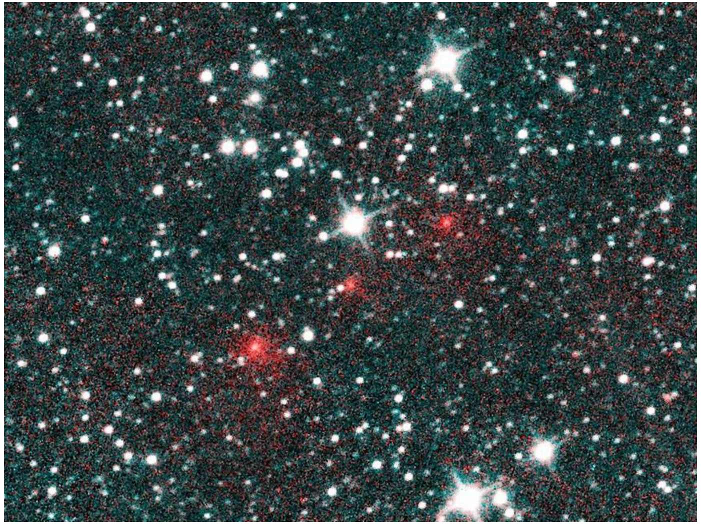
# Types of Visualizations Review

10 THE KNOWLEDGE HOUSE
Celebrating 10 Years of Diversifying Tech

# Agenda - Schedule

1. Warm-Up
2. Visualizing Data
3. Creating Visualizations in Python
4. Break
5. TLAB



[*"The existence of Comet NEOWISE (here depicted as a series of red dots) was discovered by analyzing astronomical survey data..."*]

# Agenda - Goals

- Understand which visualizations to choose for your analysis
- Learn how to make data visualizations in Python
- Understand the fundamentals of matplotlib

# Warm-Up

```
{
 "added_at": "2024-11-19T02:31:08Z",
 "track": {
      "album": {
           "album_type": "album",
           "artists": [
                {
                 "external_urls": { "spotify":
             "https://open.spotify.com/artist/7G1GBhoKtEPnP86X2PvEYO"},
                 "href": "https://api.spotify.com/v1/artists/7G1GBhoKtEPnP86X2PvEYO",
                 "id": "7G1GBhoKtEPnP86X2PvEYO",
                 "name": "Nina Simone",
                 "type": "artist",
                 "uri": "spotify:artist:7G1GBhoKtEPnP86X2PvEYO"
                }
             ],
             "uri": "spotify:album:4bGiPtwVEKcXbXs7oKCMqD"
         }
    }
}
```

Evaluate this JSON object. Assume we've loaded this into a dictionary called **songs**. Which syntax will give us the value of the uri key?

```
{
 "added_at": "2024-11-19T02:31:08Z",
 "track": {
        "album": {
                "album_type": "album",
                "artists": [
                        {
                         "external_urls": { "spotify":
                      "https://open.spotify.com/artist/7G1GBhoKtEPnP86X2PvEYO"},
                         "href": "https://api.spotify.com/v1/artists/7G1GBhoKtEPnP86X2PvEYO",
                         "id": "7G1GBhoKtEPnP86X2PvEYO",
                         "name": "Nina Simone",
                         "type": "artist",
                         "uri": "spotify:artist:7G1GBhoKtEPnP86X2PvEYO"
                        }
                      ],
                      "uri": "spotify:album:4bGiPtwVEKcXbXs7oKCMqD"
                }
        }
}
```

Which syntax will give us the value of the spotify key?

# Review of Variables

# Visualizing Data

Usually for each Wednesday, we will work on more math/theory-heavy concepts as opposed to Python.

However today, we will try to strike a balance and instead go over common data analysis visualizations as well as how to implement them using a package called **matplotlib.**

**First**, let's go over our knowledge of different types of variables.

# Independent vs Dependent Variable

**Independent variables** do not change when other variables change, but they can cause change. **We want to manipulate these** to see what changes they make.

**Dependent variables** are named that because they depend on other variables.

A lot of data science entails figuring out if something is **truly being influenced.**

# Quantitative vs Categorical Variables

Going even further, we can also speak about variables in terms of being **quantitative** or **categorical.**

A **quantitative variable** measures a numerical value that implies implicit value ($1000 < $10,000, 35 > 34). It makes sense to sort these values.

A **categorical variable** represents group membership, and often it does not make sense to sort these in any order (heads or tails on a coin-flip, preference for food).

Dependent variable

| | Day 1 | Day 2 | Day 3 | Day 4 |
|---|---|---|---|---|
| Crow Amount | $1 | $10 | $20 | $5 |
| Food Provided Yesterday | cashews | granola bar | granola bar | cashews |

Independent variable

Now that we're equipped with these terms, which **variable is quantitative** and which **variable is categorical**?

Quantitative dependent variable ($$$)

| | Day 1 | Day 2 | Day 3 | Day 4 |
| --- | --- | --- | --- | --- |
| Crow Amount | $1 | $10 | $20 | $5 |
| Food Provided Yesterday | cashews | granola bar | granola bar | cashews |

Categorical independent variable (cashews or granola)

Now that we're equipped with these terms, which **variable is quantitative** and which **variable is categorical**?

Sometimes,discrete quantitative variables are actually categorical variables.

# Discrete vs Continuous Numbers

Furthermore, when we discuss **quantitative variables** we should also notice as to **what kind of number** this variable is.

**Discrete numbers** are numbers which are measured as whole numbers. (1,2,3,4,5, ...42, …)

**Continuous numbers** are numbers which can exist as any real number (3.14, 1.28, …)

This impacts our **methodology** and how we **showcase the data.**

Quantitative dependent variable ($$$)

| | Day 1 | Day 2 | Day 3 | Day 4 |
|---|---|---|---|---|
| Crow Amount | $1 | $10 | $20 | $5 |
| Food Provided Yesterday | cashews | granola bar | granola bar | cashews |

Categorical independent variable (cashews or granola)

Again, let's continue to define these terms. Is our dependent variable **continuous** or **discrete**?

Discrete quantitative dependent variable ($$$)

| | Day 1 | Day 2 | Day 3 | Day 4 |
|---|---|---|---|---|
| Crow Amount | $1 | $10 | $20 | $5 |
| Food Provided Yesterday | cashews | granola bar | granola bar | cashews |

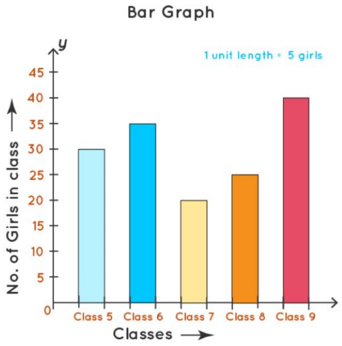
Categorical independent variable (cashews or granola)

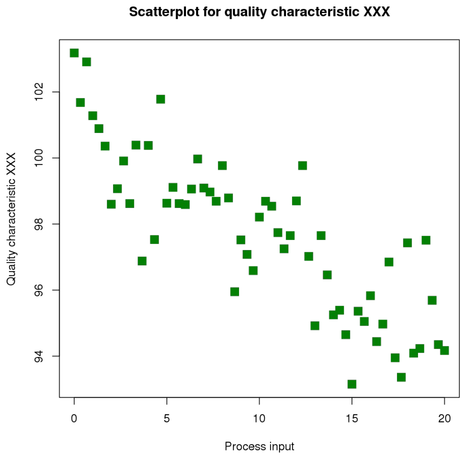
You wouldn’t be at fault for assuming this is discrete ($1, $10, $20, $5). But instead of just using the information you see before you, think to yourself, **can money be continuous?**

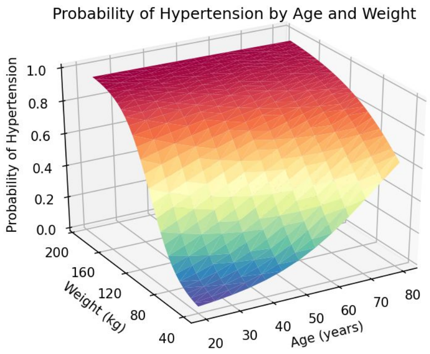
Let’s apply this knowledge to different types of visualizations.

# Visualizing Data

| Univariate | Bivariate | Multivariate |
| --- | --- | --- |




Anything more than 3-dimensions quickly loses interpretability

When it comes to visualizing data we can either do **univariate analysis** (one-dimension), **bivariate analysis** (two-dimensions), or **multivariate analysis** (many dimensions). We usually group bivariate and multivariate together

# Visualizing Data

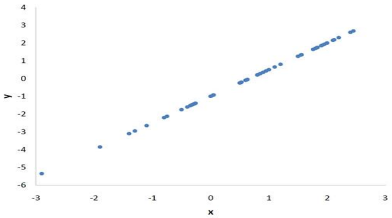


Figure 1. A perfect positive linear relationship, r = 1.

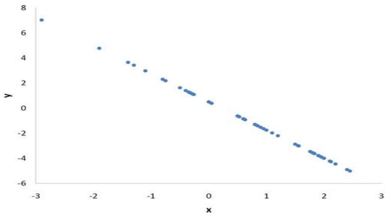


Figure 2. A perfect negative linear relationship, r = -1.

Before we get into the mathematics of **predicting how one dimensions influences another**, let's first dive into the process of **visually discovering** and **presenting** relationships between variables.

Sometimes, half the battle of data exploration is simply choosing the appropriate visualization.

Just a thought: are we limited to only 1 independent variable for every 1 dependent variable? Or could we have multiple independent variables?

# Visualizing Data - Describing vs Prescribing

Before we begin speaking about specific types of graphs, let's go over the difference between two types of data analysis that you might perform in your work and which graphs they entail:

- **Descriptive** data analytics
- **Prescriptive** data analytics

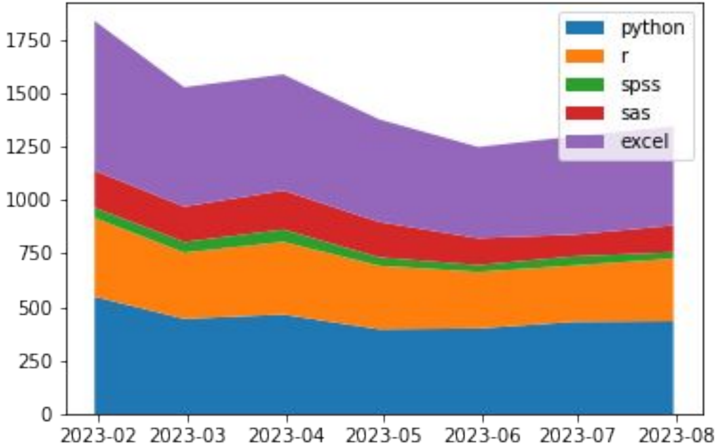
The meaning is self-explanatory. You're either **describing what happened already**, or you're **prescribing next steps** based on your predictions.

# Visualizing Data - Descriptive Analytics

In descriptive analytics, we are:

- ***Exploring what occurred in the past***
- ***Identifying anomalies***
- ***Identifying relationships and patterns***

**Ex**: *Current users using our platform, last-years sales, historical placement rates, current flying conditions (wind, temp, etc)*



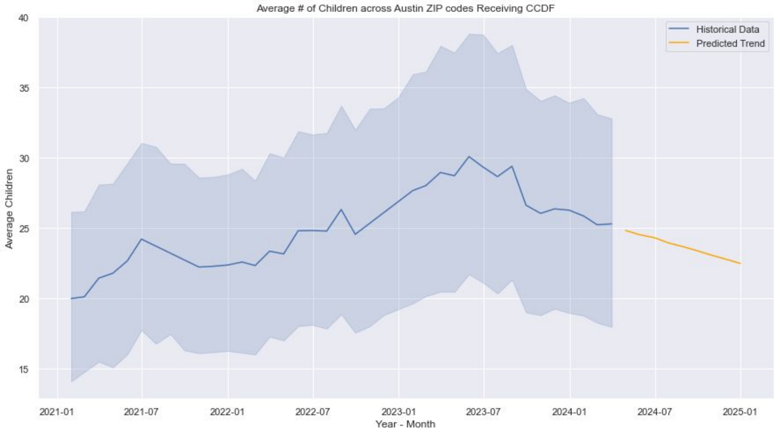
# Visualizing Data - Prescriptive Analytics

Notice that next-actions are built off of descriptive data analytic insights (history predicts future).

In prescriptive analytics, we are:

- ***recommending best course of action***
- ***predicting results***
- ***interpolating results based on history***

**Ex**: *Average number of users next year based on website changes, next year's sales based on incoming administration, next year's placement rates based on program change*

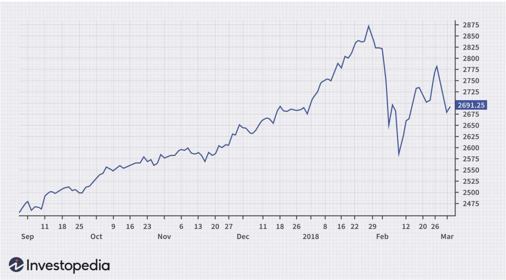


This list is not exhaustive! You may find yourself using a graph outside of this list.

# Visualizing Data

When it comes to presenting our findings for both these types of analytics, we have a few options:

- *bar graph*
- *histogram*
- *scatter plot*
- *line chart*
- *boxplot*
- *pie-chart*




These are **not interchangeable** and must be chosen intentionally. Let's discuss the appropriate use case of each one.

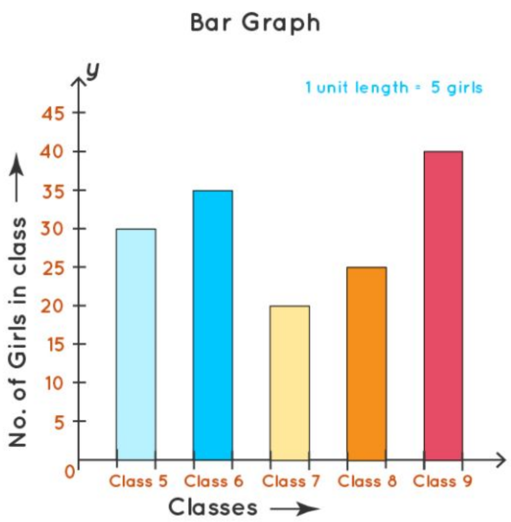
# Visualizing Data - Bar Graph

We use **bar-graphs** to represent differences in **categories in one dimension** and sometimes **time**. This visualization is **univariate.**

That is, our **x-axis is always categorical**.

And our **y-axis is always quantitative**.

