



Activity 3.2.1

Trends in Temperature

AP CLASSROOM CONNECTION

As you progress through the Unit, additional resources are provided by the College Board through [AP Classroom](#). Your teacher will assign the topics and personal progress checks at the end of each unit.

GOALS

- Describe what information can be extracted from data.
- Identify the challenges associated with processing data.
- Extract information from data using a program.
- Select appropriate libraries or existing code segments to use in creating new programs.








What Is Big Data?

There is so much data generated every day. In private homes or public spaces, on computers or mobile devices, data is constantly being collected and used for various purposes. Examples include:

- Understanding customers and improving marketing strategies
- Enhancing day-to-day experiences, such as traffic monitoring and weather forecasting
- Helping scientists with their research, such as health and biomedical conditions, human behavior and psychology, and environment and ecology

All this data being collected contributes to what is referred to as **big data** . Big data is an evolving term that describes data sets that are too large or complex to be handled by traditional data processing software. In fact, it is common that big data be processed and analyzed using a **distributed**  network of machines that can handle the volume and complexity of the data at hand. Big data is analyzed in order to uncover patterns and trends and can sometimes be used to inform **artificial intelligence**  systems.

Note: In this lesson, you will be working with data sets that are larger than the ones you used in lesson 3.1, but are not large enough to be considered big data. They can be processed and analyzed on one machine, using a data-processing application.

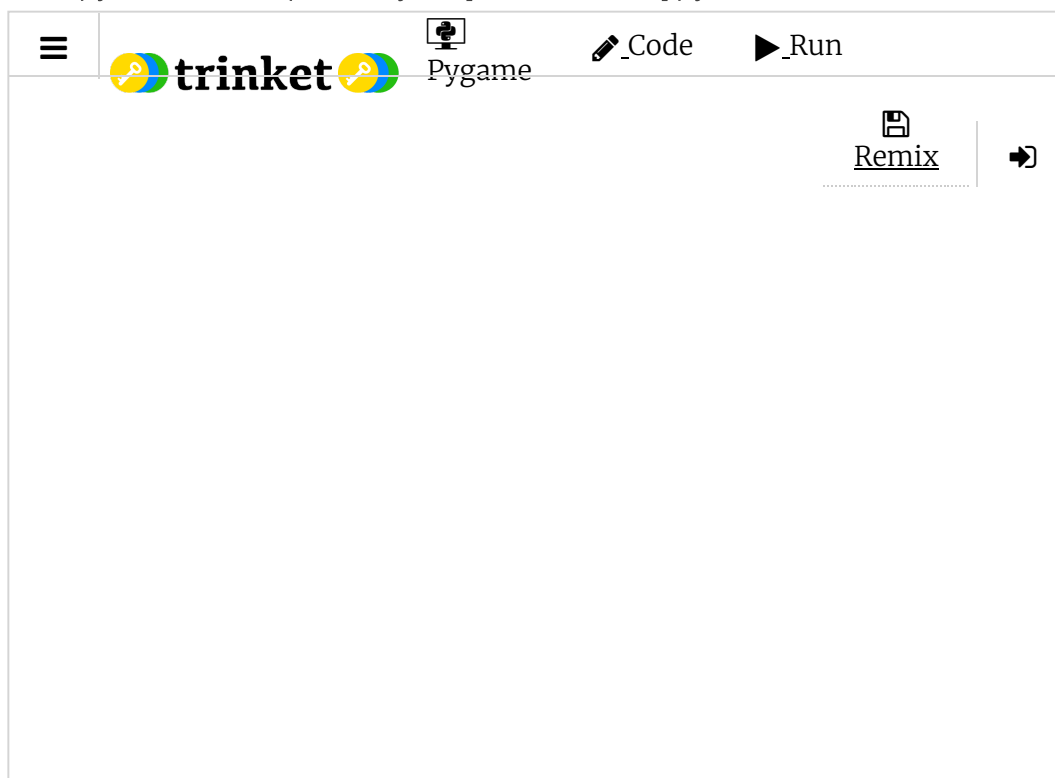
Temperature Data Collection

Climate data has been recorded for decades across the globe. From temperature readings to sea level measurements, millions of records exist to inform scientists about the earth and the environment. In this activity, you will use publicly available climate data that has been collected over multiple decades.

