Enrique's Guide to uploading data, processing data and graphing data in Jupyter Notebooks

by Enrique Arce-Larreta, West High School, Salt Lake City, Utah, enrique.arce-larreta@slcschools.org

As part of this course, we will be conducting various experiments throughout the year. Variables will be analyzed and plotted as part of the course. The International Baccalaureate program requires you to conduct your own experiment (the IA), take measurements, process (make calculations) the data, and finally present your data (in a graph) from which you draw conclusions.

As such, it is useful to familiarize yourself with plotting and processing software, such as spreadsheets or other software. Jupyter is a free online platform (no installs necessary) that allows you to process and present data. This is especially good for large datasets, fyi.

This tutorial notebook walks you through how to:

- import data into the Notebook
- process or manipulate data (make calculations)
- graph the data in a scatter graph (add labels, titles)
- apply error bars with a fixed uncertainty
- add a line of best fit to your graph

After doing this notebook, you should be able to reuse these commands to graph any dataset you do for the rest of the year. We will be using Jupyter periodically throughout the year, so its important to learn how to use it now.

Setup your notebook

- Useful Tip Dont forget to name your notebook (above the file menu)! Save often.

Jupyter Notebooks use python as a programming language. Typically the first step is to load in all the modules and libraries of functions needed for your project. So our first step is to load the notebook with plotting and math functions. This is done by running the cell below (click the play button):

```
import pandas as pd
import numpy as np
from math import cos,sin,pi
import matplotlib as mpl
import matplotlib.pyplot as plt
```

Try this:

Now just test that you imported these libraries by running (by clicking play or shift-enter) the following:

```
pi
```

```
3.141592653589793
```

```
11.1+13.22+15.667
```

```
39.987
```

You can assign variable names to values and then use them.

```
a=2
b=3
a*b
```

```
6
```

Exercise 1:

In the code cell below solve for the year you born by subtracting your age from the current year.

```
# Type in your work here below here:
2021 - 2003
```

```
18
```

Exponents

to apply the square root, you can either use the square root function or just raise the power using the **, for example:

Double-click (or enter) to edit

```
3**2
```

```
9
```

```
3**3
```

```
27
```

```
16**0.5
```

```
4.0
```

Double-click (or enter) to edit

The square root function is in the numpy library and is called like:

```
np.sqrt(16)
```

```
4.0
```

a function is called by a case sensitive name, followed by paranthesis that require inputs. So in the square root case, the name is: np.sqrt and the input is (16). So it runs the square root algorithm for the input of 16.

Exercise 2

Evaluate 22 raised to the power 3.

```
#Type here

22**3
```

```
10648
```

Importing or loading a data set.

We will be working with data collected from a pendulum experiment. In this experiment, we will be working with a simple pendulum and changing the length of the pendulum and measuring the time for one back and forth motion (called the period of the pendulum. This is the number of seconds for

the pendulum to repeat its position. You can conduct trials of this experiment on the online simulation located at: https://phet.colorado.edu/en/simulation/pendulum-lab

I did this and collected a record of all the trials on an Excel spreadsheet. You can then save your spreadsheet as a csv file. CSV stands for comma seperated values and are common/standard in loading into data processing applications. If you want to see what my file looks like just open this link: https://raw.githubusercontent.com/fearlab/Datasets/master/PendulumData.csv you should see lots of commas and numbers. After saving the document, it is nice to store the file someplace remotely. There is a free web repository site called github, at github.com where you'll need to make an account and then upload your data.

Once you have done that, you can always upload your csv files there and access them remotely.

```
# a hashtag tells the program "don't read the rest of the line"
# That way we can write "comments" to humans trying to figure out what the code does

data = pd.read_csv('https://raw.githubusercontent.com/fearlab/Datasets/master/Pendulur


#this command displays the first 5 rows of the data (this is a way to check that the c
data.head(5)
```

	Length	Trial 1	Trial 2	Average T
0	5	12	11.5	11.75
1	6	14	13.0	13.50
2	7	18	17.5	17.75
3	8	20	21.0	20.50
4	9	25	24.0	24.50

Exercise 3

Download my pendulum data. (you can do this by going to: https://raw.githubusercontent.com/fearlab/Datasets/master/PendulumData.csv and then right clicking and selecting "save as" Go onto github.com, make an account and upload this file to your new account.
Hints:

1. make a new repository (I called mine "Datasets")
2. click "add file" or "upload"
3. after you select, click on "committ changes"

After your file is uploaded, click on it until you have the raw data page (this will look pretty much exactly like mine where you downloaded it)

```
# Type the command to load the pendulum data from your github.

my_data = pd.read_csv('https://raw.githubusercontent.com/FifiTeklemedhin/Physics-Jupyt
```

```
# Lets see the first 5 rows, just to make sure it worked:
print(my_data.head(5))
```

```
   Length  Trial 1  Trial 2  Average T
0       5       12     11.5      11.75
1       6       14     13.0      13.50
2       7       18     17.5      17.75
3       8       20     21.0      20.50
4       9       25     24.0      24.50
```

Plotting Data

The function for plotting is plt.plot(x,y) where x is your x values and y is your y values.

