Lets plot two data points and fit a line to them

- Taking two points and figuring out the slope and intercept is easy. You have been doing this for years. You do rise over run and you figure it all out. But now you need to do it in Python. Switching your brain to do this in Python is the hard part. You need to translate one method to another. It takes time and different people get it at different rates. So be patient, learn, and help each other!
- For this everyone needs to make their own plot but before you begin work in groups of about 4 people to figure it out.
- When you have data you usually plot the data as points on a graph and then you plot the best fit line to that data. We are going to do something simple and plot two points and then fit the straight line.
- Rember that the equation of a line is

y=mx+b

- Two points define a line!
- If you had two points in a class, you would graph them, fit a line, and report the equation. We are going to do it in python! It is slightly different
- If you have two points they would be stored in a list or an array. We are going to transition to only using numpy arrays or other types of arrays for data.
    - I call this list x_data and y_data. Remember you use your list notation to get the data
    - the first point on the graph is x_data[0],y_data[0]
    - the second point is x_data[1],y_data[1]
    - Remember we have said nothing about the values of x_data and y_data just how the computer is storing them
- We could solve the equation now since we have two equations and two unknowns. you would get

y_data[0]=m*x_data[0]+b This is similar to writing y$_0$=mx_0+b

y_data[1]=m*x_data[1]+b This is similar to writing y$_1$=mx_1+b

- With these two equations and two unknowns you can solve for m and b. So you should solve for m and b and get the equations you will use.

- Now that you have m and b you can then calculate the fit. I call these x_fit and y_fit.
- What function can we use to generate a set of numbers for x_fit where they are not all integers and you have floats?
- Then we plot the data. I use
  - ax.scatter() for the data as this shows just points.
  - If you add the keyword argument label='' then it gives a label to that part of the plot that can later show up in a legend. I had this in the answers from last time.
  - I used ax.plot() like usual to plot the line. But I added the label='' keyword.
  - I label the axes.
  - I turn on the legend with ax.legend(loc='best')
  - When I get fancy I can add a title that includes the equation of the line. We can do this by making a string with the format notation. The format notation is nice to control the number of decimal places. Remember this from your hw...

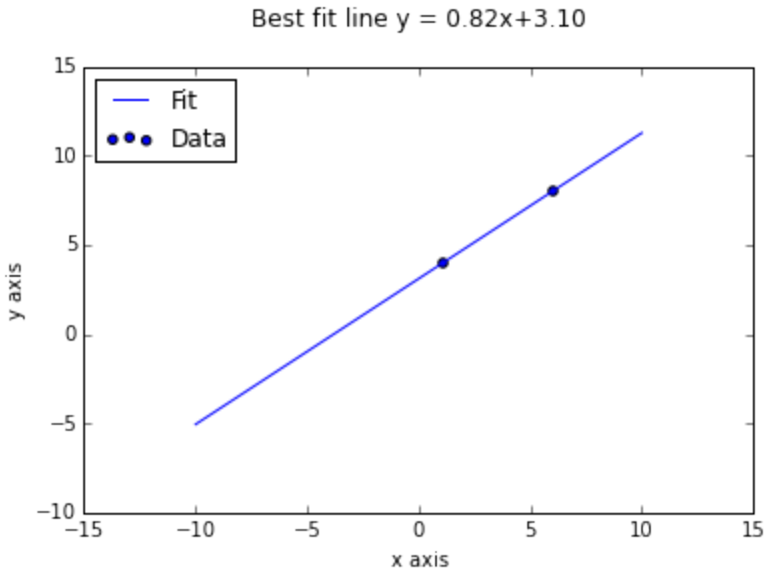
Fill it all in below and see how it goes! Make sure everyone in your group understands it all and makes a great plot.

In [ ]:
```
%matplotlib ipympl
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
#This is my data
x_data=np.array([1.1,6])
y_data=np.array([4,8])

#this is my solution to the line
m=
b=

#This is the best fit line
x_fit=np.
y_fit=

#this is plotting the data
fig,ax=plt.subplots()
```

In [1]:
```
### my answer
```

Out[1]:
```
<matplotlib.legend.Legend at 0xa668f98>
```


Best fit line y = 0.82x+3.10
Fit
Data
y axis
x axis
15
10
5
0
−5
−10
−15
−10
−5
0
5
10
15

In [ ]: