



Girls Who Code At Home

Depicting Data
Data Visualization in Python

Activity Overview

Data visualization is the process of transforming data that contains large amounts of information both in numbers and words/symbols into graphical art. These pieces of art tell a story about the data. Different representations can highlight/emphasize important information to an audience. For example, when searching for the number of [COVID-19 cases](#) you can see this information in multiple forms: a line graph, [choropleth map](#), table, bar graph, and many more.

In this activity you will learn how to depict data using various types of graphs: line, bar, pie, histogram, and scatter plot using Python. We will specifically be looking at data around [Kickstarter](#) projects, a crowdfunding platform. Data Scientists today take on many roles when working with data. They must first clean the data, this means removing any inaccurate data or looking for duplicate data, then analyze the data using various types of algorithms, and finally they present their data to stakeholders (depending on the data, stakeholders could include people like policy makers, business leaders, researchers, doctors, the general public, etc.).

If you want to learn more about how to manipulate the [Kickstarter data](#), check out our activity:

[Data Playground.](#)

The field of [data science](#) is vast, connecting concepts of artificial intelligence, data mining, big data, and machine learning. According to [Glassdoor](#), data scientists are one of the most sought after jobs with an average base salary of \$110,000!

Learning Goals

By the end of this activity you will be able to...

- ◆ utilize different parameters of the `plot()` method in `pandas` to create custom graphs.
- ◆ determine the best graphical representation based on the type of data.
- ◆ understand how to navigate Kaggle and Jupyter Notebook to explore a dataset.

Materials

- ◆ [Kaggle Kickstarter Data](#)
- ◆ [Depicting Data Sample Project](#)
- ◆ **Optional:** If you completed our [Data Playground](#) activity, check out the [Data Playground Sample Project](#) to preview how you might depict your final results.

Prior Knowledge

Before embarking on this project, we recommend that you:

- ◆ can explain what a [variable](#) is in your own words and describe how they can be used in a program.
- ◆ can explain what a [conditional statement](#) is in your own words and describe how they can be used in a program.
- ◆ can explain what a [method/function](#) is in your own words and describe how they are used in a program.
- ◆ have experience using a text-based language like JavaScript, Python, Swift, etc.

If you want a quick refresher on Python we highly suggest you check out our activity

[Can I Help You?](#)

Women in Tech Spotlight: Fernanda Viégas



Image Source: [Google](#)

How do you transform data/information into art? When you want to present information maybe you have created Venn diagrams, bar graphs, tables, or even used pictures to represent data. In Fernanda Viégas' work she experiments with using Machine Learning to transform data into beautiful visualizations.

Fernanda began her journey as a traditional graphic designer. After being introduced to the MIT Media Lab, she was inspired by the artists she met who combined art with technology. She began learning how to code and eventually founded

[Flowing Media](#), a startup that specializes in using data visualization to express ideas and tell stories. It was through this lens that [Google](#) began to recognize her work and offered her a job to lead the [PAIR](#) (People and Artificial Intelligence Research) Team.

Everyday about 2.5 quintillion bytes of data is generated in the world and only about 30-40% is accessible for research. That is a lot of data! Fernanda uses [machine learning](#) and [artificial intelligence \(AI\)](#) to sort and learn from the data in order to help us understand it better. She then applies an art and design framework to transform the data into forms that make it easier for us to visualize and comprehend. Part of her work at Google is developing the [TensorFlow](#) library in JavaScript to make it easier for others to use machine learning to visualize data on their own without having to write code from scratch.

Watch this [video](#) to learn more about how Fernanda's work impacts the way we see data!

Want to learn more about Fernanda? Check out [Fernanda's website](#) where she highlights some of her awesome work! Feel free to also watch this [video](#) where she talks a little bit about her background and her work at Google.

Reflect

Being a computer scientist is more than just being great at coding. Take some time to reflect on how Fernanda and her work relates to the strengths that great computer scientists focus on building - bravery, resilience, creativity, and purpose.



CREATIVITY

Fernanda combines her two passions of graphic design and computer science to forge her own career in data visualization. What are two or more things that you are interested that might seem disconnected? Can you think of activities that might connect your interests?

Share your responses with a family member or friend. Encourage others to read more about Fernanda to join in the discussion!

Step 1: What is Big Data? (5-10 mins)

About 70% of the data generated by the internet goes unused, but why? In this section we will discuss what **big data** is and why it is valuable to companies in every industry.

If you have already completed our [Data Playground](#) activity, feel free to skip Steps 1-3.

Big Data (2 mins)

Big data is just as it is named, a lot of data! We distinguish big data from regular data because big data is generated at a rate that is difficult to maintain. Big data can take up a few Terabytes (TB) of space while regular data typically only takes up tens or hundreds of Gigabytes (GB). That's almost 100 times larger than regular data! Big data often requires a lot of effort to get it "clean" or ready to use and interpret. For example, think about how you might text your friend a simple phrase like "ok". You might text "okay", "ok", "k", "kay", and many more! This might also vary with how we use uppercase or lowercase letters and special characters as well. All of these possible choices created data and we need to train the computer to recognize that all of these words all mean the same thing.



Despite the sheer amount of data available, most of it may be considered unusable due to the existence of bias. What is bias? Bias in data occurs when a particular result might be more favorable to certain outcomes. This can happen for a variety of reasons, one being that the information gathered is not representative of the population as a whole. Let's say you were surveying your people at school. If you only asked the first 20 people you see, do you think your data is representative of the whole school? Not really. Not only is it important *who* or *what* you survey to represent the whole population, but also what kind of information you gather. To obtain a diverse dataset, it is important to have diversity in your population and the team who will be conducting the analysis. If you want to learn more about data bias, check out this [article](#) by Elder Research or this [video](#) by Google.

- ◆ [Statistical & Cognitive Biases in Data Science: What is Bias?](#) by Elder Research
- ◆ [Machine Learning and Human Bias Video](#) by Google.
- ◆ [Dealing with Bias in Artificial Intelligence](#) by The New York Times

Step 1: What is Big Data? (cont.)

Getting Started with Kaggle (5-8 mins)



Kaggle is a website that hosts real data that is contributed by an online community. Data hosted on this website ranges from statistics on [COVID-19](#), [YouTube videos](#), [Apps on Google Play](#), [breast cancer](#), [avocado prices](#), and [even Pokémon](#)! It is a great place to explore real data that affects the decisions we make today. Within Kaggle, you can create a **notebook** associated with a particular dataset to write code within the website! This is a great way to organize your projects as well as submit work to real live projects!

- **Create a Kaggle Account.** Click this [link](#) to create an account on Kaggle. Alternatively, you can click the **Register** button at the top-right of the Kaggle website to create an account. *If you are under 13, you'll need your guardian's permission and email address to sign up.*



- **Open the [Kickstarter dataset](#).** At the top of this dataset there is a main header containing the title of the dataset, creator, and when it was last updated. Right below this several options are presented: Data, Tasks, Notebooks, Discussion, Activity, and Metadata. You can learn more about features Kaggle provides for each dataset [here](#).

Step 2: Explore the dataset (10-15 mins)

Before we get started on depicting the data, we need to examine the dataset we chose for this activity. It is important to understand what data is represented and to see if there exists any bias.

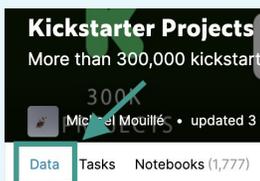
Kickstarter Projects (1 min)



Kickstarter is a crowdfunding platform where an individual or small business can fund creative projects through community investments. Those looking to invest in a project make a pledge of a certain amount. In return for their pledge they are awarded gifts at the completion of their projects. You can explore some examples of projects hosted on Kickstarter [here](#).

You may be thinking, *why did we choose this dataset?* This Kickstarter data contains values that are both [categorical](#), or data that can be grouped into categories, and numerical. We chose it because it gives you a nice snapshot of different types of data. In this activity we will walk you through a basic example of *some* of the things you can do to visualize both types of values.

Setting Up Your View (5-8 min)



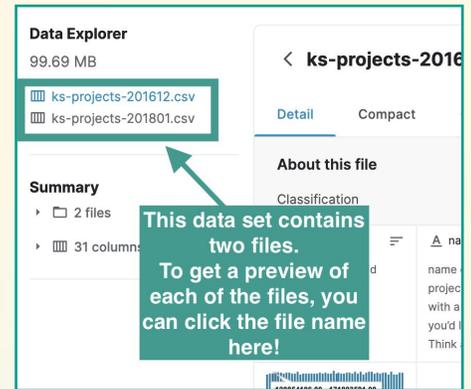
Let's dive into our data! First, let's open the [Kaggle Kickstarter dataset](#). Make sure that you are on the **Data** tab. You should notice that the word "Data" at the top header bar is [blue](#) and [underlined](#). Next, scroll down to the **Data Explorer** part.

Step 2: Explore the dataset (cont.)

Taking a look on the left side, you may notice that there are two files, [ks-projects-201612.csv](#) and [ks-projects-201801.csv](#).

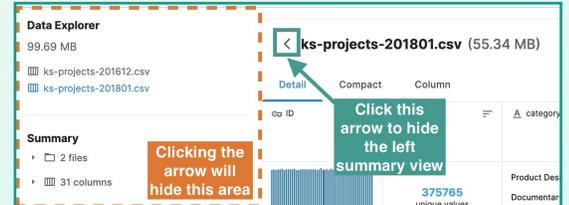
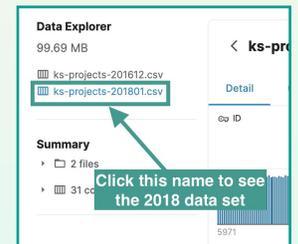
- ◆ [ks-projects-201612.csv](#): This dataset contains all Kickstarter projects that were launched before December 2016.
- ◆ [ks-projects-201801.csv](#): This dataset contains all Kickstarter projects that were launched before January 2018.