First, you will need to download the data file.

1

Download and extract the following program and data files into VSC, renaming *main.py* to *a321_temps_analysis_[studentinitials].py*.



2

Locate the `temperature_data.csv` file and open it to view its contents.

Note: You can use spreadsheet software, Visual Studio Code, or the code editor to view the data.

- a. The first column (Year) represents the year the data was collected.
- b. The second column (Anomaly) represents the change in global surface temperature relative to the average temperatures collected between 1951 and 1980.
- c. The third column (Lowess) stands for “LOcally WEighted Scatter-plot Smoother.” It represents the estimates that can be used to fit a smooth curve through points in a scatter plot.

Note: This data file has no incomplete records and requires no cleansing.

3

Close the file.

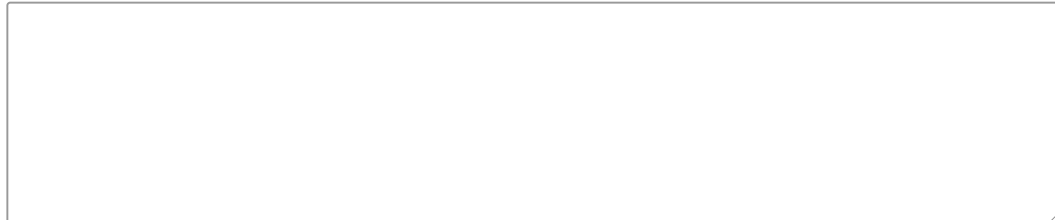
Data Analysis Using Python

So far in this unit, you have collected and analyzed data using the Vernier Graphical Analyzer and spreadsheets. Despite the many features these applications have, they still require human intervention to combine, cleanse, and visualize the data. In this activity, you will explore methods to automate the data cleansing, analysis, and visualization using *Python*®.

4

Consider the benefits of automating this process.

Why do you think automating this process can be useful?



5

Share your thoughts with your class.

6

Review the code and comments in your *temps_analysis* program.

- What are the two new modules introduced in this code and what are they used for?

Check your

- What do you think the `read_csv` function does?

Check your

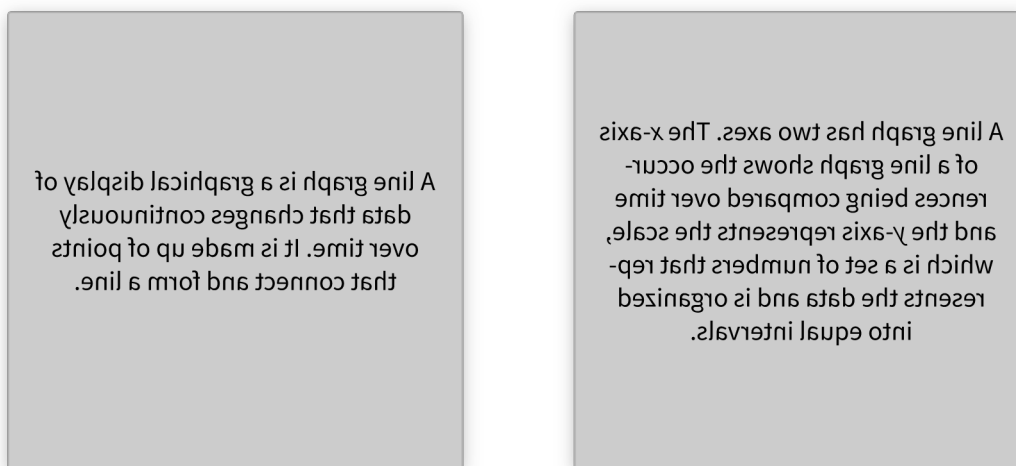
The `temp_data` variable contains the *pandas* `DataFrame` object. Both the columns and the rows start at index 0. Think of it as a table in memory:

| temp_data (DataFrame) | Column 0 'Year' | Column 1 'Anomaly' | Column 2 'LOWESS' |
|--------------------------|--------------------|-----------------------|----------------------|
| Row 0 | 1880 | -0.18 | -0.11 |
| Row 1 | 1881 | -0.1 | -0.14 |
| Row 2 | 1882 | -0.11 | -0.17 |
| Row 3 | ... | ... | ... |

- 7 If you changed the name of your data file, complete TODO #1 to update the CSV file name to your file name.
- 8 Save your work.

