Homework: Data Visualization Workbook

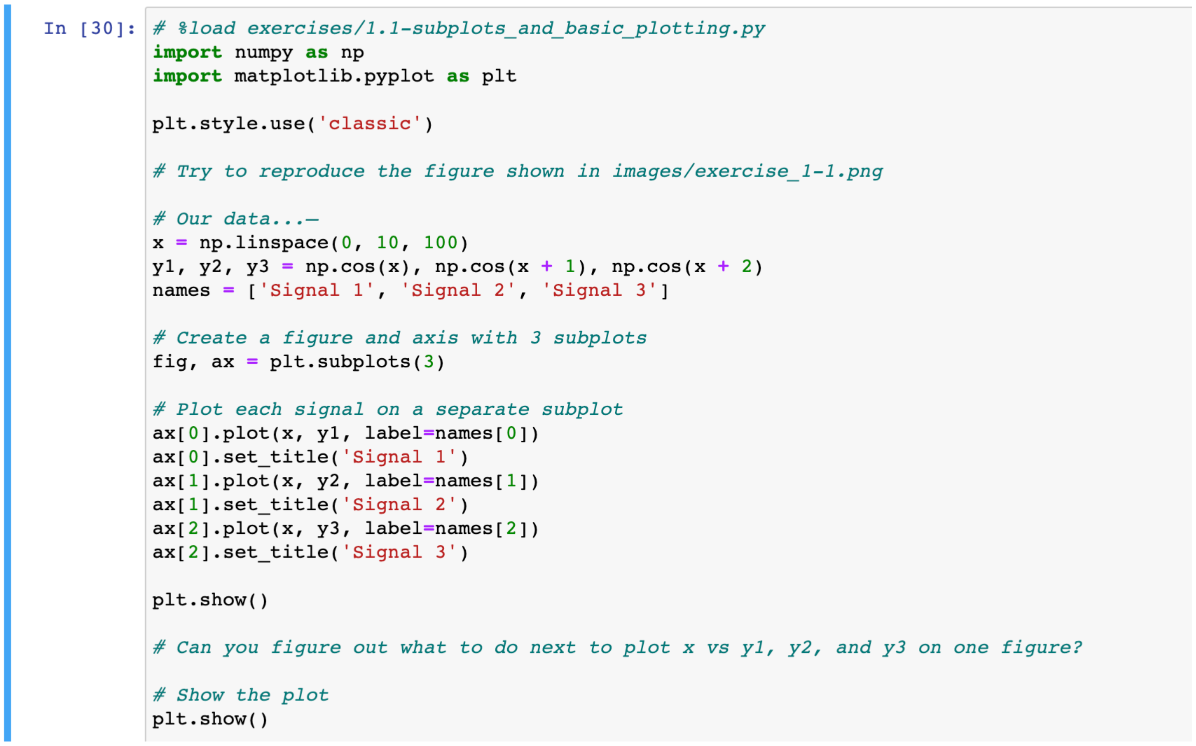
Part 1

In this notebook, we have the concepts of "figures," "subplots," and "layouts" which are explained and demonstrated through examples. These terms refer to different components of a Matplotlib plot.

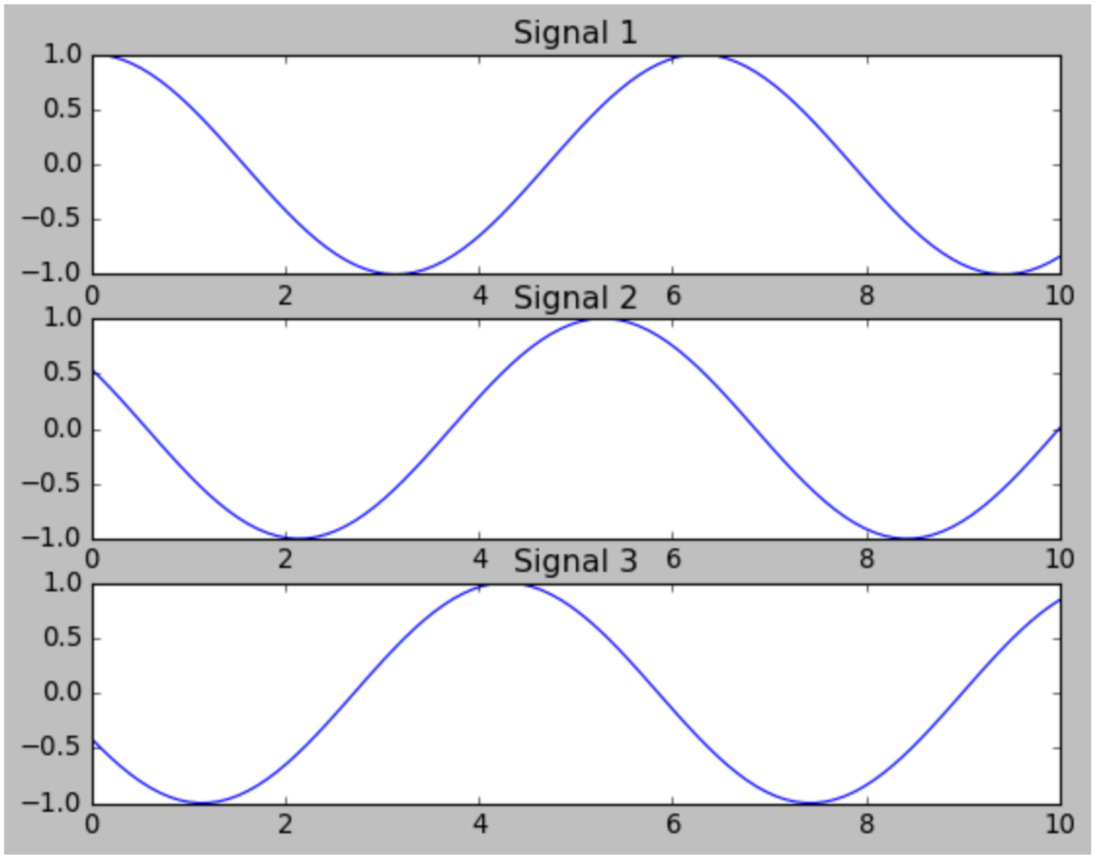
I had some knowledge of matplotlib and had used it occasionally in the past, so I was aware of the basic concept behind graphical. While I had some experience with R I never tried to use command to display plots but in matplotlib we must use plt.show() which I think I will probably forget to use to display graphs, but I did do some practice on that. While I was looking into this notebook I wasn't able to figure out what difference between matplotlib and ggplot and I found thishttps://www.statology.org/ggplot2-vs-matplotlib/ - :~:text=The ggplot2 library is used,of code compared to Matplotlib to be more helpful.

I never used to do subplots in R, but I do know that we use for ex: par(mfrow=c(1,2)) which set the plotting area into a 1*2 array but in matplotlib we use subplots method. While doing the exercise I was able to do the plots but was unable to fiqure out how to do labels so I went ahead and googled it and https://matplotlib.org/stable/gallery/text_labels_and_annotations/label_subplots.html this link helped me to understand ax.set_title().