This graph is almost always **descriptive**.



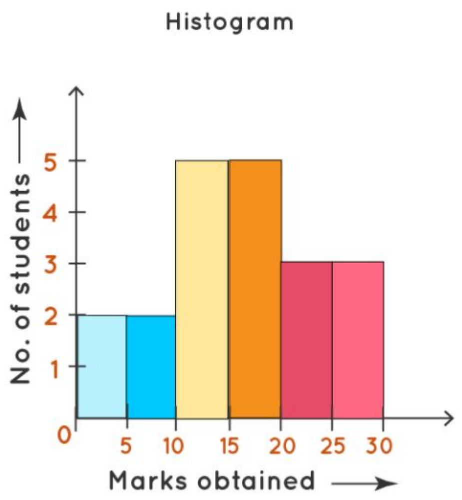
# Visualizing Data - Histogram

We use **histograms** to represent **distributions of one dimension** (aka **the frequency of different values in a dimension)**. This visualization is **univariate.**

That is, our **x-axis is always quantitative**.

And our **y-axis is always quantitative**.

This graph is almost always **descriptive**.



# Visualizing Data - Line Plot

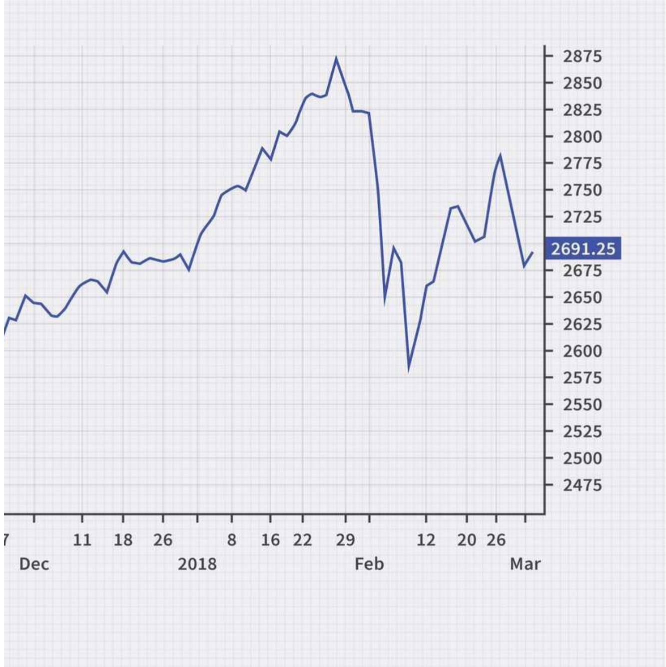
We use **line plots** to represent **changes in quantity of one dimension across time**. This visualization is **univariate.**

That is, our **x-axis is always time**.

And our **y-axis is always quantitative**.

This graph could be **descriptive or prescriptive**.

We can express the results of regression using line plots.



We can express the results of either regression or cluster analysis using scatter plots.

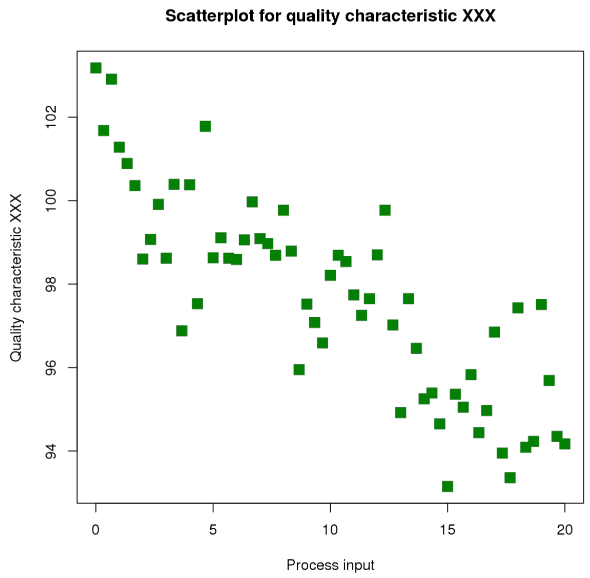
# Visualizing Data - Scatter Plot

We use **scatter plots** to represent **distributions of more than one dimensions.** This visualization is **bivariate/multivariate.**

That is, our **x-axis is always quantitative**.

And our **y-axis is always quantitative**.

This graph could be **descriptive or prescriptive.**



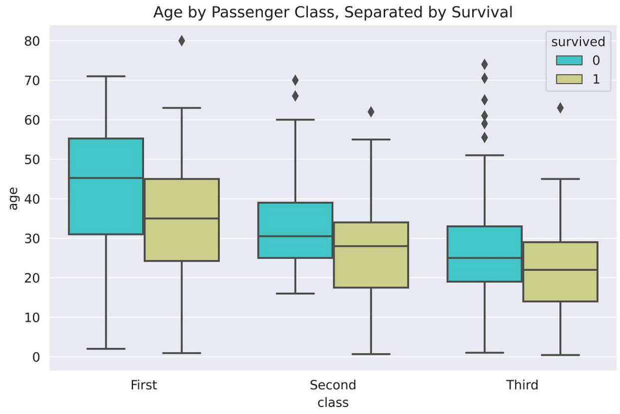
# Visualizing Data - Box Plot

We use **box plots** to represent **distributions of different categories in more than one dimension.** This visualization is **bivariate/multivariate.**

That is, our **x-axis is always categorical**.

And our **y-axis is always quantitative**.

This graph is usually **descriptive**.

