5.1.4 Annotate Charts

V. Isualize is famous for asking tough questions during presentations. During your weekly meeting with Omar, he gives you a tip that will make preparing easier and answer some of V. Isualize's questions before she has a chance to ask them! The tip? Annotate all your graphs.

Adding labels to the axes, a title, and a legend helps the viewer understand what they're looking at and helps tell your data story in a clean, clear manner. After all, even the most visually appealing graph isn't much use if we don't know what it's about!

With Matplotlib, we can add labels to the axes, a title, and a legend. We can also change the thickness of the graphed line and add markers for the data points.

Annotate Charts Using the MATLAB Method

Here are a few methods that you can use to annotate charts using the MATLAB method:

Matplotlib Functions	Feature
`plt.figure(figsize=(w, h))`	Change the size of the figure in pixels. Added on the first line of the script.
`plt.plot(x, y, label='line')`	Add a label that will be added to the legend.
`plt.xlim(min, max)`	Set the min and max range of the x-axis.
`plt.ylim(min, max)`	Set the min and max range of the y-axis.
`plt.xlabel('x label')`	Add a label to the x-axis.
`plt.ylabel('y label')`	Add a label to the y-axis.
`plt.title("Title")`	Add a title.
`plt.legend()`	Add a legend.
`plt.grid()`	Add a grid to the chart.
`plt.savefig("add a path and figure extension")`	Save the figure with the given extension. Added at the end of the script.

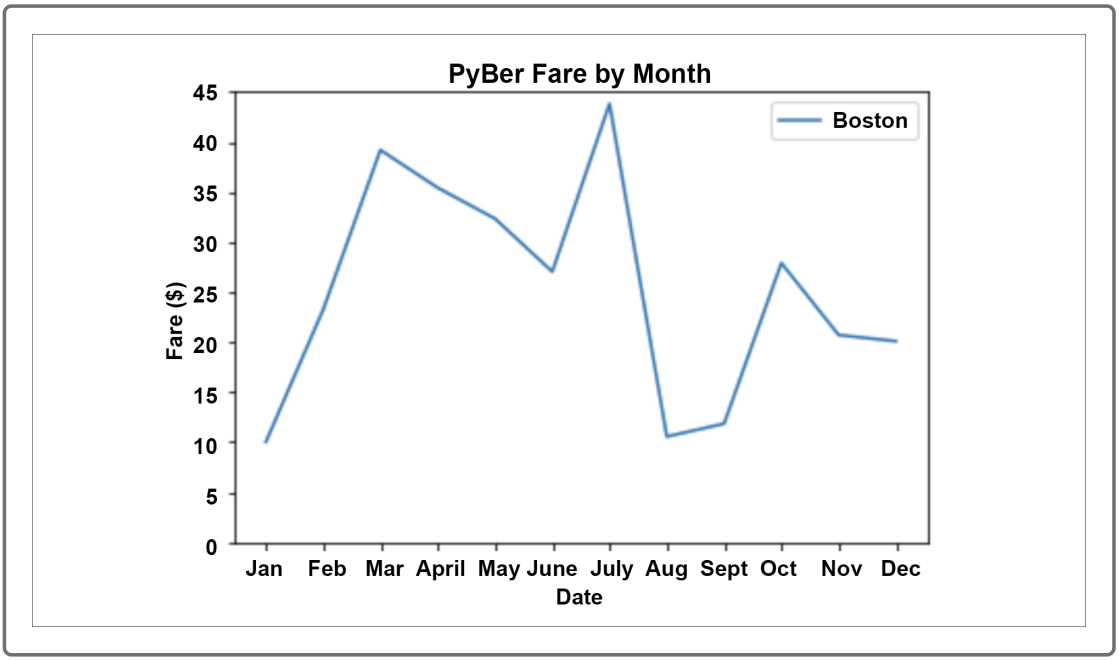
It is highly recommended to add x-axis and y-axis labels and a title to every graph you create so the viewer knows what it conveys. If you have more than one line or bar on a graph, making each line or bar stand out with a distinct color or line style is helpful, as is adding a legend for each dataset that the lines or bars represent. In addition, thoughtfully setting your x-axis and y-axis ranges can make the data more appealing.