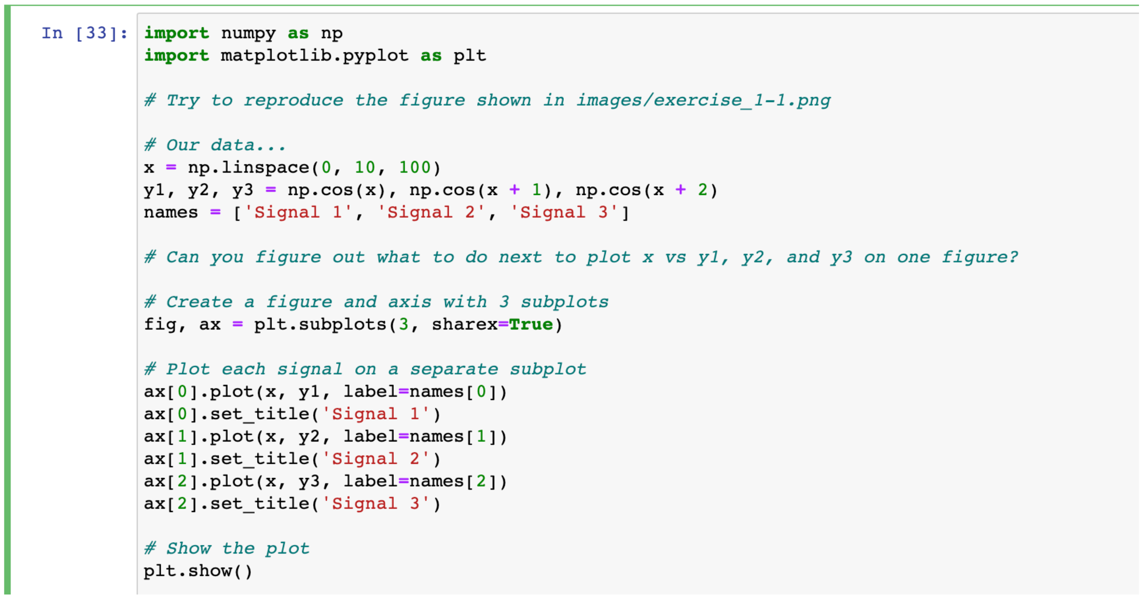
Here is the screenshot of the exercise:



```
In [30]: # %load exercises/1.1-subplots_and_basic_plotting.py
         import numpy as np
         import matplotlib.pyplot as plt

         plt.style.use('classic')

         # Try to reproduce the figure shown in images/exercise_1-1.png

         # Our data...—
         x = np.linspace(0, 10, 100)
         y1, y2, y3 = np.cos(x), np.cos(x + 1), np.cos(x + 2)
         names = ['Signal 1', 'Signal 2', 'Signal 3']

         # Create a figure and axis with 3 subplots
         fig, ax = plt.subplots(3)

         # Plot each signal on a separate subplot
         ax[0].plot(x, y1, label=names[0])
         ax[0].set_title('Signal 1')
         ax[1].plot(x, y2, label=names[1])
         ax[1].set_title('Signal 2')
         ax[2].plot(x, y3, label=names[2])
         ax[2].set_title('Signal 3')

         plt.show()

         # Can you figure out what to do next to plot x vs y1, y2, and y3 on one figure?

         # Show the plot
         plt.show()
```



For the 2nd exercise https://matplotlib.org/3.1.0/api/_as_gen/matplotlib.pyplot.subplots.html this link helped me to learn about sharex = "True" where x- or y-axis will be shared among all subplots. The below is the solution.



```
In [33]: import numpy as np
         import matplotlib.pyplot as plt

         # Try to reproduce the figure shown in images/exercise_1-1.png

         # Our data...
         x = np.linspace(0, 10, 100)
         y1, y2, y3 = np.cos(x), np.cos(x + 1), np.cos(x + 2)
         names = ['Signal 1', 'Signal 2', 'Signal 3']

         # Can you figure out what to do next to plot x vs y1, y2, and y3 on one figure?

         # Create a figure and axis with 3 subplots
         fig, ax = plt.subplots(3, sharex=True)

         # Plot each signal on a separate subplot
         ax[0].plot(x, y1, label=names[0])
         ax[0].set_title('Signal 1')
         ax[1].plot(x, y2, label=names[1])
         ax[1].set_title('Signal 2')
         ax[2].plot(x, y3, label=names[2])
         ax[2].set_title('Signal 3')

         # Show the plot
         plt.show()
```