Since we want to work with the most recent data, we will only be working with [ks-projects-201801.csv](#), or for short we will call this the 2018 dataset.



Since the [ks-projects-201801.csv](#) contains all projects before January 2018, this dataset contains ALL of the data in [ks-projects-201612.csv](#) and more!

- ◆ **Open the 2018 Kickstarter Dataset.** In the **Data Explorer**, click on the name [ks-projects-201801.csv](#) on the left side. You should then see this name highlighted in blue.
- ◆ **Expand the View.** Click the box icon  on the top-right corner of the data explorer window.
- ◆ **Hide the Summary Information.** Click the arrow to the left of the dataset name. This should close the left hand side giving even more space for us to look at the data!



This view gives us a quick snapshot of the data and some of the values inside. Note, this does not display the full dataset because it is very large! In fact, the entire dataset contains information for 375,765 Kickstarter projects!

Kickstarter Project Features (1 min)

The *columns* of this dataset are called **features** of the data. **Features** let us know what type of information is captured for each project. Each *row* of the dataset represents a single Kickstarter project, or an **entity**. For example, if we had a dataset on humans, a person would be an *entity* in the data set and some *features* we might include in this data set would be a person's name, eye color, hair color, date of birth, etc.

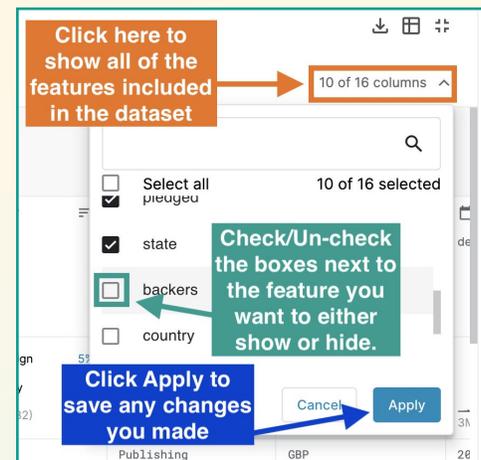
	Name	Eye Color	Hair Color	Date of Birth
Example	Reshma Saujani	Brown	Brown	November 18, 1975

Example of a single **entity**, Reshma Saujani, and some of the **features** that define her.

Step 2: Explore the Dataset (cont.)

Let's breakdown the **features**, or columns, in this Kickstarter data! This dataset has a total of 15 features: **ID, name, category, main_category, currency, deadline, goal, launched, pledged, state, backers, country, usd pledged, usd pledged_real, usd_goal_real**. The first row tells us the name of the feature and a brief description while the second row provides a visual snapshot of the some of the values for each feature. Take a minute to review each of the features and their short description.

Note: In the data explorer you can only see 10 of the 15 columns, there is a drop-down menu on the top-right corner of the dataset that allows you to view additional columns. You will need to scroll down in the menu to find the additional 5 columns that are not shown in the default view.



Step 3: Intro to Python & Pandas (20-25 mins)

Data scientists use a number of different tools and languages to process and analyze data, including [R](#), [Structured Query Language \(or SQL\)](#), and [Python](#). In this activity we will focus on using the Python programming language to help us manipulate our data with the help of a Python library, [Pandas](#).

Python (2 mins)



Python is a text-based programming language which means that all commands will need to be typed! Many programmers choose to use Python because it is easy to learn and understand. Python is an **open source** language, meaning that it is freely available for the public to use and modify as necessary. There are strict guidelines on how updates are accepted and implemented to the language, but essentially anyone can contribute to the evolution of the language!

Since Python is an open source language, this has led to the community to develop additional libraries. A **library** is a collection of [methods](#) and [variables](#). A library can make writing code easier since we can just use the commands in a library to do an action instead of writing several lines of code to do the same thing. In this activity, we will be using the **pandas** library. This special library has been made specifically for data scientists to analyze datasets easily without having to write a lot of lines of code to do simple actions like search, filter, compare, modify, and remove information from a dataset. Before we start diving into how we will use **pandas** specifically to do actions on our data, let's set up our programming environment in Kaggle by creating a new notebook.

Step 3: Intro to Python & Pandas (cont.)

Creating a New Notebook (5-8 mins)

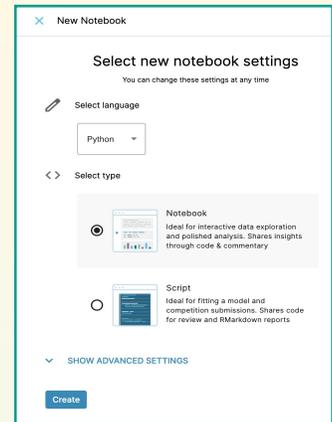
Let's go back to our [Kickstarter dataset](#).

- **Create a New Notebook.** Click the **New Notebook** button under the main header image to the right. This should bring you to a screen to select new notebook settings. Be sure that your notebook has the following:

- ◆ **Language:** Python
- ◆ **Type:** Notebook

Confirm Settings and Click **Create**.

Your dataset may still be displaying in full-view. To **minimize it**, click the **box icon** on the top-right corner.



If you have programmed in Python before, this “notebook” might look different from programming in Trinket or other Python editors. Kaggle uses a tool called [Jupyter Notebook](#) to program in Python. Jupyter Notebook is a tool used application that can be used to show both code, text, and visualization all in one place. It is easy to run blocks of code without having to run the entire program.

- **Rename Your Notebook.** To do this, click the default name at the top-left of your screen and replace the text with your new title. Make sure your name tells us a little about what you will be accomplishing in this project, like “Depicting Data Exploration”. You may want to include your name in the title as well, be sure to only display either initials or your first name and last initial.

Exploring the Starter Code (3-5 mins)

Let's take a moment to take a look at some of the starter code that is included in your notebook. You may notice a few lines of code starting with the `#` symbol throughout the starter code. These lines of code are **code comments**. These are used by programmers to help organize their code making it more readable. In Python, any text written after a `#` symbol is considered a code comment and it is then colored in teal.



Next, let's look at the first lines of code, starting with the keyword `import`.

PYTHON	DESCRIPTION
<pre>import numpy as np import pandas as pd</pre>	<ul style="list-style-type: none">◆ import: This keyword lets the computer know that we are using a Python library.◆ as: This keyword gives a nickname to the package. This is an optional step but can make programming easier.◆ numpy/np: This Python library is used to do mathematical operations on the dataset. We have given this package a nickname of <code>np</code>.◆ pandas/pd: This Python library is used to convert the dataset into a more usable format for analysis. We have given this package a nickname of <code>pd</code>.

We will not be using the [numpy](#) library in this activity. As you continue to explore data analytics on your own, feel free to learn more about how data scientists use the numpy library.

Step 3: Intro to Python & Pandas (cont.)

Finally, let's take at the last lines of code.

PYTHON	DESCRIPTION
<pre>import os for dirname, _, filenames in os.walk('/kaggle/input'): for filename in filenames: print(os.path.join(dirname, filename))</pre>	One of the best features of the notebook in Kaggle is that you can use the dataset easily without having to do additional steps. How? Kaggle already associates the dataset you are interested in with your notebook! These optional lines of code helps let us know what the filename is for our dataset.

Running Code (2 mins)

First, click anywhere inside the code block. Click the **blue play button**  that appears to the left of the window. This should run all of the lines of code inside the code block and display any output right below the window. When you run these lines of code you should get the following output:

```
/kaggle/input/kickstarter-projects/ks-projects-201801.csv
/kaggle/input/kickstarter-projects/ks-projects-201612.csv
```

These are the file names of our dataset! Now that we understand a little bit about what is included in the starter code and how to run code in our notebook, let's start coding!

Importing the Kickstarter Dataset (10-15 mins)

Now that we know the file names of our dataset, we need to actually import the data into our program. Right now it lives as a separate file, but we need to connect this information with our Python program. To do this, we will use the `read_csv()` method from the `pandas` library to help us. Remember that a **method/function** is a set of instructions (lines of code) that performs a specific task. Let's break down the syntax of this method.

CSV, short for comma separated values, is a type of file used to hold data in plain text. The `pandas` library can read CSV files easily and convert it into a table-like format for us to easily read and manipulate.

PYTHON	DESCRIPTION
<pre>pd.read_csv("filename")</pre>	<ul style="list-style-type: none">◆ pd: This keyword lets the computer know that we are using a method from the <code>pandas</code> library. We use <code>pd</code> instead of <code>pandas</code> because this is the nickname we gave the library when importing the library at the start of our program.◆ <code>.</code>: This symbol lets the computer know that we are using a method.◆ <code>read_csv()</code>: This method in <code>pandas</code> reads in a CSV file and converts it to a table-like format so that it is easier to use in Python.◆ "filename": We need to tell the computer which file to open! Add the filename to the CSV file here. We include the name in quotation marks (can be single or double quotes) since it is a name.

Step 3: Intro to Python & Pandas (cont.)

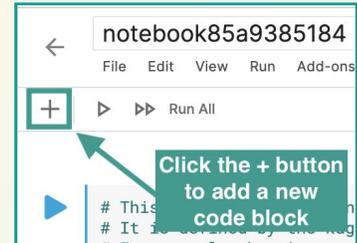
- **Add a New Code Block.** Code blocks are a great way to organize your code to map to the subproblems we defined when we decomposed our problem. You may already have an empty code block in your notebook, indicated with a light-gray box with the symbol [] to the left of the box.