Line Graph

It's time to visualize the data on the screen. You will begin with a line graph.



Line graphs are very simple to plot with *matplotlib.pyplot*. Recall that you imported the module in the first line of code and gave it the alias of *plt*.

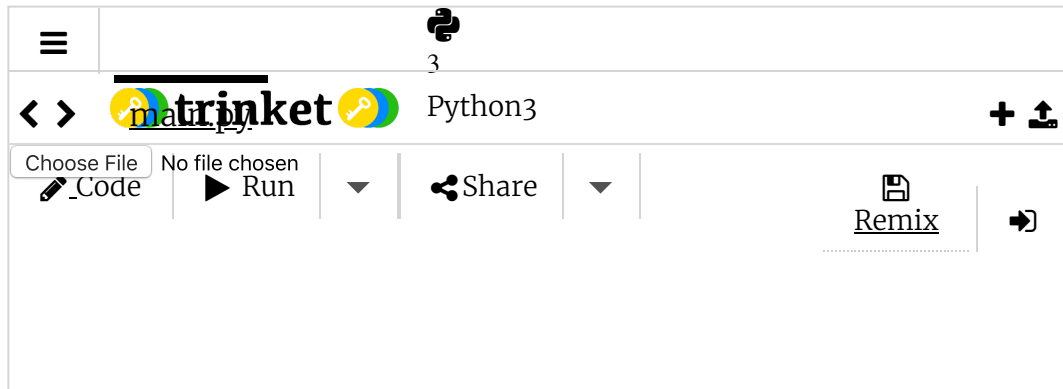
The `plot` method from `matplotlib` is what you will need to plot a line graph. This method requires at least two parameters.

1. The first is the list that makes up the x-axis values.
2. The second is the list that makes up the y-axis values.

Note: There are other options that can be specified, such as `color`, `linewidth`, and more. Some of which you will learn about in this lesson.

9

Copy the code below under `TODO #2` to create the graph including the axes labels and graph title.



10

Observe the `plot` parameters.

- a. What does the `temp_data['Year']` return?

Check your

- b. What does the `temp_data['Anomaly']` return?

Check your

- c. What do the last three lines do?

Check your

11

Test your code. Do you see the graph?

- 12 For the graph to appear on a *matplotlib* screen, you will need to add a call to `matplotlib` method `show`.
- 13 Test your code again and troubleshoot any errors. Check for any typos in your code.
- 14 Save your work.
- 15 Under TODO #3, add similar code to plot the LOWESS values in a line graph. Use another color for the line.

Check your code

- 16 Test your code and make any necessary adjustments.

Notice how the two lines are plotted on separate graphs. It is not the most effective way to compare the anomalies to the LOWESS values. It would make more sense to plot them both on the same graph for a better visual comparison.

To do so, you will only need one call to `show` after both `plot` method calls have been specified. Also since both lines will appear on the same graph, there is no need to specify the axes labels and graph title twice!