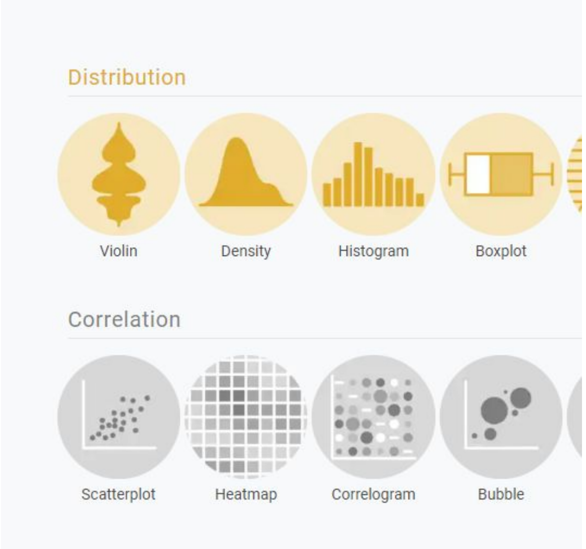


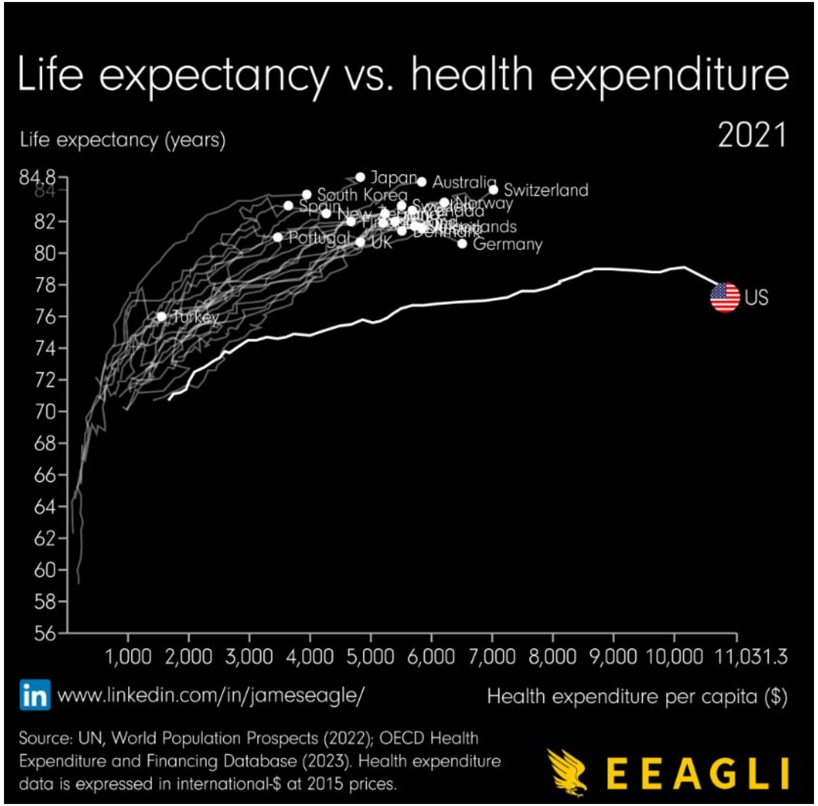
# Visualizing Data - More Visualizations

Note that these visualizations are just the beginning to making **effective and informative graphs.**

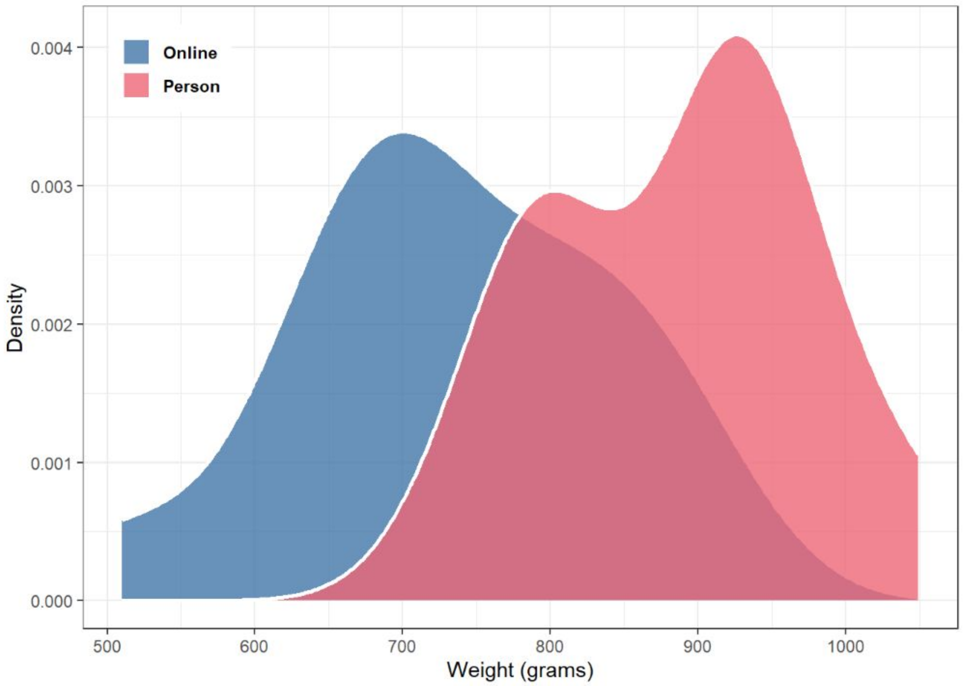
You should also consider other eye-catching visualizations that best express what you are aiming to highlight

https://python-graph-gallery.com/






At some point, creating visualizations is more of an art than a science, so feel free to explore past the visualizations we introduce you to: https://www.reddit.com/media?url=https%3A%2F%2Fi.redd.it%2Fsyq1012cdy0d1.jpeg




And don’t hesitate to have fun!
https://old.reddit.com/r/dataisbeautiful/comments/1buup90/oc_if_you_order_chipotle_online_you_are_probably/

# Creating Visualizations in Python

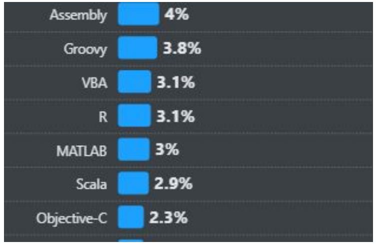
# Using matplotlib

MATLAB

Now that we've reviewed the types of visualizations we could use in our data analysis work, let's go over usage of the **matplotlib** package to generate these visualizations.

**Matplotlib is a Python package** from the creators of **Matlab**, a domain specific language that researchers use to do scientific analysis.

Unless you plan to pursue neuroscience or some other niche analytical field, you do not need to learn matlab, but you should learn about **matplotlib**.




Only 3% of techis using matlab in their work.

# Matplotlib - General Form

There are two ways to use matplotlib:

- **Explicit:** *Manually create the objects you need (good for creating **multiple** plots)*
- **Implicit:** *Use method calls to create objects automatically (good if you're only creating **one** plot)*

For this exercise, we will use the **explicit** method. However we will also show you **equivalent implicit** code. First, let's review the general pattern for matplotlib.

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

ax.plot([1,2,3,4,3,5,2,4])
ax.set_title("Test Plot")
ax.set_xlabel("x-axis")
ax.set_ylabel("y-axis")

fig.savefig("test.png")
```

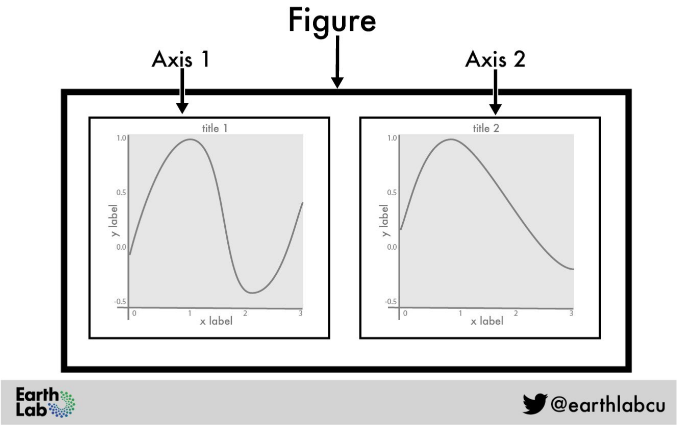
Let's begin by reviewing an idiomatic piece of matplotlib code. All data visualization code that you generate will follow this specific pattern (when implementing **explicitly**)