There are two options to add a new code block:

- ◆ Click the + button at the top menu of the notebook.
- ◆ Hover your mouse below your last code block. You should notice an option that reads "+ Code". Select this button to create a new code block below.

- **Import the Kickstarter dataset.** Now that we have a new code block, it's time to use the `read_csv()` method to import our dataset. But wait,



we need the file name of our dataset! Remember that after running the first block, this outputs the file names of our dataset. In fact, this outputted two names, the 2016 and 2018 dataset. We will only use the 2018 Kickstarter dataset since this is the most recent information. In your new code block, use the `read_csv()` method to import the 2018 dataset. **Copy and paste** the 2018 dataset file name from the output of your previous code block.

```
pd.read_csv("/kaggle/input/kickstarter-projects/ks-projects-201801.csv")
```

- **Run your code block.** Click the blue play button to the left of the code block to run your code. You should see a small snapshot (similar to what you saw in the Data Explorer) of your dataset.

RESULTS

ID	name	category	main_category	currency	
0	1000002330	The Songs of Adelaide & Abullah	Poetry	Publishing	GBP
1	1000003930	Greeting From Earth: ZGAC Arts Capsule For ET	Narrative Film	Film & Video	USD
2	1000004038	Where is Hank?	Narrative Film	Film & Video	USD
3	1000007540	Tosh/Capital Rekordz Needs Help to Complete Album	Music	Music	USD
4	1000011046	Community Film Project: The Art of Neighborhood...	Film & Video	Film & Video	USD
...
378656	999976400	ChknTruk Nationwide Charity Drive 2014 (Canceled)	Documentary	Film & Video	USD
378657	999977640	The Tribe	Narrative Film	Film & Video	USD
378658	999986353	Walls of Remedy - New Urban Romantic Comedy I...	Narrative Film	Film & Video	USD
378659	999987933	BioDefense Education Kit	Technology	Technology	USD
378660	999988282	Nou Renmen Ayiti! We Love Haiti!	Performance Art	Art	USD

378661 rows x 5 columns

DEBUGGING TIPS

- ◆ Check that you included the correct 2018 file name: `/kaggle/input/kickstarter-projects/ks-projects-201801.csv`
- ◆ Make sure that there are quotation marks around the file name
- ◆ Remember that in Python, spelling counts! Check that not only everything is spelled correctly but also in the correct case.
- ◆ Check that you include a period when calling a method of **pandas**.
- ◆ Check that you do not have extra parentheses `()`. You may notice that when you type a `(` symbol the notebook automatically adds a closing bracket `)`. This can cause you to add additional brackets by accident.

Step 3: Intro to Python & Pandas (cont.)

- **Store your dataset in a variable named `ds`.** We are almost done! Now that we got our dataset linked to our Python program, we need to store this in a variable. Remember that [variables](#) are used to store information (data) in a computer program. Store your dataset in a variable named `ds` (an abbreviation of dataset).

```
ds = pd.read_csv("/kaggle/input/kickstarter-projects/ks-projects-201801.csv")
```

- **Use the `info()` method to print out information about your dataset.** We can use the `info()` method to get a quick snapshot of the features and number of rows this dataset.

PYTHON	DESCRIPTION
<code>ds.info()</code>	<ul style="list-style-type: none">◆ <code>ds</code>: We use the <code>info()</code> method on our variable, <code>ds</code>, NOT the whole <code>pandas</code> library. This is because we want to know information about our specific dataset.◆ <code>info()</code>: This method in <code>pandas</code> gets information about the dataset, including features, rows, and data type.

- **Run your code block.** Click the blue play button to the left of the code block to run your code. This should output some information about your dataset, including the number of entries (or number of Kickstarter projects) and number of features. This also includes the [type of values](#) of each feature (number or words) and the number of entries that are “**non-null**” or not empty.

RESULTS

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 378661 entries, 0 to 378660
Data columns (total 15 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ID                    378661 non-null  int64
1   name                  378657 non-null  object
2   category              378661 non-null  object
3   main_category        378661 non-null  object
4   currency              378661 non-null  object
5   deadline              378661 non-null  object
6   goal                  378661 non-null  float64
7   launched              378661 non-null  object
8   pledged               378661 non-null  float64
9   state                 378661 non-null  object
10  backers               378661 non-null  int64
11  country               378661 non-null  object
12  usd pledged           374864 non-null  float64
13  usd_pledged_real     378661 non-null  float64
14  usd_goal_real         378661 non-null  float64
dtypes: float64(5), int64(2), object(8)
memory usage: 43.3+ MB
```

DEBUGGING TIPS

- ◆ **Name 'ds' is not defined:** Each code block in your notebook runs separately, sometimes your notebook might lose its place in the program and require you to **run all** of your code blocks. To do this, click the double arrow button `▶▶` at the top of your notebook to run all code blocks.
- ◆ Remember that in Python, spelling counts! Check that not only everything is spelled correctly but also in the correct case.
- ◆ Check that you include a period when calling a method.

Step 4: Understanding Data in Pandas (15-20 mins)

Before we begin learning how to code visual graphs, let's take a moment to explore the different types of data and how the **pandas** library processes a dataset.



Categorical vs. Numerical Data (2-4 mins)

Data can be classified into two different types: **categorical** or **numerical**. **Categorical data** is data that can be divided into categories or groups while **numerical data** is data that can be expressed in the form of a number. Numerical data must contain data that represents a continuous value (where no numbers are skipped) within a certain range. Sometimes numbers may represent categories, for example, taking a survey where they ask you to rate your experience with a number between 1 to 5. Here we can see that experience is *grouped* by the numbers 1-5. Take a moment and try to classify the following examples as either categorical or numerical.

- ◆ Lengths of cats in centimeters
- ◆ Final grades for a class (A, B, C, D, or F)
- ◆ Representing the Months in a Year with Numbers (1-12)
- ◆ Amount of Sleep in hours
- ◆ People's Favorite Colors

Pause here before checking the solutions below.

- ◆ Lengths of cats in centimeters → **Numerical**
- ◆ Final grades for a class (A, B, C, D, or F) → **Categorical**
- ◆ Representing the Months in a Year with Numbers (1-12) → **Categorical**
- ◆ Amount of Sleep in hours → **Numerical**
- ◆ People's Favorite Colors → **Categorical**

Pandas DataFrame (5-8 mins)

In **pandas**, data is stored in a table-like object called a **DataFrame**. It stores the data similar to how we viewed it in the Data Explorer on Kaggle. A **DataFrame** mimics the data structure's rows and columns. To access the data, we use `[]` symbols. If you are familiar with **Arrays** in JavaScript and **Lists** in Python, this is a very similar concept.

The diagram shows a table with 7 rows and 5 columns. The columns are labeled 'Name', 'Team', 'Number', 'Position', and 'Age'. The rows are labeled '0' through '6'. A bracket on the left side of the table is labeled 'Rows', and a bracket at the bottom is labeled 'Data'. The 'Number' column has a value of 8.0 in row 2, which is highlighted with a red box. The 'Position' column has a value of 'C' in row 5, which is highlighted with a red box. The 'Age' column has a value of 'NaN' in row 5, which is highlighted with a red box. The 'Number' column has a value of 'NaN' in row 3, which is highlighted with a red box. The 'Position' column has a value of 'PG' in row 4, which is highlighted with a red box. The 'Age' column has a value of '22.0' in row 4, which is highlighted with a red box. The 'Number' column has a value of '12.0' in row 4, which is highlighted with a red box. The 'Position' column has a value of 'PF' in row 3, which is highlighted with a red box. The 'Age' column has a value of '21.0' in row 3, which is highlighted with a red box. The 'Number' column has a value of '30.0' in row 1, which is highlighted with a red box. The 'Position' column has a value of 'SG' in row 1, which is highlighted with a red box. The 'Age' column has a value of '27.0' in row 1, which is highlighted with a red box. The 'Number' column has a value of '0.0' in row 0, which is highlighted with a red box. The 'Position' column has a value of 'PG' in row 0, which is highlighted with a red box. The 'Age' column has a value of '25.0' in row 0, which is highlighted with a red box. The 'Number' column has a value of '11.0' in row 6, which is highlighted with a red box. The 'Position' column has a value of 'SG' in row 6, which is highlighted with a red box. The 'Age' column has a value of '27.0' in row 6, which is highlighted with a red box. The 'Number' column has a value of '7.0' in row 5, which is highlighted with a red box. The 'Position' column has a value of 'C' in row 5, which is highlighted with a red box. The 'Age' column has a value of '22.0' in row 5, which is highlighted with a red box. The 'Number' column has a value of '12.0' in row 4, which is highlighted with a red box. The 'Position' column has a value of 'PF' in row 4, which is highlighted with a red box. The 'Age' column has a value of '21.0' in row 4, which is highlighted with a red box. The 'Number' column has a value of 'NaN' in row 3, which is highlighted with a red box. The 'Position' column has a value of 'PF' in row 3, which is highlighted with a red box. The 'Age' column has a value of '22.0' in row 3, which is highlighted with a red box. The 'Number' column has a value of '30.0' in row 1, which is highlighted with a red box. The 'Position' column has a value of 'SG' in row 1, which is highlighted with a red box. The 'Age' column has a value of '27.0' in row 1, which is highlighted with a red box. The 'Number' column has a value of '0.0' in row 0, which is highlighted with a red box. The 'Position' column has a value of 'PG' in row 0, which is highlighted with a red box. The 'Age' column has a value of '25.0' in row 0, which is highlighted with a red box.

	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

Image Source: [GeeksforGeeks](https://www.geeksforgeeks.org/)

With a total of **15 features**, or columns, we might want to only focus on a few features in the dataset. Let's practice on how we can use the `[]` symbols to select specific features.

Step 4: Understanding Data in Pandas (cont.)

- **Add a new code block at the bottom of your notebook.** Either hover your mouse below your last code block and press the **+Code** button or press the **+** button at the top-left of your notebook.
- **Use the `[]` symbols on our dataset variable, `ds`, to select the `'state'` feature.** Remember that we stored the reference to our dataset in a variable named `ds`. Here we use the `[]` symbols to indicate that we want to select information from the dataset and then include the desired features *inside* of the square brackets.

PYTHON	DESCRIPTION
<code>ds["state"]</code>	<ul style="list-style-type: none">◆ <code>ds</code>: The variable that stores the Kickstarter dataset.◆ <code>[]</code>: This symbol is used to access information from our dataset.◆ <code>"state"</code>: We indicate which feature we want to select <i>inside</i> the <code>[]</code>. Features must be included in quotation marks since it is a name.

- **Run your code block.** Click the blue play button to the left of the code block to run your code. You should get a snapshot of all of the data in your dataset but only displaying the `main_category` and `state` features. If your code doesn't run properly, check the following:

RESULTS