Part2

In this part there is some exciting ways to plot graphs and I never came across something like this. It was helpful and below is the practice for exercises. These exercises helped lot try different things.

exercise 2.1

```
# Now you're on your own!

fig, ax = plt.subplots()

ax.fill_between(x = x_pred, y1 = y_max_pred, y2 = y_min_pred, color =
fillcolor)

ax.bar(x = x_pos + (bar_width/2), height = y_avg, yerr = y_err, color =
barcolor, width = bar_width, edgecolor='black')

ax.plot(x_raw,y_raw, color = linecolor)

ax.set_xlabel("Minutes since class began")
ax.set_ylabel("Snarkiness(Snark units)")
ax.set_title("Future Projection of Attitudes")

plt.show()
```

exercise 2.2

```
# Now you're on your own!

im1 = axes[0].imshow(data1, vmin=0, vmax=3,interpolation='nearest')
im2 = axes[1].imshow(data2, vmin=0, vmax=3,interpolation='nearest')
im3 = axes[2].imshow(data3, vmin=0, vmax=3,interpolation='nearest')


# Add a colorbar for each plot using the colorbar method
fig.colorbar(im1, cax=cax, orientation='horizontal')
fig.colorbar(im2, cax=cax, orientation='horizontal')
fig.colorbar(im3, cax=cax, orientation='horizontal')


plt.show()
```

The real challenging was figuring out interpolation = ‘nearest’ and also I found https://stackoverflow.com/questions/12473511/what-does-matplotlib-imshowinterpolation-nearest-do this link helpful to understand more about it.

Part 3:

Exercise 3.1 :

```
# %load exercises/3.1-colors.py

t = np.arange(0.0, 5.0, 0.2)
plt.plot(t, t, "#451EE5" , t, t**2,"#9B3822", t, t**3,"#EE6F12" )
plt.show()
```

for the 1st exercise 3.1 I used https://htmlcolorcodes.com/ this link to generate hex values for different colors and tried plotting them.

For next exercise 3.2 :

```
# %load exercises/3.2-markers.py

t = np.arange(0.0, 5.0, 0.2)
plt.plot(t, t,"." , t, t**2,"+" , t, t**3,"D" )
plt.show()
```

As a reference for my future work with Matplotlib, I will undoubtedly save this notebook. I already have markdown page for ggplot, and I find this very helpful. One thing I find intriguing is that while matplotlib allows you to specify what the appearance of geometries should be using actual ASCII characters like "+" .

For exercise 3.3 :

```
t = np.arange(0.0, 5.0, 0.1)
a = np.exp(-t) * np.cos(2*np.pi*t)
plt.plot(t, a, 'r:', marker='D', markeredgecolor='green',
markeredgewidth=2, markerfacecolor='yellow')
plt.show()
```

The challenges in this notebook where understanding the different components and terminology used in Matplotlib, such as "artists," "handles," and "patches," and learning how to manipulate them effectively to produce desired visualizations. This part is very helpful for creating a good visualization.

Part 4:

The challenges in this section include understanding the syntax for setting limits and adding legends, as well as experimenting with different layout options to find the best visual representation for a given dataset.

Exercise 4.1:

```
# Try to plot the two circles, scale the axes as shown and add a
legend

# Hint: it's easiest to combine `ax.axis(...)` and
`ax.margins(...)` to scale the axes

fig, ax = plt.subplots()
ax.plot(x1, y1, color=colors[0], label='Inner')
ax.plot(x2, y2, color=colors[1], label='Outer')
ax.legend()
ax.axis('equal')
ax.margins(0.1)
plt.show()
```

for the exercise 4.2 I did face a bit difficulty in defining the function, but I looked into the solution and tried to do a for loop but didn't get there.

Overall the notebooks are very handy for further references. And I almost never looked into solutions except the last one.