```python
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

ax.plot([1,2,3,4,3,5,2,4])
ax.set_title("Test Plot")
ax.set_xlabel("x-axis")
ax.set_ylabel("y-axis")

fig.savefig("test.png")
```

Notice that the matplotlib package name is quite lengthy, therefore we alias it typically as “plt” to make calling this package easier

Since matplotlib is not a built in package, you must first download it via pip and then import it to all subsequent code

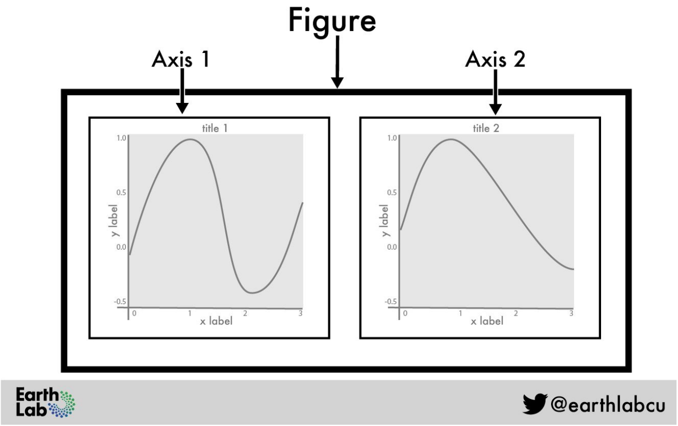
```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

ax.plot([1,2,3,4,3,5,2,4])
ax.set_title("Test Plot")
ax.set_xlabel("x-axis")
ax.set_ylabel("y-axis")

fig.savefig("test.png")
```




Think of the **figure** as the **image of the graph**, while the **axes** is the **graph itself.** For example, you could have multiple axes (graphs) in the same figure (image).

We then create **two** objects from a plt method called "subplots()." This gives us two ways to manipulate our data visualization: through the **figure** and through the **axes**.

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

ax.plot([1,2,3,4,3,5,2,4])
ax.set_title("Test Plot")
ax.set_xlabel("x-axis")
ax.set_ylabel("y-axis")

fig.savefig("test.png")
```




We don't expect you to memorize all these methods. Instead we invite you to use the [Axes] and [Figure docs] to determine what each object can do.

ax.pie() # create pie chart

ax.bar() # create bar chart

ax.scatter() # create scatter plot

ax.set_ylim() # set y-axis range

ax.set_xlim() # set x-axis range

ax.set_xlabel() # set x-label

ax.set_ylabel() # set y-label

fig.savefig() # save image

fig.show() # show image

fig.clear() # clear image

Remember, your axes and figure objects do not have to be called “ax” and “fig.” You could call them whatever you want, as long as they appropriately express what kind of object they are.

This brief “cheat-sheet” should give you a good idea of the functionalities and differences between these two objects. However, look to the docs:
https://matplotlib.org/stable/api/axes_api.html
https://matplotlib.org/stable/api/figure_api.html

```
ax.pie() # create pie chart

ax.bar() # create bar chart

ax.scatter() # create scatter plot

ax.set_ylim() # set y-axis range

ax.set_xlim() # set x-axis range

ax.set_xlabel() # set x-label

ax.set_ylabel() # set y-label
```

```
fig.savefig() # save image

fig.show() # show image

fig.clear() # clear image
```

Notice how the **axes** methods have to do with **manipulation** of the graph itself (*set the x-axis, set the y-axis, etc*)

```
ax.pie() # create pie chart

ax.bar() # create bar chart

ax.scatter() # create scatter plot

ax.set_ylim() # set y-axis range

ax.set_xlim() # set x-axis range

ax.set_xlabel() # set x-label

ax.set_ylabel() # set y-label
```

```
fig.savefig() # save image

fig.show() # show image

fig.clear() # clear image
```

Whereas the **fig** object has methods which control where the image is saved and how it is presented.

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

ax.plot([1,2,3,4,3,5,2,4])
ax.set_title("Test Plot")
ax.set_xlabel("x-axis")
ax.set_ylabel("y-axis")

fig.savefig("test.png")
```

First we create a plot by feeding in a **list of data** *hint hint*
By default this **makes a line plot** *HINT HINT*

We set a title called "Test Plot" by feeding in a string to our args.

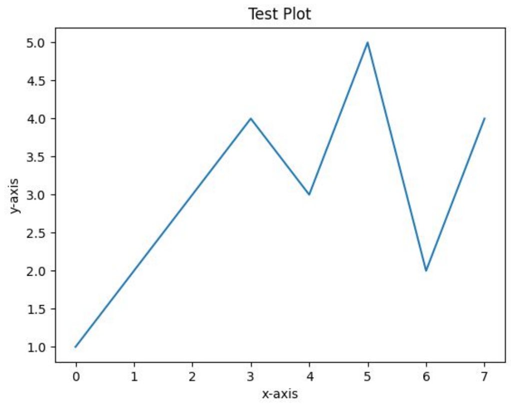
Same for xlabel

Same for ylabel

With this review of methods, let's break down what these **axes** methods do.

```python
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

ax.plot([1,2,3,4,3,5,2,4])
ax.set_title("Test Plot")
ax.set_xlabel("x-axis")
ax.set_ylabel("y-axis")

fig.savefig("test.png")
```

And finally, we save this image inside of our folder and call it "**test.png**"



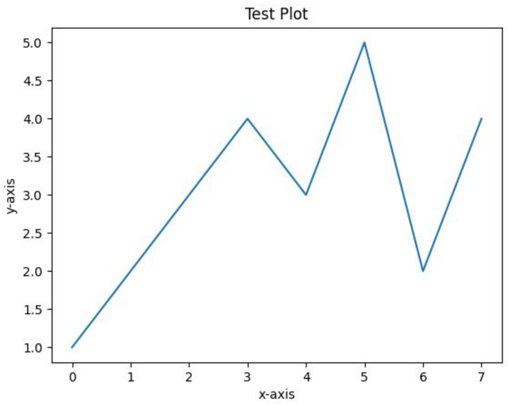

As well as what our **figure** method does. This creates a new image!

```
import matplotlib.pyplot as plt

plt.plot([1,2,3,4,3,5,2,4])
plt.title("Test Plot")
plt.xlabel("x-axis")
plt.ylabel("y-axis")

plt.savefig("test2.png")
```

Now that we've gone over the **explicit** method of creating objects, let's review the **implicit**, which is largely the same but this time we do not need to manually create the objects.

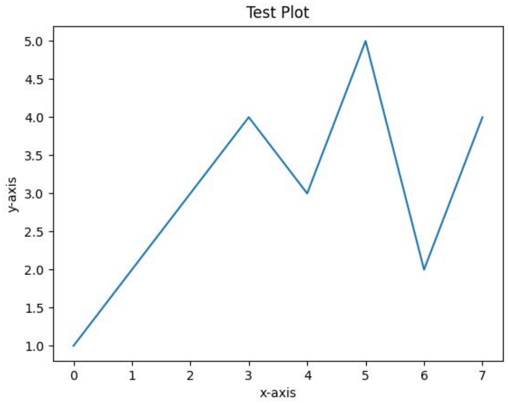
```python
import matplotlib.pyplot as plt

plt.plot([1,2,3,4,3,5,2,4])
plt.title("Test Plot")
plt.xlabel("x-axis")
plt.ylabel("y-axis")

plt.savefig("test2.png")
```

Just like before, this will result in the following graph.



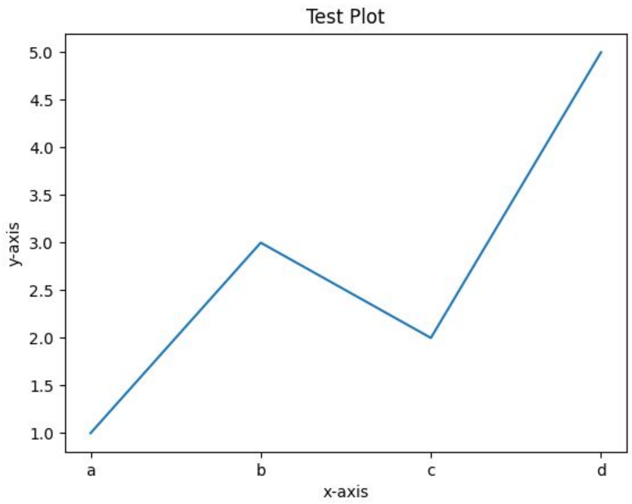

Notice these are *almost* the same as the implicit methods, except this time we only use the **plt** package name to call of our methods.

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

ax.plot([1,2,3,4,3,5,2,4])
ax.set_title("Test Plot")
ax.set_xlabel("x-axis")
ax.set_ylabel("y-axis")

fig.savefig("test.png")
```