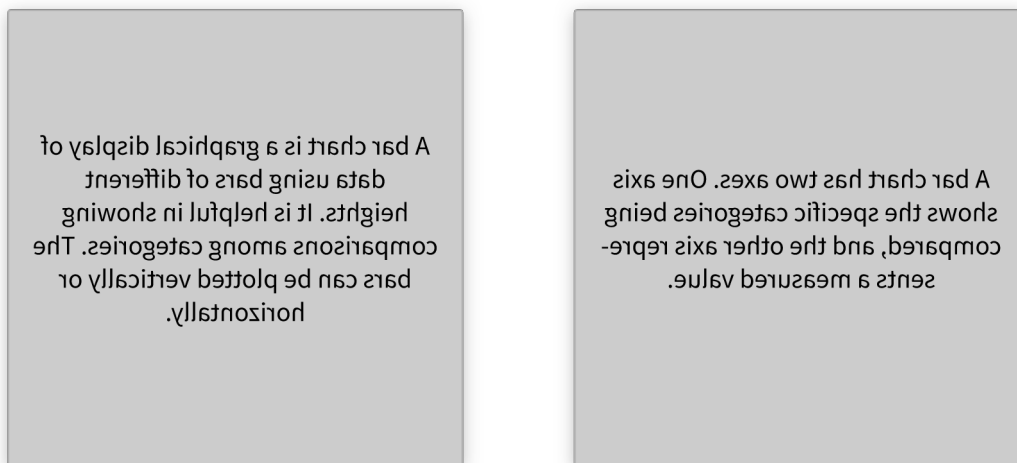
- 17 Remove the `show` method call from the TODO #2 section, and the duplicate calls for the axes labels and graph title under the TODO #3 section.

Check your code

- 18 Test your code now. You should see both lines on the same graph.
- 19 Save your work.

Bar Chart

The `matplotlib.pyplot` offers various options for plotting. The next one you will explore is the bar chart.



The `bar` method is what you will need to plot a bar chart. Like the line graph, it also requires at least two parameters.

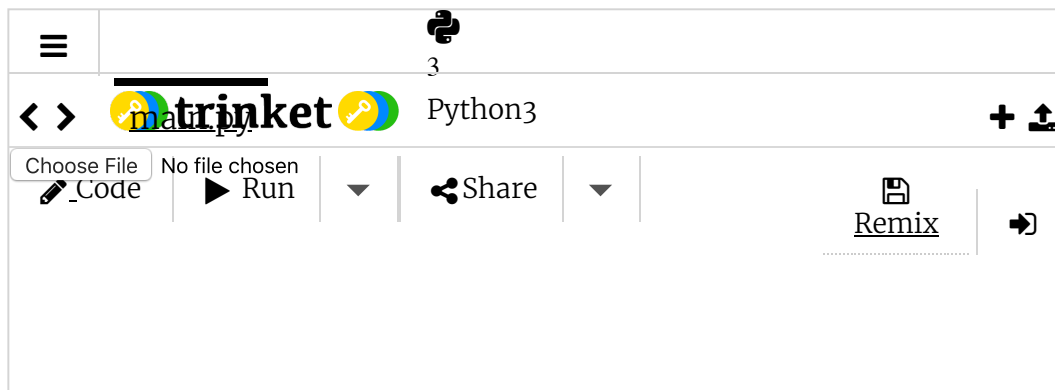
1. The first is the list of the x-axis values.
2. The second is the list of the y-axis values.

Note: There are other options that can be specified, such as `align`, `color`, and more.

You can also specify labels for the axes and a title for the graph.

20

Copy the code below under **TODO #4** to create the bar chart including the axes labels and graph title.



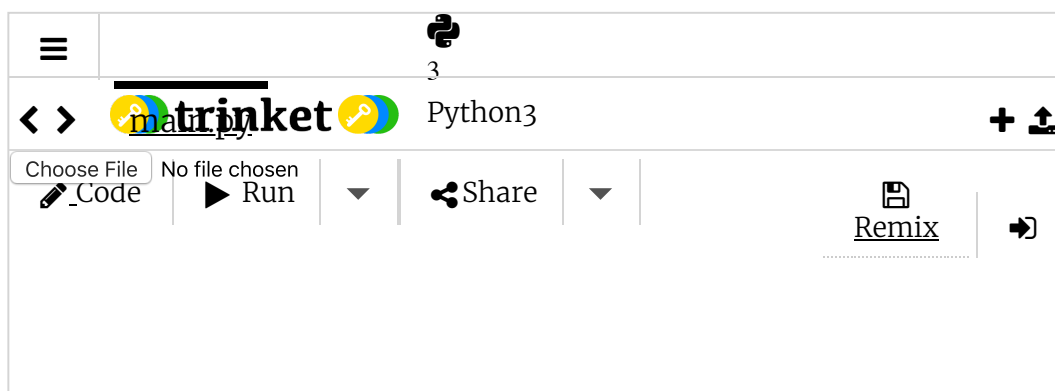
21 Test your code again and troubleshoot any errors. Check for any typos in your code.