```
0      failed
1      failed
2      failed
3      failed
4      canceled
...
378656  canceled
378657  failed
378658  failed
378659  failed
378660  failed
```

DEBUGGING TIPS

- ◆ **Name 'ds' is not defined:** Each code block in your notebook runs separately, sometimes your notebook might lose its place in the program and require you to **run all** of your code blocks. To do this, click the double arrow button at the top of your notebook to run all code blocks.
- ◆ Did you use quotation marks around names of features? Check that you use quotation marks around the names of your features. Be sure to *type* all lines of code because quotation marks do not copy with the correct character when coding.
- ◆ Did you spell names and variables correctly? Check that your feature names are spelled correctly. Remember that Python is also case sensitive.
- ◆ Do you have extra square brackets `[]`. You may notice that when you type a `[` symbol the notebook automatically adds a closing bracket `]`. This can cause you to add additional brackets by accident. You should have a total of two sets of brackets: one around the feature names, and the other surrounding the list of features.

Step 5: Meet the `plot()` method (2 mins)

The **pandas** library is a powerful Python library that provides tools to help analyze, manipulate, and even visualize information in a dataset. Allow us to introduce the `plot()` method in **pandas**.

PYTHON	DESCRIPTION
<code>ds.plot()</code>	<ul style="list-style-type: none">◆ ds: The variable that stores the Kickstarter dataset.◆ plot(): This method in pandas gets information in the dataset and displays it in the form of a graph. We use the <code>plot()</code> method on our variable, ds, NOT the whole pandas library. This is because we want to plot information stored in our dataset variable.

The `plot()` method is a very powerful method that helps take the data stored in our dataset and convert it into different graphs, including bar graph, line graph, box plot, scatter plot, histogram, and many more. In order to specify the type of graph and identify which data should be displayed, the `plot()` method takes in a number of different **parameters**, or inputs. In this activity we will highlight the use of some of these parameters, in particular the **kind** attribute. The **kind** parameter takes in a String, or word, input that allows us to tell the computer what type of graph we want displayed. Some possible input values include, 'pie', 'bar', 'scatter', etc. To learn more about the many different inputs the `plot()` method takes you can explore the official [pandas documentation](#). We will use this method in the next steps to graph categorical and numerical data.

Step 6: Displaying Categorical Data (20-30 mins)

Recall that **categorical data** is information that can be grouped into categories. In this section we will review different types of graphs that can best display categorical data. We will highlight key features of each graph and then and then code it in Python using the `plot()` method.

Pie Chart (2 mins)

A **pie chart** is a type of graph in the shape of a circle where data takes up a certain percentage of the circle and represented in the shape of a slice (like a pizza slice!). Let's break down the components of a pie chart:

- ◆ **Title**: Usually at the top of the graph and let us know what is being described.
- ◆ **Slice**: Each slice represents a percentage of a specific data compared to the whole dataset. Sometimes the percentage is also displayed next to each slice.
- ◆ **Legend**: This tells us which color represents which data value. Sometimes this appears as a list on the side of the graph or labeled individually next to each slice.

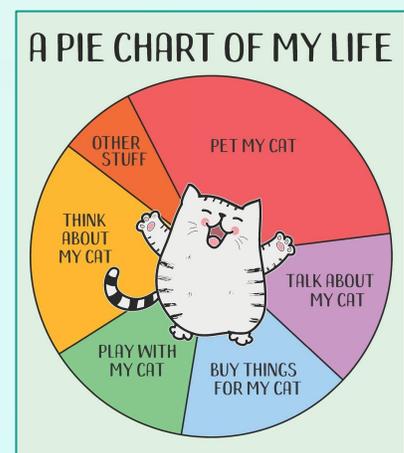


Image Source: [Amazon](#)

Questions to consider when choosing a pie chart:

- ◆ Do you have categorical data?
- ◆ Is it important to compare the number of data points in a category to the dataset as a whole?¹⁴
- ◆ Do you have a lot of categories? Consider the size of each slice. If you have a lot of small slices it may be difficult to read in a pie chart.

Step 6: Displaying Categorical Data (cont.)

Graphing Pie Charts in Pandas (5-8 mins)

Open back up your notebook in Kaggle. We will be walking through how you can use the `plot()` method in `pandas` to display a pie chart with the `kind` attribute. Our Kickstarter dataset represents one project's information on one row, but doesn't really contain specific rows that tells us about *all* of the projects in our dataset. For example, what if we wanted to know the number of projects classified as Film and Video? Or the number of projects in each state? To get this information we will use the `value_counts()` method that lets us know the breakdown of projects in a specific feature. The `value_counts()` method helps us group values based on certain features, like finding out the number of projects in each category, so that we can use the `plot()` method to display a graph that is representative of the whole dataset.

In this example we will walk you through how to graph a pie chart to display the breakdown of projects based on `state`. The `state` features is the current status of a Kickstarter project: successful, failed, suspended, cancelled, live, or undefined.

- **Add a new code block at the bottom of your notebook.** Either hover your mouse below your last code block and press the `+Code` button or press the `+` button at the top-left of your notebook.
- **Select the `state` feature from the dataset.** Remember that we stored the reference to our dataset in a variable named `ds`. Next we include the feature "`state`" inside the square brackets `[]` symbols to indicate that we want only select this column from the dataset.
- **Use the `value_counts()` method on the filtered dataset.**

PYTHON	DESCRIPTION
<pre>ds["state"].value_counts()</pre>	<ul style="list-style-type: none">◆ <code>ds["state"]</code>: We want to know the breakdown of projects based on <code>state</code>. We use the <code>[]</code> symbols to select only the <code>state</code> feature in our dataset and use double quotation marks around the feature to indicate that it is a name.◆ <code>.</code>: This symbol lets the computer know that we are using a method.◆ <code>value_counts()</code>: This method returns the number of entities (or rows) associated with each unique feature value.

- **Run your code block.** Click the blue play button to the left of the code block to run your code. You should have a result identical to the breakdown below:

RESULTS	
failed	197719
successful	133956
cancelled	38779
undefined	3562
live	2799
suspended	1846

DEBUGGING TIPS	
◆	Do you need to rerun all code blocks?
◆	Did you use quotation marks around any names?
◆	Did you spell names and variables correctly? Remember that Python is also case sensitive
◆	Do you have extra square brackets <code>[]</code> ?

Step 6: Displaying Categorical Data (cont.)

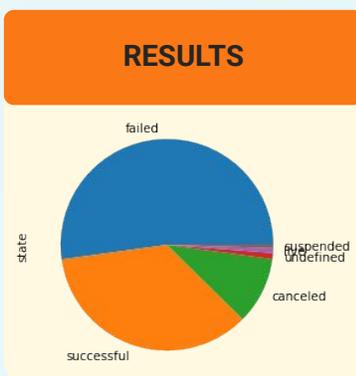
- **Store the data we just output in a new variable.** We want to be able to transform the data we just got from our state feature into a pie chart. This is much easier to do if we store it in a variable. You can name your variable anything, just be sure that your variable name is descriptive and follows [camelCase](#). For our example, we named our variable `projState`. Be sure to keep this in mind if you named your variable differently.