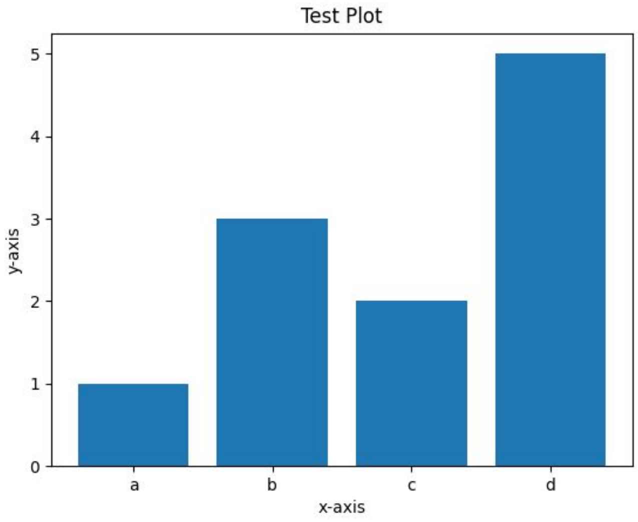

Now that we understand what this code says, let's iterate through all possible visualizations you can make. The default **plot** method gives you a line-plot. You can either include **one list** to **plot the y-axis**...

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

ax.plot(["a", "b", "c", "d"], [1,3,2,5])
ax.set_title("Test Plot")
ax.set_xlabel("x-axis")
ax.set_ylabel("y-axis")

fig.savefig("test.png")
```



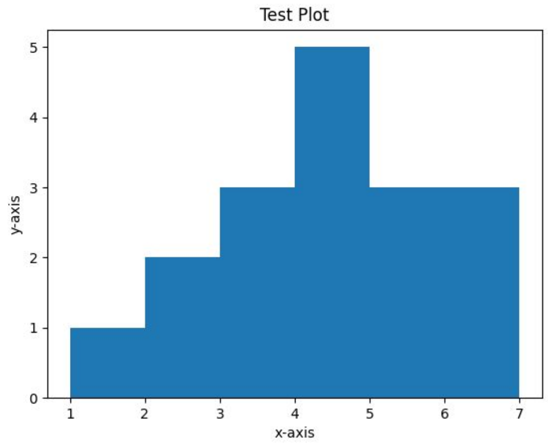

Or with **two lists** to plot the x and y axis. Notice how the first list acts as the x and the second list acts as y.

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

ax.bar(["a", "b", "c", "d"], [1,3,2,5])
ax.set_title(“Test Plot”)
ax.set_xlabel(“x-axis”)
ax.set_ylabel(“y-axis”)

fig.savefig(“test.png”)
```

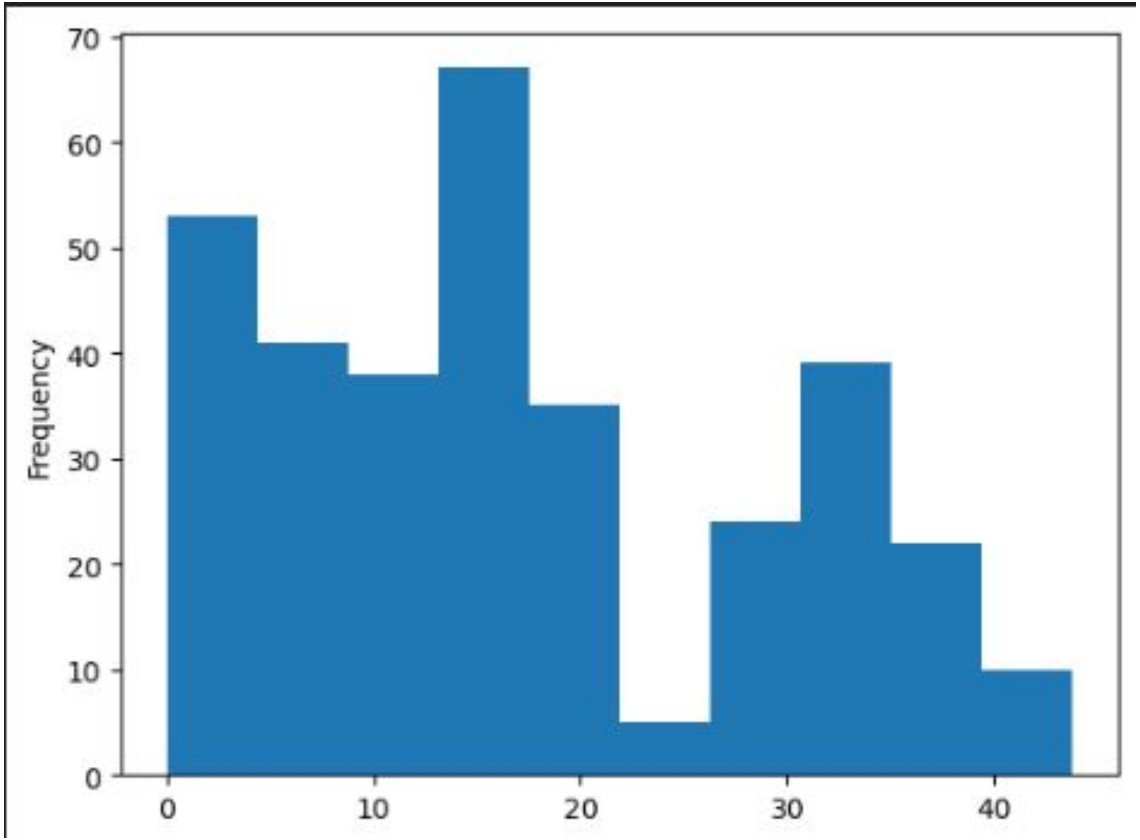



We can use the **bar()** plot to make bar graphs (2 lists, just like with our line-plot)

