DS2000
Fall 2021
Handout: Data Visualization with Matplotlib

Data Visualization is an important tool for any data scientist. We use graphs, charts, and maps to communicate what we've learned about a dataset to peers, colleagues, and strangers. It's often more impactful to share an image than to share numbers or descriptions.

We'll use Matplotlib for most of our data viz in this class. It's a huge, complex Python library and has the capability to do all kinds of visualizations. We'll learn more about it as we go through the semester; this week we're just dipping our toes in the water. We've set up this handout to capture the most important pieces you need to begin creating some plots with matplotlib.

Starting with Matplotlib

First, you need to import the matplotlib module, which is called `matplotlib.pyplot`. And because "matplotlib.pyplot" is a mouthful, use this import statement at the very top of your .py file:

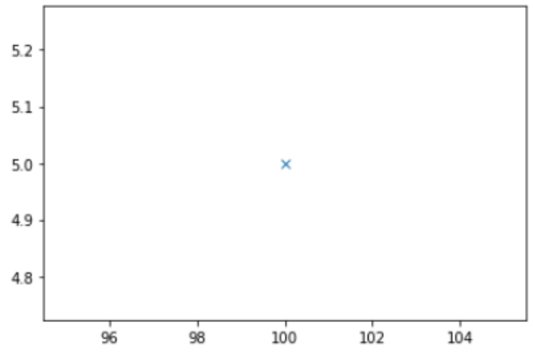
```
import matplotlib.pyplot as plt
```

This is a shortcut. Now, when you need to refer to matplotlib.pyplot, you just need to say *plt*.

Our simplest visualization will be a point on a x, y plane (x-coordinate comes first). We'll use *plt.plot()* to make our point show up, and we'll mark it with an X:

```
# Draws an x at position (100, 5)
plt.plot(100, 5, "x")
```

In Spyder, click on the "plots" tab to see what the plot looks like, which should be like this:



Plotting a single point with an X.

I can plot more than one point by simply calling *plt.plot()* once for every point I want to show. The values for the points can come from anywhere. We can get data by prompting the user, or by reading from a file.

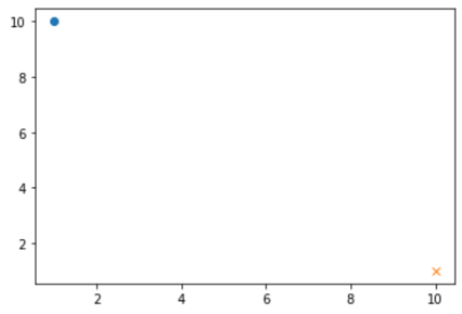
Here's how I would prompt the user for two points and plot them, wherever they are:

```
x1 = int(input("Enter x1\n"))
```

```
y1 = int(input("Enter y1\n"))
x2 = int(input("Enter x2\n"))
y2 = int(input("Enter y2\n"))

plt.plot(x1, y1, "o")
plt.plot(x2, y2, "x")
```

Depending on what the user typed in, I'd get a plot that looks something like this:



Plotting two points from the user.

Customizing Your Plots

plt.plot() is really all we need for now (again, matplotlib has WAY more functionality than this, and we'll work our way up to it!). Even with this one function, we can...

- Plot multiple points (as we were doing above). Later this week we'll cover while loops, which will enable us to plot a ton of points without repeating a ton of code.

- Customize the color of each point
  Here's a red circle: `plt.plot(x, y, "o", color = "red")`
  All the possible colors are here: https://matplotlib.org/3.1.0/gallery/color/named_colors.html

- Customize the marker of a point (it doesn't need to be x's and o's)
  Here's a blue square: `plt.plot(x, y, "s", color = "blue")`
  All the possible markers are here: https://matplotlib.org/api/markers_api.html

- Customize the size of a point
  Here's a BIG blue square: `plt.plot(x, y, "s", color = "blue", markersize = "25")`

- Add a legend to the plot
  Here's two points with different labels, and then display them using *legend*
  ```
  plt.plot(x1, y1, "s", color = "blue", label = "point one")
  plt.plot(x2, y2, "d", color = "orange", label = "point two")
  plt.legend()
  ```

- Add labels and a title to the plot
  Once you’ve plotted all your points, you can label the x- and y-axes, as well as an overall title
  ```
  plt.xlabel("Label along the x axis")
  plt.ylabel("Label along the y axis")
  plt.title("Plot Title")
  ```