```
projState = ds['state'].value_counts()
```

- **Use the `plot()` method on your state dataset and set the `kind` parameter to `'pie'`.** This is the line of code that actually tells our computer to generate a pie chart based on the `state` feature data we just stored in the `projState` variable.

PYTHON	DESCRIPTION
<pre>projState.plot(kind = 'pie')</pre>	<ul style="list-style-type: none">◆ projState: We specifically want to display the breakdown of the project states in the form of a pie chart. .◆ <code>.</code> : This symbol lets the computer know that we are using a method.◆ plot(): This method in pandas gets information in the dataset and displays it in the form of a graph.◆ kind = 'pie': We use the kind parameter (or input) and set this equal to the name of the graph we want to display. In this case we display a pie chart so we set the value to <code>'pie'</code>. The type of graph must be in quotation marks since it is the <i>name</i> of the graph.

- **Run your code block.** Click the blue play button to the left of the code block to run your code. You should have a result identical to the breakdown below:



DEBUGGING TIPS

- ◆ Do you need to rerun all code blocks?
- ◆ Did you use quotation marks around any names?
- ◆ Did you spell names and variables correctly? Remember that Python is also case sensitive.
- ◆ Did you use parentheses `()` around the method?

Step 6: Displaying Categorical Data(cont.)

Bar Graph (2 mins)

A **bar graph** is a graph that displays data with the use of bars at varying heights. Let's break down some of the components of a bar graph:

- **Title**
- **Bars:** Each bar represents a category in the data. The height of the bar graph is dependent on what is being measured on the other axis. There should be a small gap between the bars.
- **Orientation:** In the example on the right, we see the use of vertical bars. Bar graphs may contain horizontal bars as well.

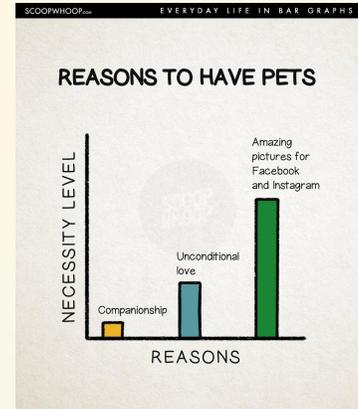


Image Source: [ScoopWoop](#)

Questions to consider when choosing a pie chart:

- ◆ Do you have categorical data?
- ◆ Is it important to display the exact amount of each category? Bar graphs depict the **actual values** of each category, while pie charts depict the *percentage* of each category compared to the whole dataset.

Graphing Bar Graphs in Pandas (5-8 mins)

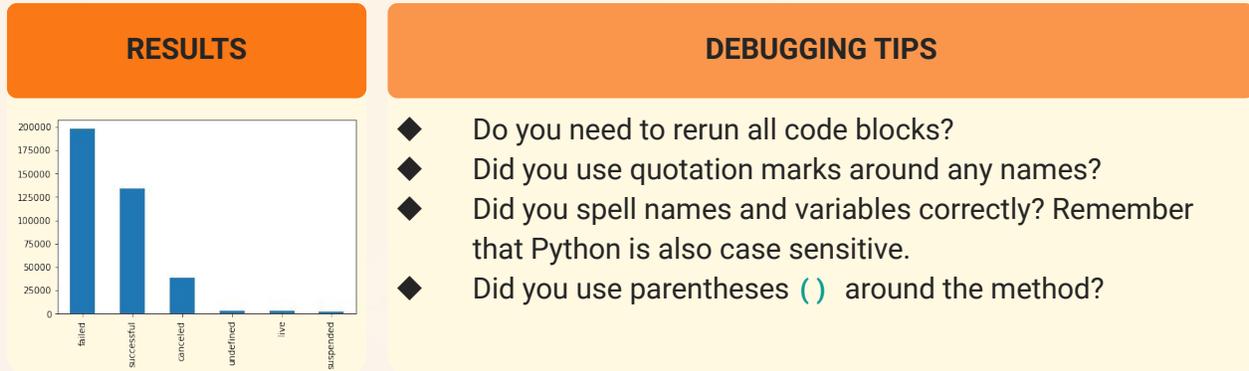
In this example we will continue to depict the state of Kickstarter projects in a bar graph this time. This way you can easily compare pie charts and bar graphs. This time when using the `plot()` method, we will set the `kind` parameter to `"bar"`.

- **Add a new code block at the bottom of your notebook.**
- **Use the `plot()` method on your state dataset and set the `kind` parameter to `'bar'`.** Remember that we stored the dataset containing the number of projects with each state in the variable `projState`. You may have named this variable differently.

PYTHON	DESCRIPTION
<pre>projState.plot(kind = 'bar')</pre>	<ul style="list-style-type: none">◆ projState: We specifically want to display the breakdown of the state of projects.◆ . : This symbol lets the computer know that we are using a method.◆ plot(): This method in pandas gets information in the dataset and displays it in the form of a graph.◆ kind = 'bar': We use the kind parameter (or input) and set this equal to the name of the graph we want to display. In this case we display a bar graph so we set the value to <code>'bar'</code>. The type of graph must be in quotation marks since it is the <i>name</i> of the graph.

Step 6: Displaying Categorical Data (cont.)

→ **Run your code block.** Click the blue play button to the left of the code block to run your code. You should have a result identical to the breakdown below:

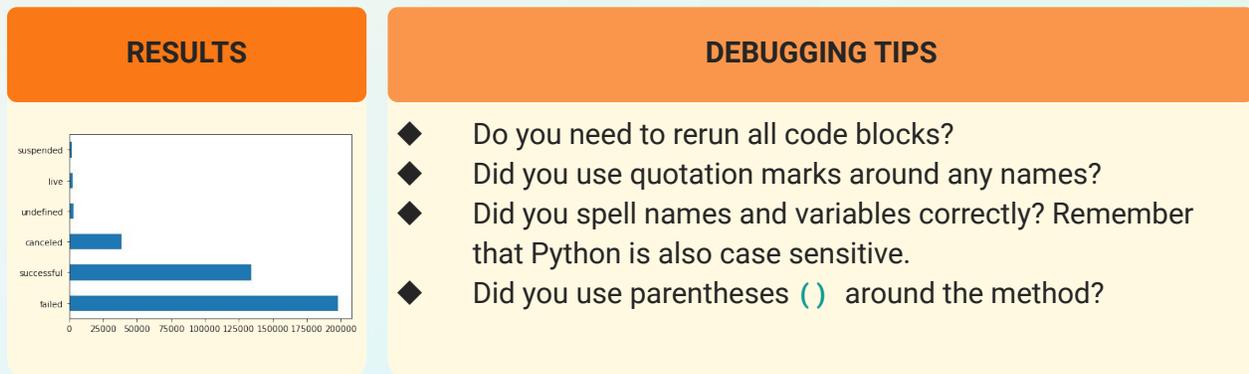


Want to change this to a horizontal bar graph? It's simple, this time when using the `plot()` method, we will set the `kind` parameter to `"barh"`.

→ **Add a new code block at the bottom of your notebook.**

→ **Use the `plot()` method on your state dataset and set the `kind` parameter to `'barh'`.**
`projState.plot(kind = 'barh')`

→ **Run your code block.** Click the blue play button to the left of the code block to run your code. You should have a result identical to the breakdown below:



Comparing Pie Charts and Bar Graphs (5-8 mins)

Take a moment to review the pie chart and bar graphs you just created and ask the following questions:

QUESTION 1

What can you conclude about the data based on the graphs?

QUESTION 2

Which graph emphasizes your conclusion(s) the most?

Step 6: Displaying Categorical Data (cont.)

Take **2 minutes** to write down as many conclusions you can make based on the graphs you created. It may be helpful to organize your thoughts in the form of a table like the one below.

CONCLUSION <i>What can you conclude about the data based on the graphs?</i>	BEST GRAPH <i>Which graph emphasizes your conclusion the most?</i>

Pause here before revealing some of the conclusions we found below.

There are many conclusions that can be made about the data, you can review some of the ones we created. Your answers on which graph articulates the conclusion best may be slightly different and that is okay! The most important thing is that you are able to justify your decision.

CONCLUSION <i>What can you conclude about the data based on the graphs?</i>	BEST GRAPH <i>Which graph emphasizes your conclusion the most?</i>
The majority of projects are “failed”	Pie Chart
Close to 200,000 of Kickstarter projects resulted in a “failed” state	Bar Graph
Close to 50% of Kickstarter projects are failed projects	Pie Chart
The number of undefined, live, and suspended projects are about the same	Bar Graph

Remember that pie charts are great when seeing the breakdown of all projects and how they compare based on percentages. Bar graphs are great at seeing the breakdown of projects and their exact values. Choosing between a horizontal and vertical bar graph comes down to what you prefer. Both display the same exact data but with a different orientation.

Step 7: Displaying Numerical Data (25-35 mins)



Recall that **numerical data** is data that can be expressed in the form of a number. It must have a continuous range (meaning that all numbers in a range are represented). When considering graphic numerical data we must also identify the **independent** and **dependent** variables.

Independent vs Dependent Variables (2 Mins)

A graph depicts the comparison between two variables. For example, the graphs we created with our pie chart and bar graph compared the number of projects and state for Kickstarter projects. We call the two variables that are being compared an independent and dependent variable. The **independent variable** is data where the amount of change is controlled or predictable. In our previous example the variable or data that remains constant are the various types of states (i.e. live, failed, suspended, etc). The **dependent variable** is the value that is changed *depending* on the value of the independent variable.

In our example, this would be the number of projects since it is dependent on the state of the project.

When comparing numerical values, the independent variable is displayed on the **x-axis** or the horizontal axis while the dependent variable is displayed on the **y-axis** or vertical axis. We need to specify these fields when graphing numerical values in **pandas**.

Line Graphs (2 mins)

A **line graph** is a graph that displays data that changes over time. A line graph uses points to represent specific data points that are connected by a line to show the trend over time. A line graph can be used to display **both** categorical and numerical data. A line graph has a few key components:

- ◆ **Title**
- ◆ **X-Axis:** On this horizontal axis we include the **independent variable** of the data.
- ◆ **Y-Axis:** On this vertical axis we include the **dependent variable** of the data.
- ◆ **Trend:** On a line graph, each of the individual data points are connected by a line. This helps highlight the trend of the data to see increases and decreases easily. In the graph above we see that on days where there is a downward trend, the line is highlighted in red. Sometimes you will see line graphs that distinguish between trends with color, or the entire graph will share the same color.

Questions to consider when choosing a line graph:

- ◆ Do you have an independent variable that measures time? (i.e. days in a week, hours in a day, months in a year, minutes in an hour)
- ◆ Is seeing the trend between each data point important?

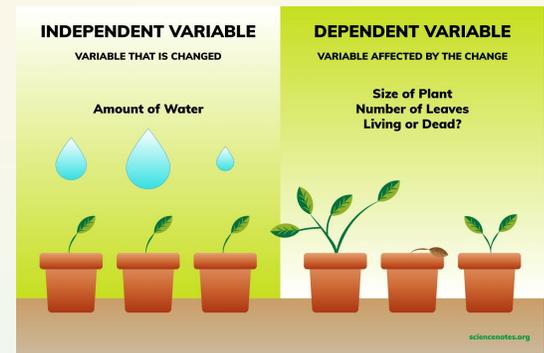


Image Source: [Science Notes](#)

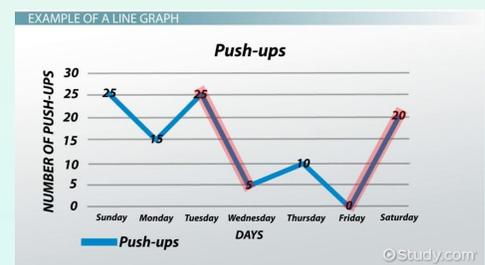


Image Source: [Study.com](#)

Graphing Bar Graphs in Pandas (5-8 mins)

Take a moment to think about which feature can be used to represent *time* in our Kickstarter dataset. Both the **launched** and **deadline** features are measured in time! For this example we will be comparing the **goal** amount for Kickstarter projects to the date of **launched**. Remember that we need to consider which feature is our independent variable and which feature is our dependent variable. Think on your own before revealing our answer below.

- ◆ **Graph:** Comparing Date Launched and Goal Amount for Kickstarter Projects
- ◆ **Independent Variable:** **deadline**. Here time remains constant!
- ◆ **Dependent Variable:** **goal**. The amount a Kickstarter project sets as its goal may vary depending on time which is why this is the dependent variable.

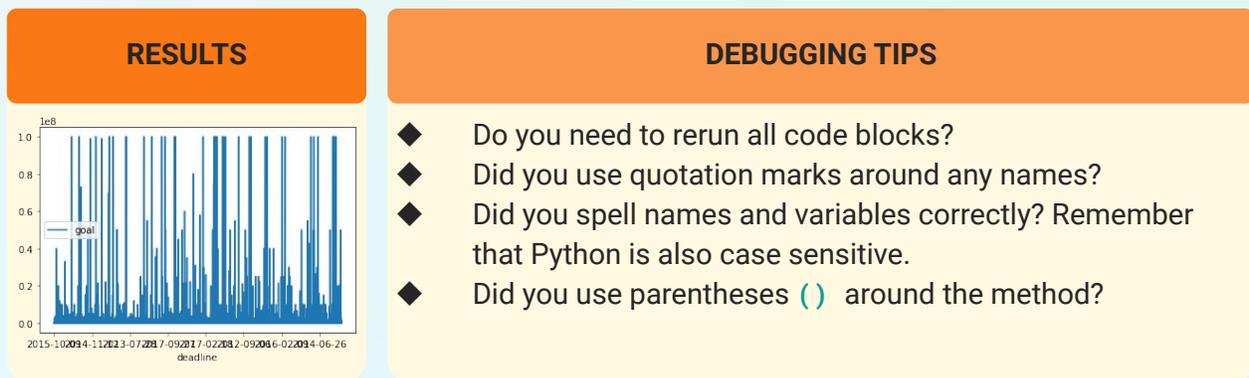
The `plot()` method defaults to a line graph, so it is optional to set the `kind` parameter to "line".

This time when using the `plot()` method, we will need to set the `x` and `y` parameters to the features we want to show and also set the `kind` parameter to "line".

- Add a new code block at the bottom of your notebook.
- Use the `plot()` method on your dataset and set the following parameters:
 - ◆ `x` parameter to your independent variable.
 - ◆ `y` parameter to your dependent variable.
 - ◆ `kind` parameter to 'line'.