Let's annotate our ride-sharing line chart with some of these features to make it more informative and visually appealing.

Add the following code to a new cell and run the cell:

```
# Create the plot and add a label for the legend.
plt.plot(x_axis, y_axis, label='Boston')
# Create labels for the x and y axes.
plt.xlabel("Date")
plt.ylabel("Fare($)")
# Set the y limit between 0 and 45.
plt.ylim(0, 45)
# Create a title.
plt.title("PyBer Fare by Month")
# Add the legend.
plt.legend()
```

After running the cell, our graph has more information, as you can see here:



We can also change the width and color of the line and add markers for the data points. This is done by declaring parameters in the `plt.plot()` function.

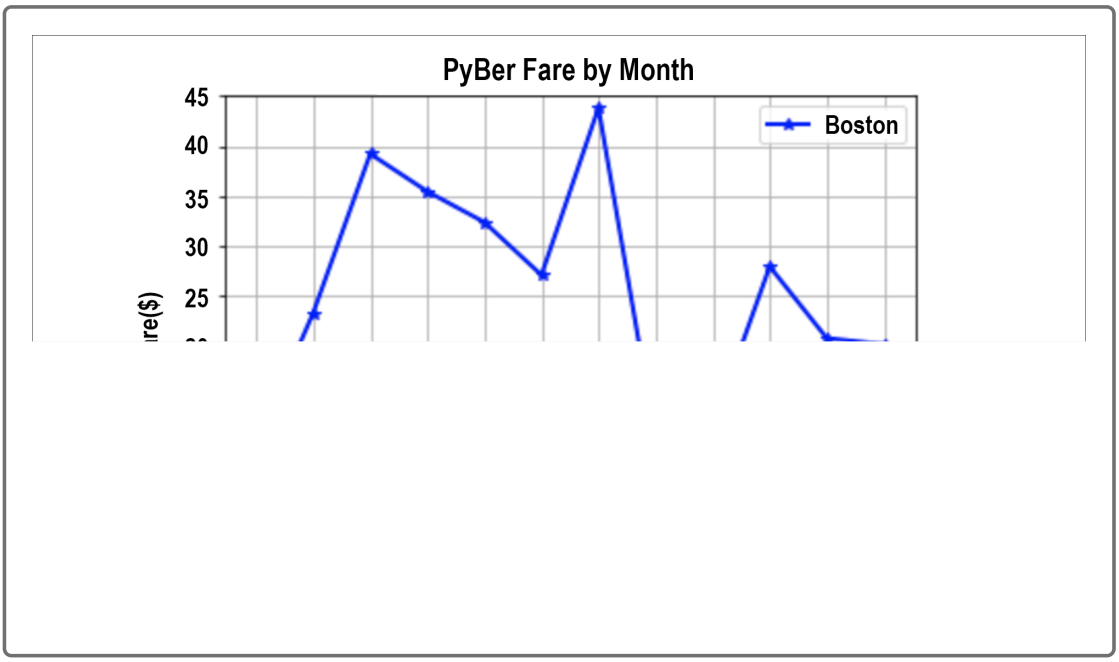
To declare the parameters for line color, width, and marker, we use `color=`, `width=`, and `marker=`, respectively, inside the `plt.plot()` function.

Let's go ahead and amend the code block to add a new line color, modified line width, and a marker for our data points.

Add the following code to a new cell:

```
# Create the plot.
plt.plot(x_axis, y_axis, marker="*", color="blue", linewidth=2, label='Bosto
# Create labels for the x and y axes.
plt.xlabel("Date")
plt.ylabel("Fare($)")
# Set the y limit between 0 and 45.
plt.ylim(0, 45)
# Create a title.
plt.title("PyBer Fare by Month")
# Add a grid.
plt.grid()
# Add the legend.
plt.legend()
```

After running the cell, our chart looks like this:



Retake

Retake

NOTE

> For more information about how to create graphs using the MATLAB method, see the following documentation:
>
> **Matplotlib documentation on matplotlib.pyplot.plot (https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html)**
>
> **Matplotlib documentation on the Pyplot API (https://matplotlib.org/stable/api/index.html)**
>
> **Matplotlib documentation on the anatomy of a Matplotlib chart (https://matplotlib.org/tutorials/introductory/usage.html#parts-of-a-figure)**

Annotate Charts Using the Object-Oriented Interface

Annotating graphs using the object-oriented interface is similar to using the MATLAB approach, with a slightly different syntax.