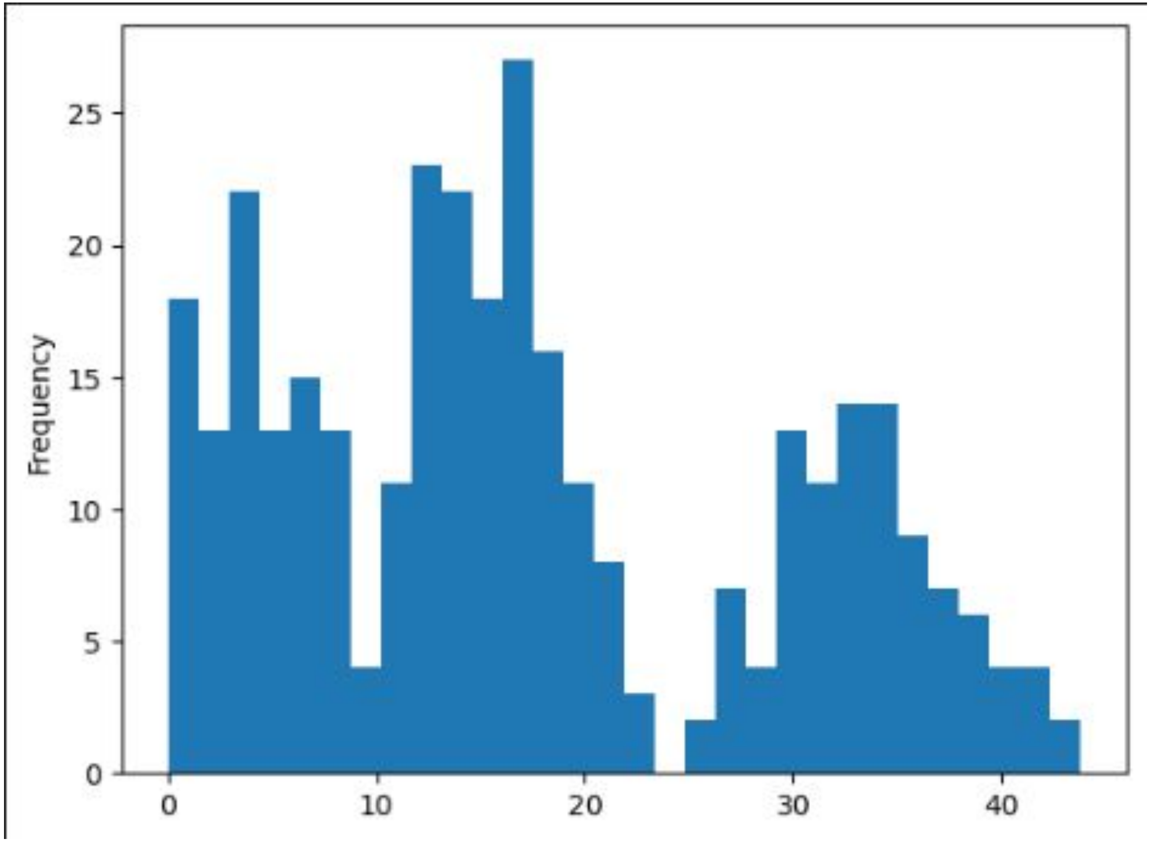
```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

ax.hist([1,2,2,3,3,3,4,4,4,4,4,5,5,5,6,6,7], bins=6)
ax.set_title(“Test Plot”)
ax.set_xlabel(“x-axis”)
ax.set_ylabel(“y-axis”)

fig.savefig(“test.png”)
```



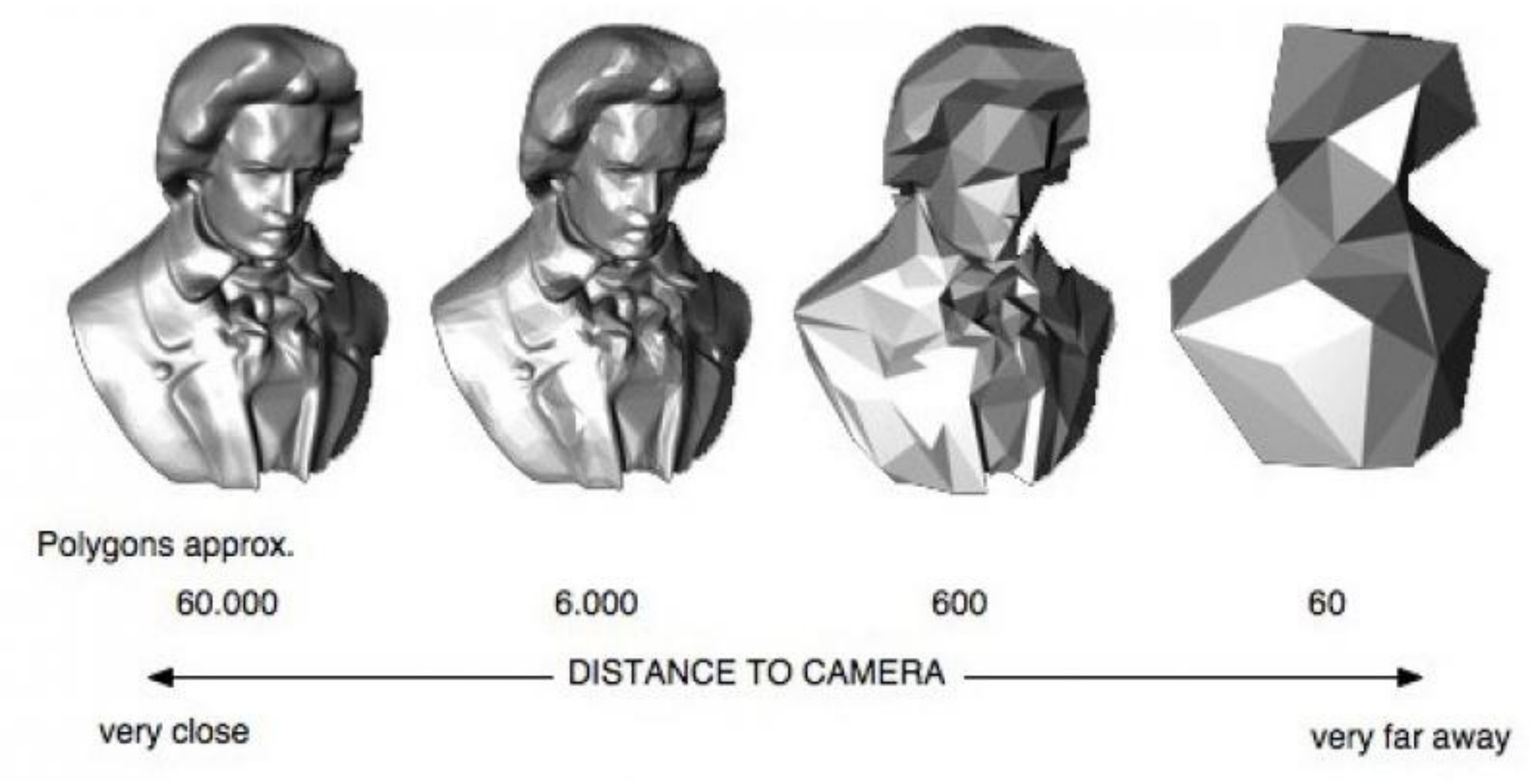

**hist()** to create a histogram. Notice that since this is a univariate visualization, we only need 1 list. The **bins parameter** allows us to control how many number ranges we present in our data.




Often times when you do data analysis, you need to increase the number of bins to capture capture the distribution of your dataset.

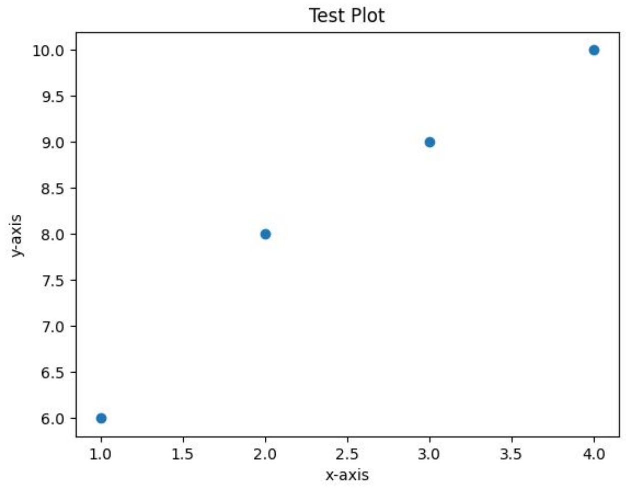



Notice that by increasing the number of bins, the more "detail" we can see. At some point however, we get "diminishing" returns in understandability, **so don't go too far here.**




**More polygons → More resolution**

```python
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

ax.scatter([1,2,3,4],[6,8,9,10])
ax.set_title("Test Plot")
ax.set_xlabel("x-axis")
ax.set_ylabel("y-axis")

fig.savefig("test.png")
```




We'll leave boxplots a mystery for now

**scatter()** to create a scatter plot. Since this is a univariate analysis we should respectively add in two lists of numeric data.