22 Save your work.

Min, Max, and Average

When studying numeric data and patterns, it is common to look at what the minimum, maximum, and average values are over a given interval of time. In this section, you will add the code to determine those values.

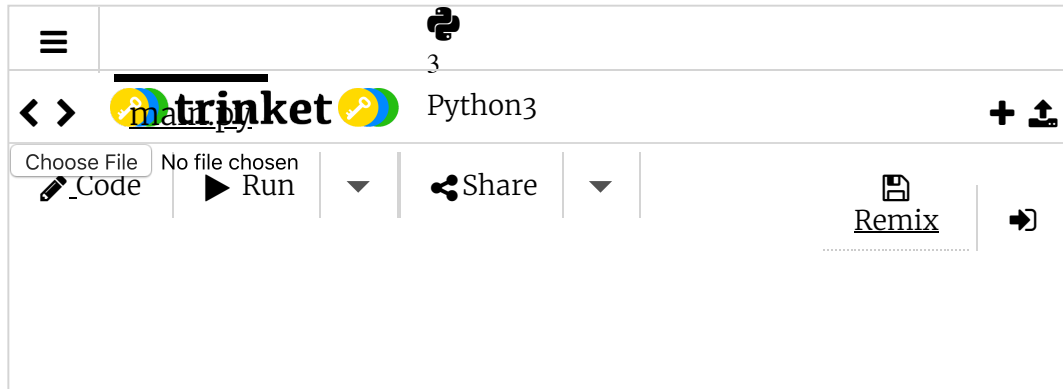
23 Start by initializing the required variables under the TODO #5.
What does this code do?



Check your response

- 24 Add the following code to loop through the anomalies and find the minimum value.

What does this code do?



[Check your response](#)

- 25 Add a similar `if` statement to the `for` loop to determine the maximum anomaly and its corresponding year.

[Check your code](#)

- 26 Add a line of code to the `for` loop to calculate the sum of the anomalies.

[Check your code](#)

- 27 After the `for` loop is complete, add a line of code to calculate the average anomaly.

[Check your code](#)

- 28 Add `print` statements to display the minimum anomaly and year, the maximum anomaly and year, and the average anomaly.

[Check your code](#)

- 29 Test your code and make sure it runs correctly.

a. What are the minimum anomaly and year?

b. What are the maximum anomaly and year?

c. What is the average anomaly?

30

Check your code and responses with an elbow partner.

31

Save your work.

Data Cleansing Using Python

The temperature data file you used in this activity was clean. There were no missing values or incomplete records, so it did not require any data cleansing. Depending on how data is collected, it may not be uniform and need to be cleaned. For example, you could collect temperature data from people across the country, and they may have entered the data with different abbreviations, spelling, or capitalization. In this section, you will learn to use *pandas* functions to help you cleanse data in *Python*.

Carbon Dioxide Analysis

Carbon dioxide in the air can increase due to human activities such as deforestation and burning fossil fuels, and natural processes such as respiration and volcanic eruptions. Let's use *Python* to clean and analyze CO₂ data collected over the last several decades.

32

Observe the contents of the `co2_data.csv` file you downloaded at the beginning of this activity.

- a. What is the unit of measure for the CO₂ average and what does it stand for?

- b. What do you think the column *decimal_year* stands for?

- c. What do the -99.99 and -1 values mean?

33

Clean the data file by removing all the informational lines (comments) so you end up with only a header row and the data rows, then save and close the file.

Note: This is a large file because it includes data per month, not just per year.

34

Create a new *Python* program `a321_co2_analysis_[studentinitials].py` in the same folder as your code and two data files.

35

Reuse the code from the temperature analysis program to create a program that reads and plots the CO₂ data.

- Read the data from the CO₂ file and store it in a `DataFrame` called `co2_data`.
- Plot the CO₂ averages as a line graph.

i. Which column should you use for the x-axis?

Need Help?

ii. Which column should you use for the y-axis?

Need Help?