In order to plot this, we need to call our x values from the data array we just imported. We call the Length file by typing: data['Length']

the y values are called by typing in data['Average T']

'Length' and 'Average T' are the column headers for the data array. If you look at the data.head(5) you will see the top column has the headers. These are case sensitive. If you type in the wrong header, it wont work, capitalization and spaces included.

```
#this command makes a line plot of the variable.  plt.plot(x,y)
#so our x data will be Length and our y data will be Average T
x = data['Length']
y = data['Average T']
plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x7f9504779990>]
```

Exercise 4

I want to see the x and y values. Can you figure out how to call those below?


```
# call the x values below:
print(my_data['Length'])
```

```
0     5
1     6
2     7
3     8
4     9
5    10
6    11
7    12
8    13
Name: Length, dtype: int64
```

```
# call the y values here:

print(my_data['Average T'])
```

```
0    11.75
1    13.50
2    17.75
3    20.50
4    24.50
5    27.00
6    29.25
7    31.00
8    32.75
Name: Average T, dtype: float64
```

Changing Marker, color and linestyle

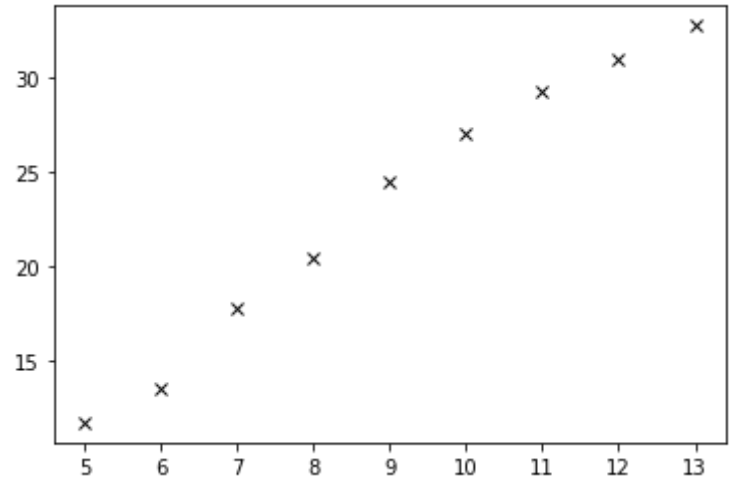
The graph is not really good. First, lets not connect the datapoints with a line. This is done by adding the linestyle = '' command. by leaving it blank, it tells the software to delete the lines connecting the points. If you want dashed lines, you enter in linestyle='--'.

marker='x' changes the data points to x's. Other marker types that are liked by IB are +'s and *'s. You can also do squares, circles, diamonds, and many others ('s','o','d'). So if I wanted to do circles as data points, the syntax would be marker='o'.

color='black' makes all the points that color. There are lots of colors.

```
#I'm going to make the same graph, but add black x's as data points and erase the line
plt.plot(x,y, linestyle='', marker='x', color='black')
```

```
[<matplotlib.lines.Line2D at 0x7f9504247710>]
```

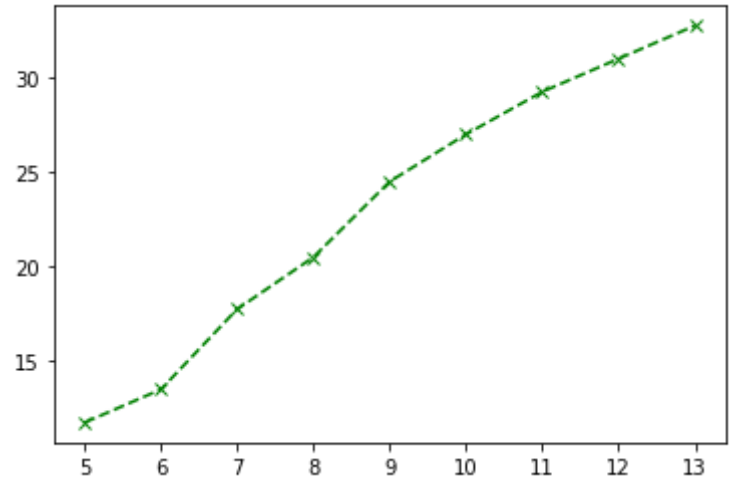


Exercise 5

Make a plot of length vs Trial 1. It will look pretty close to the Average T, fyi. I want: green diamonds connected by dashed lines

```
# Type your line of code to plot here:
x = my_data['Length']
y = my_data['Average T']
plt.plot(x,y, linestyle='--', marker='x', color='green')
```

```
[<matplotlib.lines.Line2D at 0x7ff95a737610>]
```



Now you are done, just go to File -> Print -> Save as PDF and upload to canvas like any normal pdf.

thanks

✓ 0s completed at 10:29 AM