```
ds.plot(x='deadline', y='goal', kind = 'line')
```

- **Run your code block.** Click the blue play button to the left of the code block to run your code. You should have a result identical to the breakdown below:



You may notice that this line graph doesn't tell us a whole lot about our data because there is so much information captured in our dataset. This line graph plotted all 378,661 projects as a single point in this graph. That is a lot! We are just trying to get a hang of what each graph is and how to code it in **pandas** - we don't need to plot all the projects.

Step 7: Displaying Numerical Data(cont.)

Let's explore using the `head()` method that only selects the first few rows (or entities) in our dataset.. We will use it to display the first 100 projects in our dataset. Here we will also practice using method chaining to group the `head()` and `plot()` method together.

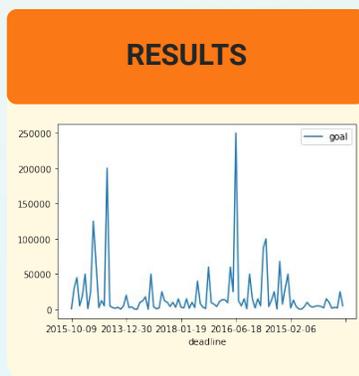
- Add a new code block at the bottom of your notebook.
- Use the `head()` method on your dataset with the parameter 100 to get only the first 100 projects

PYTHON	DESCRIPTION
<code>ds.head(100)</code>	<ul style="list-style-type: none">◆ <code>ds</code>: This is the variable we stored our dataset in.◆ <code>.</code>: This symbol lets the computer know that we are using a method.◆ <code>head()</code>: This method in <code>pandas</code> gets the first few entities (or rows) in our dataset.◆ <code>100</code>: the <code>head()</code> method takes in an integer (or whole number) input which helps specify the number of rows to select. In our example we only want the first 100 rows.

- Chain the `plot()` method on your dataset and set the `x`, `y`, and `kind` parameters:

```
ds.head(100).plot(x='deadline', y='goal', kind = 'line')
```

- Run your code block. Click the blue play button to the left of the code block to run your code. You should have a result identical to the breakdown below:



DEBUGGING TIPS

- ◆ Do you need to rerun all code blocks?
- ◆ Did you use quotation marks around any names?
- ◆ Did you spell names and variables correctly? Remember that Python is also case sensitive.
- ◆ Did you use parentheses `()` around the method?

Step 7: Displaying Numerical Data(cont.)

Scatter Plot (2 mins)

A **scatter plot** is a graph that uses points to represent each data point when comparing two variables. It is similar to a line graph where each data is represented by a point but it is not connected by a line. Scatter plots are often used to display graphs with a lot of data to determine if there is an overall trend or correlation. Trends can be in the form of a line, various types of curves, or non-existent at all.

Let's break down the components of a scatter plot:

- **Title**
- **X-Axis:** On this horizontal axis we include the **independent variable** of the data.
- **Y-Axis:** On this vertical axis we include the **dependent variable** of the data.
- **Trend:** Individual points, or data, are not connected. Instead the trend often tries to summarize or group the data to highlight a specific relationship. If the trend can best be described as a line, you may have heard this called "the line of best fit".

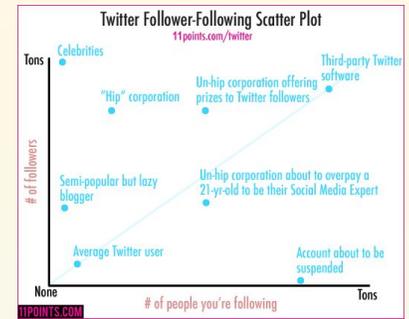


Image Source: [11](#).

Graphing Scatter Plots in Pandas (3-5 mins)

One of the largest challenges we noticed in the line graph was that the lines connecting each point made it difficult to see an overall trend of the data. Let's try to compare again the **goal** amount for a Kickstarter project to the date of **launched**. This time when using the `plot()` method, we will need to set the **x** and **y** parameters to the features we want to show and set the **kind** parameter to "scatter".

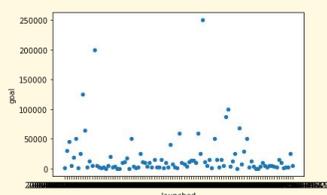
- **Add a new code block at the bottom of your notebook.**
- **Use the `plot()` method on your dataset and set the following parameters:**
 - ◆ **x** parameter to your independent variable.
 - ◆ **y** parameter to your dependent variable.
 - ◆ **kind** parameter to 'scatter'.

You may notice that Kaggle takes a while to run a scatter plot over the whole dataset. If this is taking too long, stop running the code block and use the `head()` method to filter the first 100 data points instead before plotting the scatter plot.