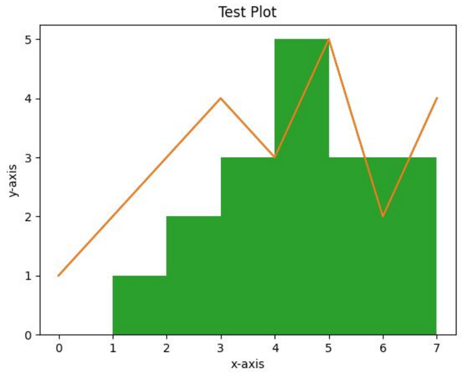
# Common Matplotlib Problem

## Common Matplotlib Problems

Just like all the code we've worked with, you will only get a comprehensive sense of the ins and outs of matplotlib after you've attempted to write your own code.

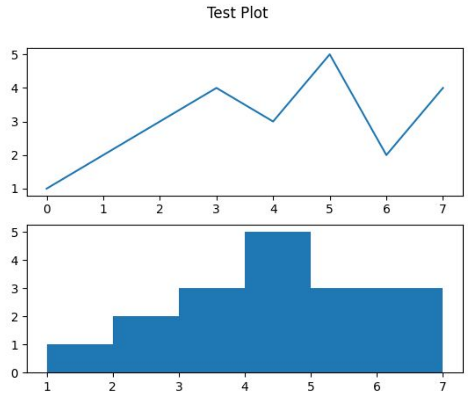
However, there is **one common issue** to look out for when working with this package

```
import matplotlib.pyplot as plt

plt.plot([1,2,3,4,3,5,2,4])
plt.title("Test Plot")
plt.xlabel("x-axis")
plt.ylabel("y-axis")

plt.hist([1,2,2,3,3,3,4,4,4,4,4,5,5,5,6,6,7], bins=6)

plt.savefig("test2.png")
```




**Issue**: I've tried making multiple graphs but everything's been plotted on the same figure!

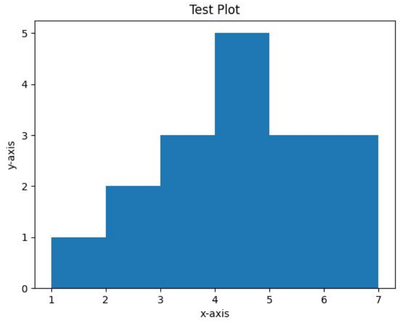
```python
import matplotlib.pyplot as plt

fig, ax = plt.subplots(2)

ax [0].plot([1,2,3,4,3,5,2,4])
ax [1].hist([1,2,2,3,3,3,4,4,4,4,4,5,5,5,6,6,7], bins=6)

plt.savefig("test2.png")
```

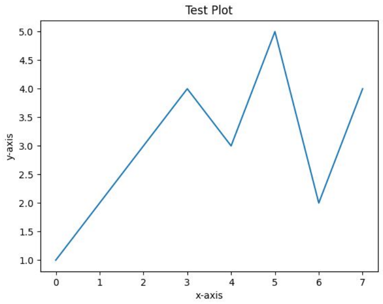



**Fix**: Use the **explicit** method if you're going to be creating multiple separate figures. If you use implicit, everything will be plotted to the same figure.

You can either create 2 subplots…

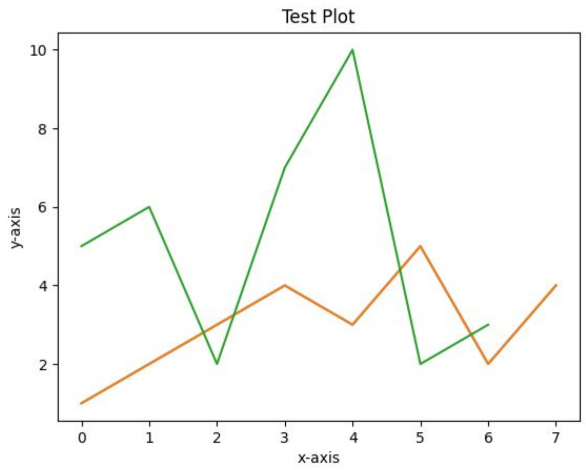
```python
import matplotlib.pyplot as plt

fig1, ax1 = plt.subplots()
fig2, ax2 = plt.subplots()

ax1 .plot([1,2,3,4,3,5,2,4])
ax2 .hist([1,2,2,3,3,3,4,4,4,4,4,5,5,5,6,6,7], bins=6)

fig1.savefig("test2.png")
fig2.savefig("test2.png")
```




...or create two separate fig, ax objects to have 2 separate plots

```python
import matplotlib.pyplot as plt

plt.plot([1,2,3,4,3,5,2,4])
plt.plot([5,6,2,7,10,2,3])
plt.title("Test Plot")
plt.xlabel("x-axis")
plt.ylabel("y-axis")

plt.savefig("test2.png")
```




Opportunity: Plotting multiple plots on the same graph helps to express more information.

# Wrap-Up

# Lab (Due 03/28)

*Taipei City, Taiwan*

The company you work for, Seng-Links, aims to identify periods when a user sleeps or exercises using their varying recorded heart rates.

Your company has provided you a data folder (*data/*) of **4 files** that contain heart-rate samples from a participant. The participants device records heart rate data every 5 minutes (aka *sampling rate*).

You are tasked with writing code that **processes each data file**. You will utilize test-driven development in order to complete this project.

# Stats Quiz (Due 03/28)

Please complete this quiz by 03/28.

This is a 10-question quiz that will test your knowledge of statistics concepts.

**2 attempts allowed.**

p

**3** Multiple Choice 1 point

How much area under the curve of a normal distribution is within 1 standard deviation?

- 50%
- 95.45%
- 68.27%
- 99.73%

**4** Multiple Choice 1 point

If the mean is less than the median, what does that tell us about the distribution?

- The data has a left skew
- The data has a right skew
- The data has no skew

# Glossary

## Thursday

### No review session!

- Please attend career class instead

*Jupyter: scratchpad of the data scientist*

*If you understand what you're doing, you're not learning anything. - Anonymous*

**population -** entire group you could possibly get data from

**sample -** a subset of your population which you collect data from

**feature -** an important component of your data

**independent variable -** something that does not change when other features in your data change

**dependent variable -** a value which changes when other features in your data change

**discrete variable -** a variable which is measured in whole numbers (think integers)

**continuous variable -** a variable which is measured with all real numbers, including decimals

**categorical variable -** something that is describes things (e.g. color, type, class, etc;)

**quantitative variable -** something that measures things (e.g. age, height, weight, etc;)

**latent variable -** a variable that is not directly observable, but can be inferred from other variables that can be directly measured