36

Test your code and fix any errors.

37

Observe the line graph. It should look similar to Figure 1.

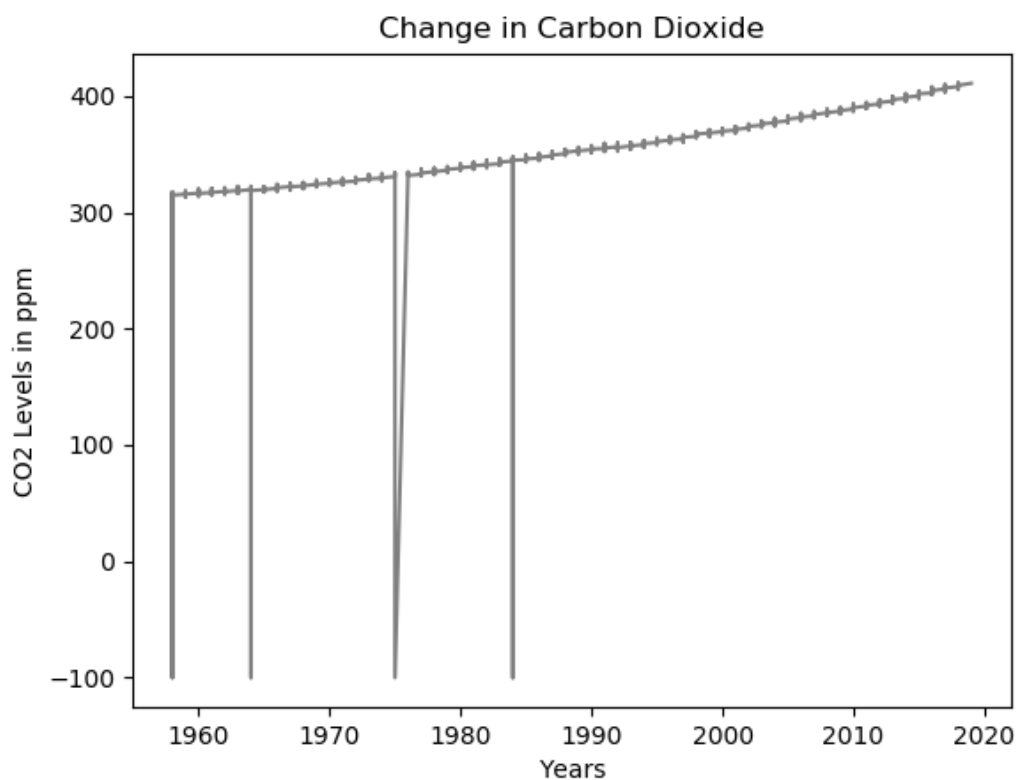


Figure 1. Example CO₂ Graph

Why does the line have those four huge dips close to -100? What should we do about them?

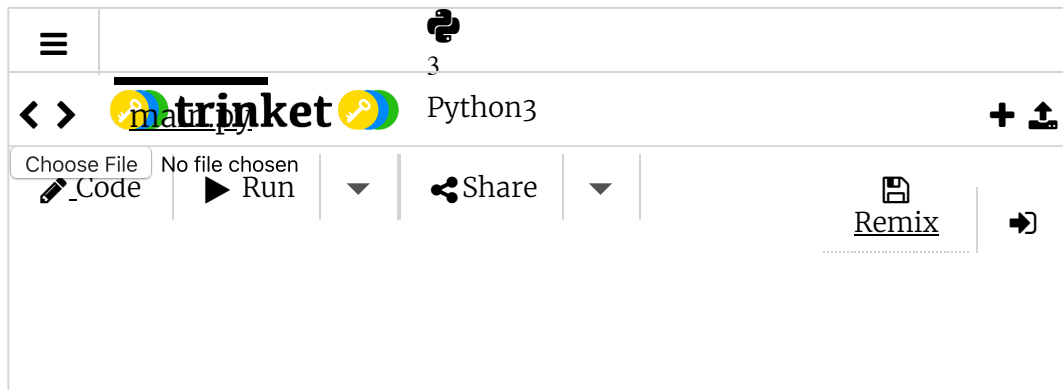
There are different ways to clean data files. Instead of manually deleting out the rows you don't need from the CSV file, you will update your *Python* code to remove those rows from the `DataFrame` object.