```
ds.plot(x='deadline', y='goal', kind = 'scatter')
```

- **Run your code block.** Click the blue play button to the left of the code block to run your code. You should have a result identical to the breakdown below:

RESULTS



**This graph depicts only the first 100 data points*

DEBUGGING TIPS

- ◆ Do you need to rerun all code blocks?
- ◆ Did you use quotation marks around any names?
- ◆ Did you spell names and variables correctly? Remember that Python is also case sensitive.
- ◆ Did you use parentheses () around the method?

A conclusion we might be able to make from this graph is that most Kickstarter projects fall below a **goal** value of 50,000.

Step 7: Displaying Numerical Data(cont.)

Histogram (2 mins)

A **histogram** is a graph that displays data using bars with varying heights. Unlike bar graphs, histograms group data into ranges. This is why histograms can only be used with **numerical data**. Let's breakdown some of the components of a histogram:

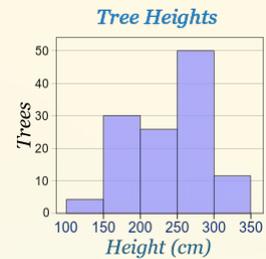


Image Source: [Math is Fun](#)

- ◆ **Title**
- ◆ **Bars:** Each bar represents a range of values in the data. The height of the bar graph measures the number of data points that fall within the range. Unlike bar graphs, there is **no gap** between the bars.
- ◆ **Bins:** This determines the number of bars that should be displayed on the graph. In **pandas** this value is defaulted to 10 bins (or bars).

Graphing Histograms in Pandas (15-20 mins)

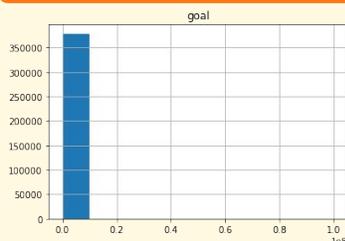
In this example we will break down the ranges of goal amounts in Kickstarter projects. To graph a histogram in **pandas**, we can either use the `plot()` method and set the `kind` parameter to `'hist'` or we can use the `hist()` method. Both methods do the same exact thing but the `hist()` method has specific parameters included that allows us to customize features that are unique to histograms, like the number of **bins**.

- ➔ **Add a new code block at the bottom of your notebook.**
- ➔ **Use the `hist()` method on your dataset and set the `column` parameter to `'goal'` feature.**

PYTHON	DESCRIPTION
<pre>ds.hist(column = 'goal')</pre>	<ul style="list-style-type: none">◆ ds: This is the variable that stores our dataset.◆ . : This symbol lets the computer know that we are using a method.◆ hist(): This method in pandas plots a histogram◆ column = 'goal': We use the column parameter to specify which numerical feature we want to see the breakdown of values.

- ➔ **Run your code block.**

RESULTS

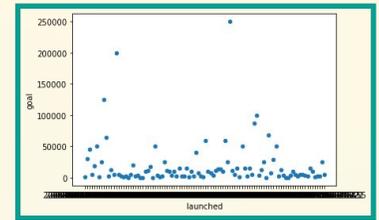


DEBUGGING TIPS

- ◆ Do you need to rerun all code blocks?
- ◆ Did you use quotation marks around any names?
- ◆ Did you spell names and variables correctly? Remember that Python is also case sensitive.
- ◆ Did you use parentheses () around the method?

Step 7: Displaying Numerical Data(cont.)

You may be wondering why you only see one bar in this histogram. The default value for `bins` when using the `hist()` is 10. This means that we should see exactly 10 bars but why do we only see one? Recall our scatter plot when we compared the goal of projects over time. The majority of projects had a goal value of 50,000 or less but there exists some projects with a project goal as



high as 100,000,000 and as low as 0.01. When pandas attempts to split this range into 10 bins each of the bins has a very large range. Most projects will lie in the first bin which is depicted in the histogram. There are projects that exist in the other range, but since they are very small they appear nonexistent in this histogram. Let's do a little **data manipulation** to view projects with a goal of 5000 and under. Remember that in **pandas** we can use the `[]` symbols and add filters inside, like when we wanted to highlight only certain features. In this case we will add a **conditional** to get `goal` values less than 5000.

- Add a new code block at the bottom of your notebook.
- Write a condition on the `goal` feature of the dataset that only gets values less than 5000.

PYTHON	DESCRIPTION
<code>(ds["goal"] < 5000)</code>	<ul style="list-style-type: none">◆ <code>ds["goal"]</code>: We want to apply filtering to our dataset, we use our dataset variable, <code>ds</code>, and then use the square brackets <code>[]</code> to specify filtering on the <code>"goal"</code> feature.◆ <code>< 5000</code>: Since we want only projects with a goal value of less than 5000.◆ <code>()</code>: We wrap the conditional in parentheses so that the computer can distinguish the conditional.

- Apply the conditional you wrote to the dataset inside the `[]`. Here we will use the variable `ds` and the symbols `[]` to apply the conditional statements from the previous step. By adding the conditional statement *inside* the square brackets `[]` we let the computer know to search in the `ds` dataset for all entities (or rows) where the value of the `'goal'` feature is less than 5000.

```
ds[(ds['goal']<5000)]
```

- Store this new filtered dataset in a new variable. We want to store this filtered dataset in a new variable so that we don't modify the original dataset. In our example we name this variable `under5000`.

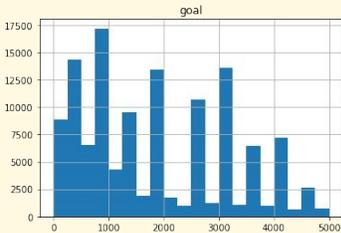
```
under5000 = ds[(ds['goal']<5000)]
```

- Use the `hist()` method on your filtered dataset and set the `column` parameter to `'goal'` feature.

```
under5000.hist(column='goal')
```

→ Run your code block.

RESULTS



DEBUGGING TIPS

- ◆ Do you need to rerun all code blocks?
- ◆ Did you use quotation marks around any names?
- ◆ Did you spell names and variables correctly? Remember that Python is also case sensitive.
- ◆ Did you use parentheses () around the method?
- ◆ Did you use square brackets [] around the dataset?
- ◆ Did you use commas to separate parameters?

By increasing the bins of our graph, we can see that there are peaks of projects at various goal amounts. This was not emphasized as much when we used 10 bins. We can experiment with parameters like `bin` to determine which parameters best fit the data. This process is called **fitting**. There is a fine balance between **overfitting**, trying to correlate too much of the data where outliers are included, and **underfitting**, not correlating enough of the data where the trend isn't accurately measured.

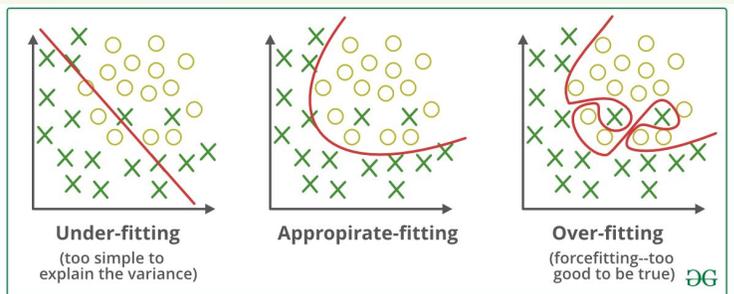


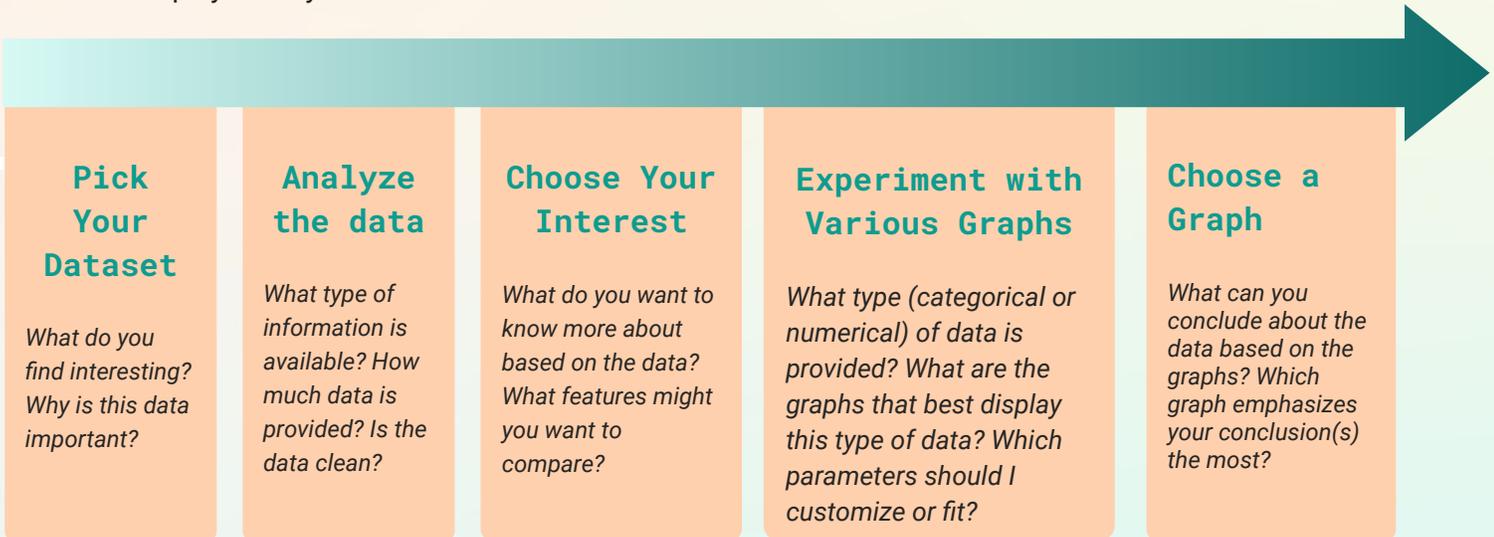
Image Source: [GeeksforGeeks](https://www.geeksforgeeks.com/)

Step 8: Extensions (5-40 mins)

Extension 1: Exploring other Kaggle Datasets (30-40 mins)

There are many datasets available on Kaggle. Take a moment to search through some datasets [here](#). You might want to consider sorting by **Usability**. This score helps you determine how “clean”, or ready to use and interpret, a dataset is.

When analyzing a new dataset, you want to make sure to understand the data deeply before beginning. Here are a few steps you may want to follow.



Once you have picked your dataset and brainstormed ways you can visualize the data, it’s time to start your analysis! In this activity we only scratched the surface of some of the things that **pandas** can help with visualizing data, but **pandas** can do so much more! Take a look at these resources to learn more about this powerful library!

- ◆ [Pandas Documentation: Visualization](#)
- ◆ [Real Python: Plot With Pandas](#)

You may find that your data may need a little updating or manipulation to take your analysis further. We highly recommend checking out our [Data Playground](#) activity around data manipulation in Python.

Extension 2: Customizing Your Graphs (5-15 mins)

The `plot()` method contains many different parameters. Let's take a moment to highlight a few parameters that may be interesting to include.

- ◆ **title:** This parameter takes in a **String**, or word, and displays it at the top of the graph as the title. If you have multiple plots, you can pass in a list of titles and **pandas** will separate the titles accordingly.
- ◆ **grid:** This parameter adds grid lines to your plot. To add grid lines, you must set this parameter to **True**.
- ◆ **legend:** By setting this parameter to **True** you can add a legend to the side of your graph
- ◆ **xlim:** **pandas** automatically configures your graph to the best range of values. You can use this parameter to restrict the range of values on the horizontal axis of your graph. You must input the range in the form of an ordered pair, with the use of parentheses.
- ◆ **ylim:** Similar to `xlim`, the `ylim` can be used to restrict the range of values on the vertical axis of your graph.
- ◆ **xlabel:** This parameter takes in a **String** and adds a label to the horizontal axis.
- ◆ **ylabel:** This parameter takes in a **String** and adds a label to the vertical axis.
- ◆ **color:** This parameter takes in a **String** and changes the color of the data in your graph. You can view the list of named colors [here](#).

Try experimenting with a few of these parameters to customize the look of your graph.

Extension 3: Exploring Seaborn (20-30 mins)

Pandas is a powerful tool used by programmers to search, filter, compare, modify, and visualize data. However you may have noticed that some of the graphs don't look that pretty. Allow us to introduce another Python library, [Seaborn](#). **Seaborn** is a library specifically for data visualization, which means that unlike `pandas`, most of the functions and attributes specialize in making graphs look more organized and giving programmers the ability for additional customization.

To use `seaborn` in your Kaggle notebook, we first must import the library.

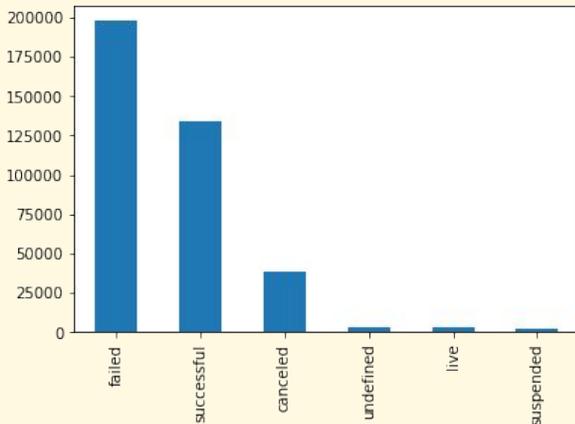
PYTHON	DESCRIPTION
<code>import seaborn as sns</code>	<ul style="list-style-type: none"> ◆ import: This keyword lets the computer know that we are using a Python library. ◆ as: This keyword gives a nickname to the package. This is an optional step but can make programming easier. ◆ seaborn/sns: This Python library is used to visualize data. We have given this package a nickname of sns.

Step 8: Extensions (cont.)

Let's compare some the bar graph we created to depict the number of projects for each state:

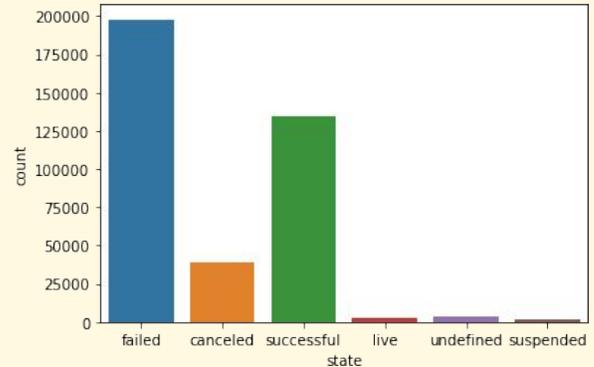
PANDAS

