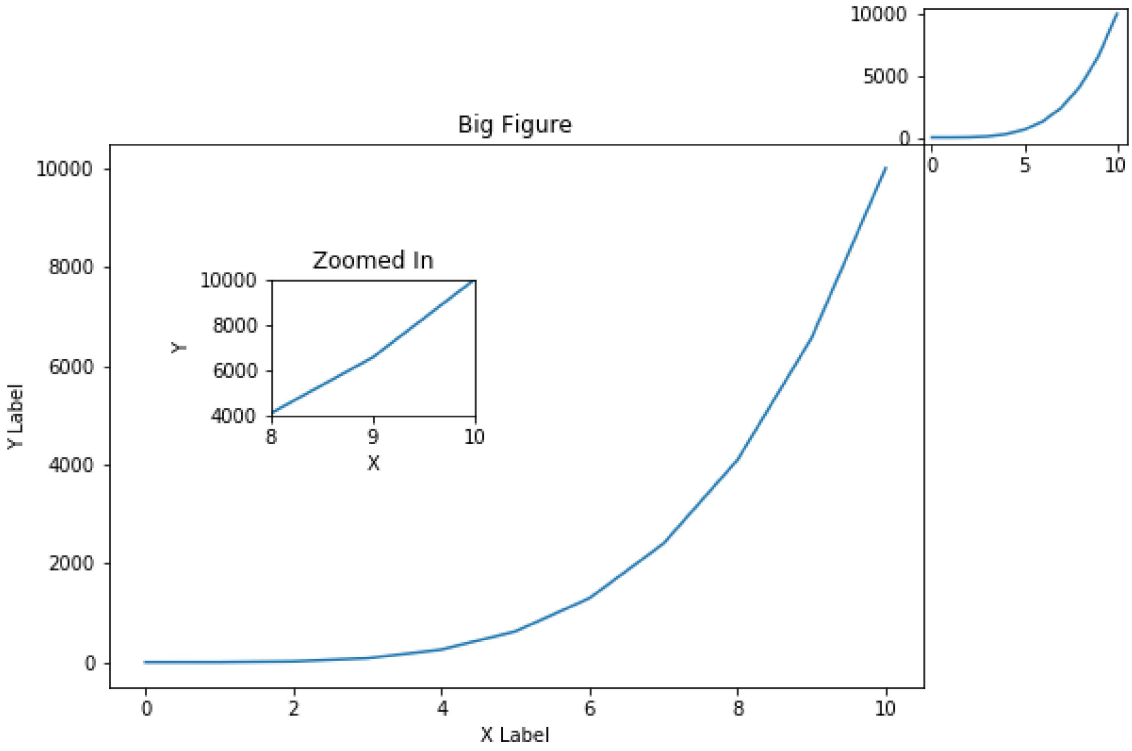


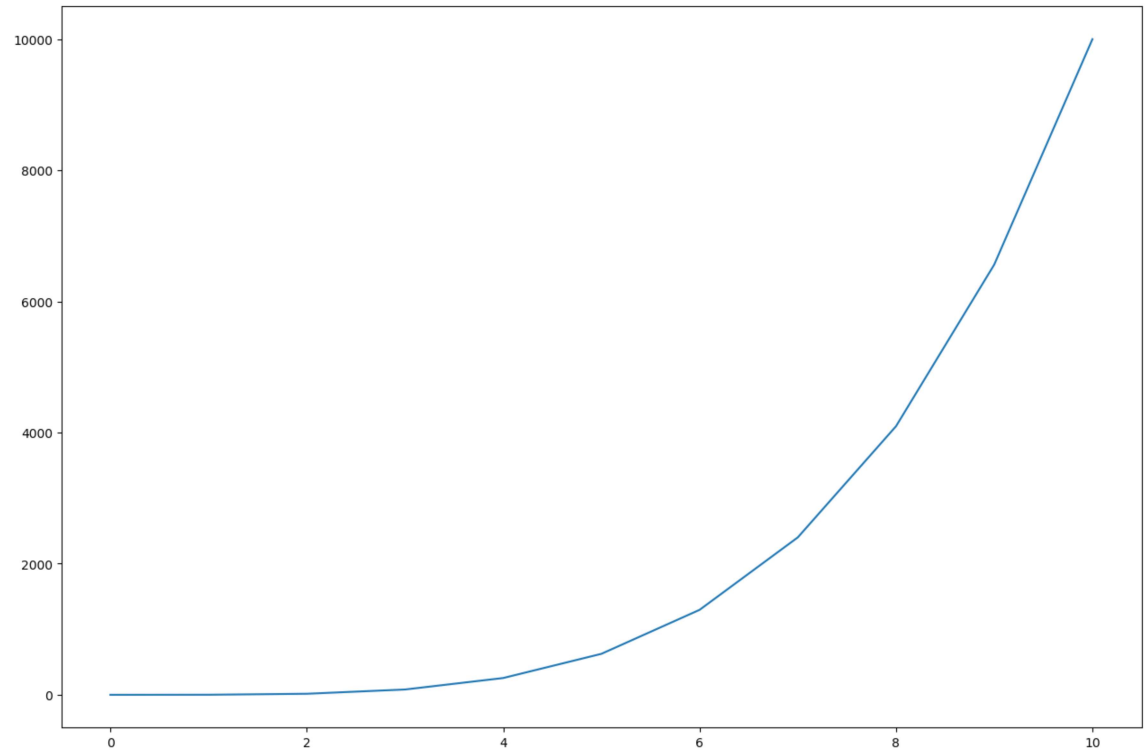
Figure Parameters

In [82]:
```
# Creates blank canvas
fig = plt.figure(figsize=(12,8),dpi=100)

axes1 = fig.add_axes([0, 0, 1, 1])

axes1.plot(a,b)
```

Out[82]:
```
[<matplotlib.lines.Line2D at 0x1cd42d53848>]
```



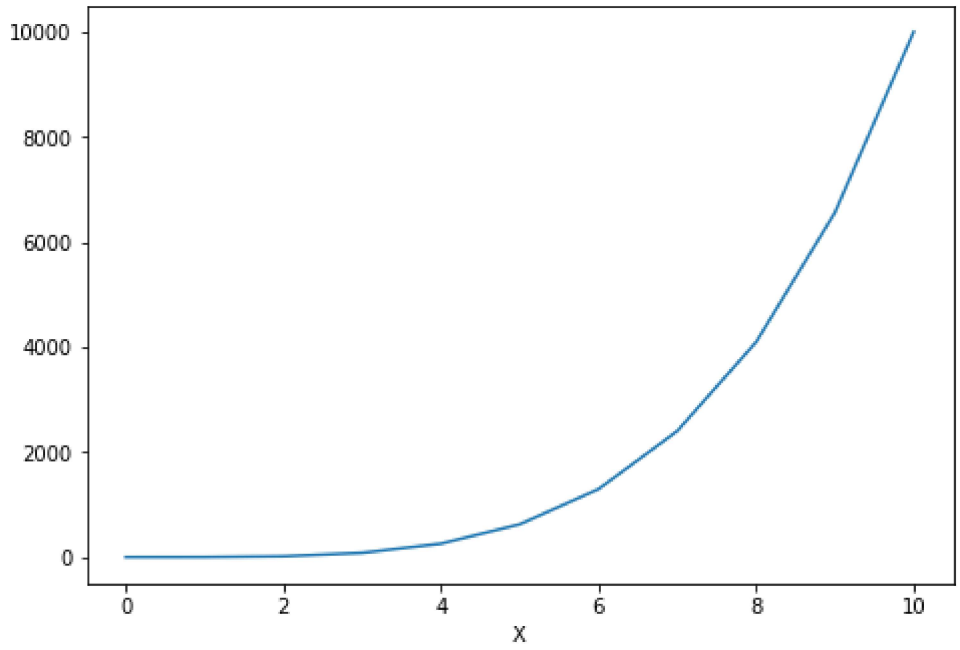
Exporting a Figure

In [95]:
```
fig = plt.figure()

axes1 = fig.add_axes([0, 0, 1, 1])

axes1.plot(a,b)
axes1.set_xlabel('X')

# bbox_inches ='tight' automatically makes sure the bounding box is correct
fig.savefig('figure.png',bbox_inches='tight')
```



In [112]:
```
# Creates blank canvas
fig = plt.figure(figsize=(12,8))

axes1 = fig.add_axes([0, 0, 1, 1]) # Full figure
axes2 = fig.add_axes([1, 1, 0.25, 0.25]) # Starts at top right corner!

# Larger Figure Axes 1
axes1.plot(x,y)

# Insert Figure Axes 2
axes2.plot(x,y)

fig.savefig('test.png',bbox_inches='tight')
```