Here are a few methods you can use to annotate your chart using the object-oriented interface method:

Matplotlib Object-Oriented Functions	**Feature**

`fig, ax = plt.subplots(figsize=(w, h))`	Change the size of the figure in pixels. Add this in the `subplots()` function.
`ax.plot(x, y, label='line')`	Add a label that will be added to the legend.
`ax.set_ylim(min, max)`	Sets the min and max range of the y-axis.
`ax.set_xlim(min, max)`	Sets the min and max range of the x-axis.
`ax.set_xlabel('x label')`	Add a label to the x-axis.
`ax.set_ylabel('y label')`	Add a label to the y-axis.
`ax.set_title("Title")`	Add a title.
`ax.legend()`	Add a legend.
`ax.grid()`	Add a grid to the chart.
`** plt.savefig("add a path and figure extension")`	Saves the figure with the given extension. Added at the end of your script.

Now test out your skills with the following Skill Drill.

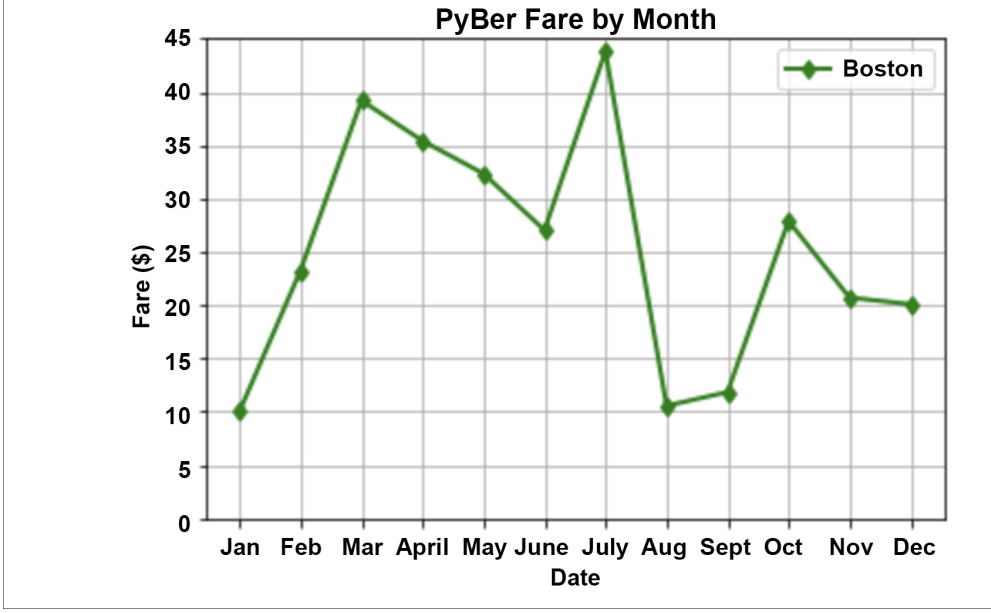
SKILL DRILL

Using the object-oriented approach, make the following changes to your line chart:

1. Change the line color to green.
2. Change the marker to a diamond.
3. Change the line width to 2 points.
4. Add a legend for the city of Boston.
5. Add a title and axes labels.
6. Add a y-axis limit.

7. Add grid lines.

When you're done, your chart should look similar to this:



NOTE

> For more information on how to create charts using the object-oriented interface method, please refer to the following documentation:
>
> **Matplotlib documentation on matplotlib.axes.Axes.plot (https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.plot.html)**
>
> **Matplotlib documentation on the Axes class (https://matplotlib.org/stable/api/axes_api.html)**

Great job on creating your first line chart! Knowing how to create a line chart is an essential skill to have for creating quick visualizations of your data.

Let's build on this and work on making our first bar chart, another essential graph to have in your data analysis tool belt.