```
projState = ds['state'].value_counts()  
projState.plot(kind='bar')
```



SEABORN

```
sns.countplot(x='state', data=ds)
```



By using the **seaborn** library we see that each bar is colored differently, the x and y axis are already labeled, and requires less lines of code. The `countplot()` method takes in a dataset and calculates the frequency of each unique value in a feature. When using **pandas** we had to use the `value_counts()` function to do this!

Take some time to explore the [seaborn](#) library and some of the other types of graphs it offers. Here are also some additional resources if you want to learn more:

- [The Ultimate Python Seaborn Tutorial: Gotta Catch 'Em All](#)
- [Data Camp's Python Seaborn Tutorial For Beginner](#)

Step 9: Share Your Girls Who Code at Home Project! (5-10 mins)

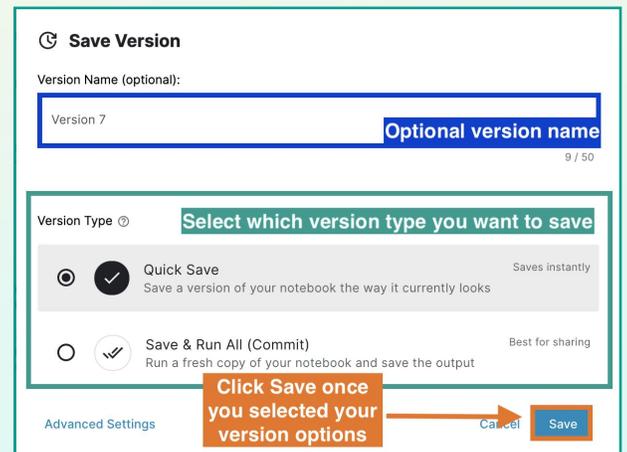
We would love to see your work and we know others would as well. Share your final project with us. Don't forget to tag [@girlswhocode](#) [#codefromhome](#) and we might even feature you on our account!

Saving Your Work (2-5 mins)

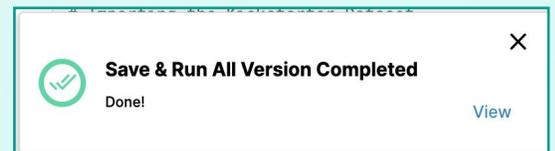
- **Click the Save Version Button.** On the top right corner of your notebook, you may have noticed the **Save Version** button. This should open a new window.
- **Add a Version Name.** This optional field is a great way for you to document what you did in this new version that may have been different than previous versions. Kaggle will automatically number your versions so it is easy to view old versions.
- **Select the Version Type.** We recommend selecting the **Save & Run All** option.



- ◆ **Quick Save:** This is a great way to save very quickly your work. This version will save everything exactly as it is displayed in your notebook. This version of save may be problematic if you made edits to your notebook but did not rerun all code blocks.
- ◆ **Save & Run All:** This saves a fresh copy of your notebook by running all code blocks and then saving this version. This is always the best way to save your notebooks if you have the time.



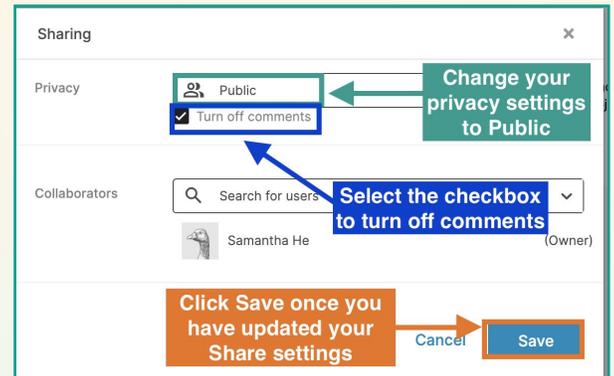
- **Click the Save button and Wait.** Once you have confirmed your save version options, click the **Save** button. You should see a pop-up window letting you know the status of your save. You might need to wait a minute for your version to be saved completely.



Step 9: Share Your Girls Who Code at Home Project (cont.)

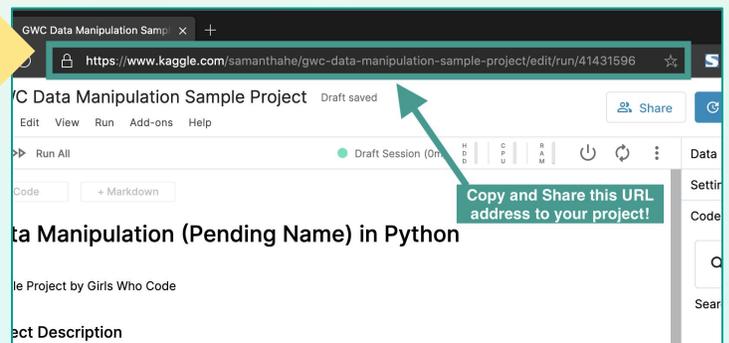
Sharing Your Work (3-5 mins)

- **Click the Share button.** On the top right of your notebook, next to the Save Version button is the Share button.
- **Change the Privacy to Public.** Select the drop down column to the right of the Privacy field and select **Public**. This will pop up a warning message make sure you are aware that other users will be able to view your project. Select **Ok, make public**.
- **Check the turn off comments box.** Click the checkbox to the left of the Turn off comments option.
- **Click Save.** Once you have confirmed that your settings are correct, click the **Save** button.
- **Share the URL address to your notebook.** Finally, just copy and paste the URL address to your project with us! Don't forget to tag [@girlswhocode](#) [#codefromhome](#).



Note about collaborators: Kaggle allows you to invite other users to program together in your notebook. This is great for working across teams. **DO NOT make anyone a collaborator!** Be selective on who you share these edit rights with.

Project Link



Stay tuned for more Girls Who Code at Home projects!