38

Add a line of code with the existing `import` statements to import the `math` module.

39

Insert the following code segment right after you load the data into the `DataFrame` object, `co2_data`.



a. What is the first line of code accomplishing?

Check your

b. What about the second line of code?

Check your

There are different approaches for removing rows from a data frame. Replacing incorrect values with a `Not a Number` object and calling the `dropna` function is one approach.

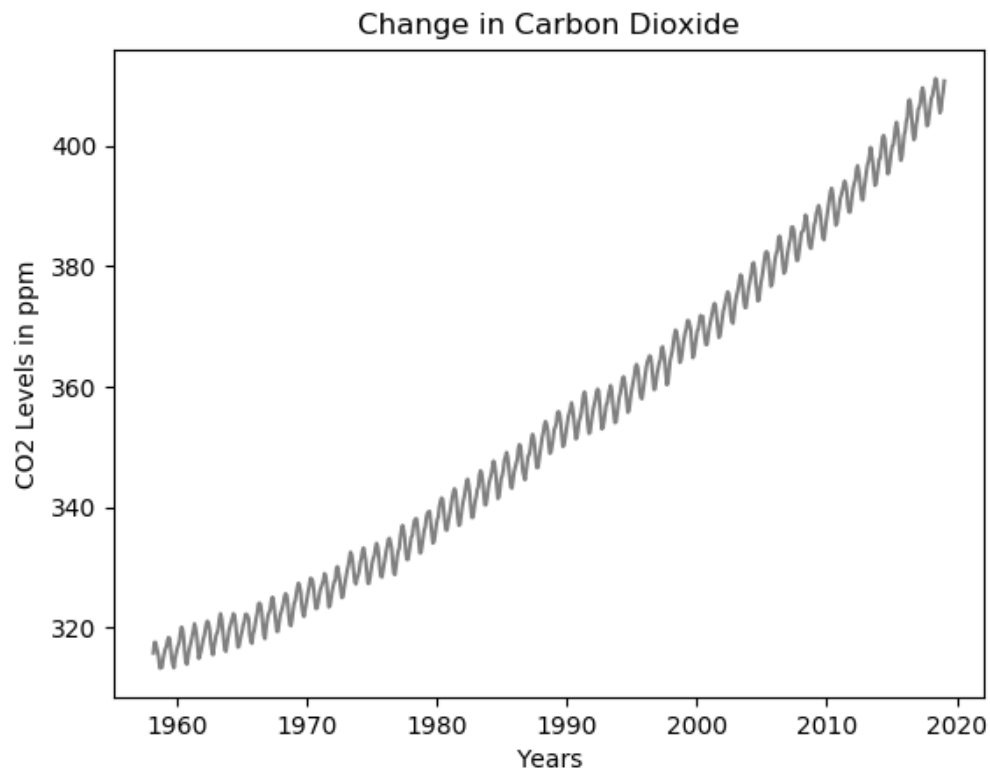
40

Add three `print(co2_data)` lines as follows:

- One after the `read_csv` call, to view the `DataFrame` contents before any cleansing is done.
- One after the `replace` call, to view the change in values.
- One after the `dropna` call, to view the effect of dropping the rows.

41

Test your code and make sure the negative values are no longer plotting.

Figure 2. CO₂ Graph after Data Cleansing

42

Observe the data printed in the *Python* terminal, taking a closer look at the first ten rows.

