CHEME 5500 "Software Carpentry"

Lab 5 - Graphing, and File Input/Output

Introduction

Python is a wonderful language that includes graphing capabilities, to make life easier on every scientist/engineer. Today's lab will involve taking what we learned last class to read in data from a file, and using python's built in graphing capabilities to plot it.

Matplotlib

To graph in python, we use the matplotlib.pyplot library. We can import it by doing the following:

```
import matplotlib.pyplot as plt
```

Note, I've aliased the imported library to the name *plt*. That means that, whenever I want to call code from matplotlib.pyplot, I can simply call plt.x (where x is some code I want to use). So, let's begin plotting! First, we need some data:

```
pokemon_str = ["Pikachu", "Vaporeon", "Mew", "Mewtwo"]
pokemon_index = [1, 2, 3, 4]
pokemon_coolness = [1, 3, 84, 82]
```

Now, we can simply plot everything:

```
plt.plot(pokemon_index, pokemon_coolness)
```

In this example, I'm planning on plotting a "Pokemon Coolness Factor" (let's call it Γ). I would like the x-axis to hold the names of pokemon, but unfortunately matplotlib only wants floats for the x and y axis values. To get around this, we can manually specify them as follows:

```
plt.xticks(pokemon_index, pokemon_str, rotation=30)
```

Now, what fundamental properies should we define for our graph? Well, it needs a title, and some labels for the axes. We can do that as follows:

```
plt.title("Pokemon Coolness")
plt.xlabel("Pokemon")
plt.ylabel("Coolness Factor")
```

Great. Now, we have the our graph! We can add a buffer around the data quickly by specifying our bounds:

```
xmin, xmax = 1, 4
ymin, ymax = 0, 90
plt.axis([xmin, xmax, ymin, ymax])
```

Before we display the graph, we need to take note of a quirk in how matplotlib behaves. That is, to display a graph we simply need to call the function *plt.show()*; however, if we do so, the entire contents of the graph are now deleted. We'll see our graph, but if we tried saving it afterwards as an image, we would not be able to do so. Thus, it's wise to save the figure as a variable first before showing it:

```
# This command get the figure from plt and returns it to the variable fig
fig = plt.gcf()
# Now we can display our graph
plt.show()
# And we can save our figure to a file
fig.savefig("pokemon.png")
```

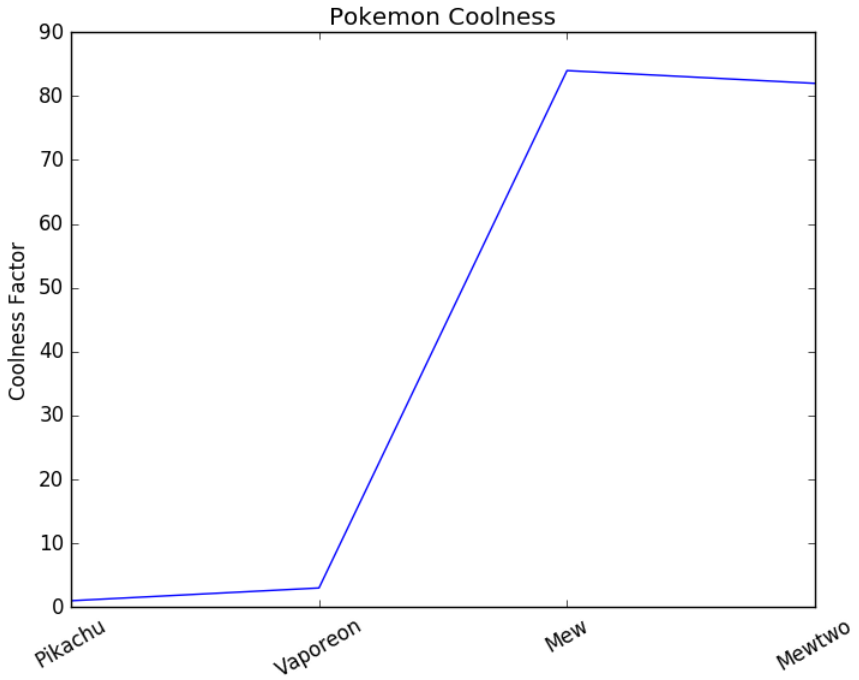

Figure 1: Pokemon coolness factor plotted for a select few pokemon.

Today's Lab Task

You will be given a file called "sin.data". In it is two columns of data, the first of which holds an angle θ, and the second of which is $sin(\theta)$. Your goals are as follows:

1. Read in the file.
2. Parse the data into two lists.
3. Using the above information, make a graph and save an image of it.

Challenge 1: If you finish early, you should also do the following:

1. Using python, generate two lists corresponding to the x and y values of a cosine wave.
2. Save the cosine wave data to a file in the same format as the sine wave data.
3. Using the code you have already written for the lab, plot the cosine wave.

Challenge 2: Try figuring out how to plot both the sine and cosine on the same plot.