- The raw data from the first print includes -99.99.

| | Year | Month | decimal_year | Average | #Days |
|----|------|-------|--------------|---------|-------|
| 0 | 1958 | 3 | 1958.208 | 315.71 | -1 |
| 1 | 1958 | 4 | 1958.292 | 317.45 | -1 |
| 2 | 1958 | 5 | 1958.375 | 317.50 | -1 |
| 3 | 1958 | 6 | 1958.458 | -99.99 | -1 |
| 4 | 1958 | 7 | 1958.542 | 315.86 | -1 |
| 5 | 1958 | 8 | 1958.625 | 314.93 | -1 |
| 6 | 1958 | 9 | 1958.708 | 313.20 | -1 |
| 7 | 1958 | 10 | 1958.792 | -99.99 | -1 |
| 8 | 1958 | 11 | 1958.875 | 313.33 | -1 |
| 9 | 1958 | 12 | 1958.958 | 314.67 | -1 |
| 10 | 1959 | 1 | 1959.042 | 315.62 | -1 |

Figure 3. CO₂ data highlighting the -99.99 values

- The data after the `replace()` function shows the NaN values.

```
[731 rows x 5 columns]
```

| | Year | Month | decimal_year | Average | #Days |
|----|------|-------|--------------|---------|-------|
| 0 | 1958 | 3 | 1958.208 | 315.71 | -1 |
| 1 | 1958 | 4 | 1958.292 | 317.45 | -1 |
| 2 | 1958 | 5 | 1958.375 | 317.50 | -1 |
| 3 | 1958 | 6 | 1958.458 | NaN | -1 |
| 4 | 1958 | 7 | 1958.542 | 315.86 | -1 |
| 5 | 1958 | 8 | 1958.625 | 314.93 | -1 |
| 6 | 1958 | 9 | 1958.708 | 313.20 | -1 |
| 7 | 1958 | 10 | 1958.792 | NaN | -1 |
| 8 | 1958 | 11 | 1958.875 | 313.33 | -1 |
| 9 | 1958 | 12 | 1958.958 | 314.67 | -1 |
| 10 | 1959 | 1 | 1959.042 | 315.62 | -1 |

Figure 4. CO₂ data after using the function replaced all -99.99 values with NaN

- The data after the `dropna()` function shows that rows 3 and 7 have been dropped.

```
[731 rows x 5 columns]
```

| | Year | Month | decimal_year | Average | #Days |
|----|------|-------|--------------|---------|-------|
| 0 | 1958 | 3 | 1958.208 | 315.71 | -1 |
| 1 | 1958 | 4 | 1958.292 | 317.45 | -1 |
| 2 | 1958 | 5 | 1958.375 | 317.50 | -1 |
| 4 | 1958 | 7 | 1958.542 | 315.86 | -1 |
| 5 | 1958 | 8 | 1958.625 | 314.93 | -1 |
| 6 | 1958 | 9 | 1958.708 | 313.20 | -1 |
| 8 | 1958 | 11 | 1958.875 | 313.33 | -1 |
| 9 | 1958 | 12 | 1958.958 | 314.67 | -1 |
| 10 | 1959 | 1 | 1959.042 | 315.62 | -1 |

Figure 5. CO₂ data highlighting the removal of the NaN values

**PLTW COMPUTER SCIENCE NOTEBOOK**

Document with pseudocode the data-cleansing algorithm you learned today and include a brief description of the cleaning effect.

Congratulations! Your data analysis is complete!

What Does This All Mean?

43

Access [NASA's Climate Site](#). This should display a menu of options on the bottom of the screen. Each option represents what the site refers to as a *vital sign of the planet*.

44

Click the **Global Temperature** menu option at the bottom of the screen to view the Global Temperature vital sign page.

a. How does their graph compare to the ones you generated in *Python*?

b. Which year was the warmest?

45

Use the visualization on the right to view the change in global surface temperatures over the years.

46

Click on the four other vital signs to view their pages and tinker with the graphs and interactive simulations.



Reflection Questions: What conclusions can you draw about global temperatures, carbon dioxide amounts, and the climate in general based on the information you just obtained? How does the data change in earlier years compare to the change in more recent years? What could be causing this change?

FEEDBACK

We hope you enjoyed this activity! Please provide feedback about your experiences.



Tell us what you think!

CONCLUSION

- 1 Read the [following article](#) on the NASA climate site, and summarize two ways that climate change is impacting the world we live in.
- 2 Compare and contrast the following types of graphs:
 - Line graph
 - Bar chart
 - Pie chart

Include in your analysis:

- a. Best use case(s) for the graph.
- b. The story or message the graph tells.

Proceed to next activity