Watch by Tuesday, October 27, 2020 | **Lesson #7**

Loading files (Colab, Google Drive), loading data (readlines, numpy), and an intro to plotting (matplotlib)

OCEAN 215 | Autumn 2020

Ethan Campbell and **Katy Christensen**

What we’ll cover in this lesson

1. Loading and saving files to Google Colab
2. Loading data using readlines and numpy
3. Intro to plotting

What we’ll cover in this lesson

1. **Loading and saving files to Google Colab**

2. Loading data using readlines and numpy

3. Intro to plotting

Real data

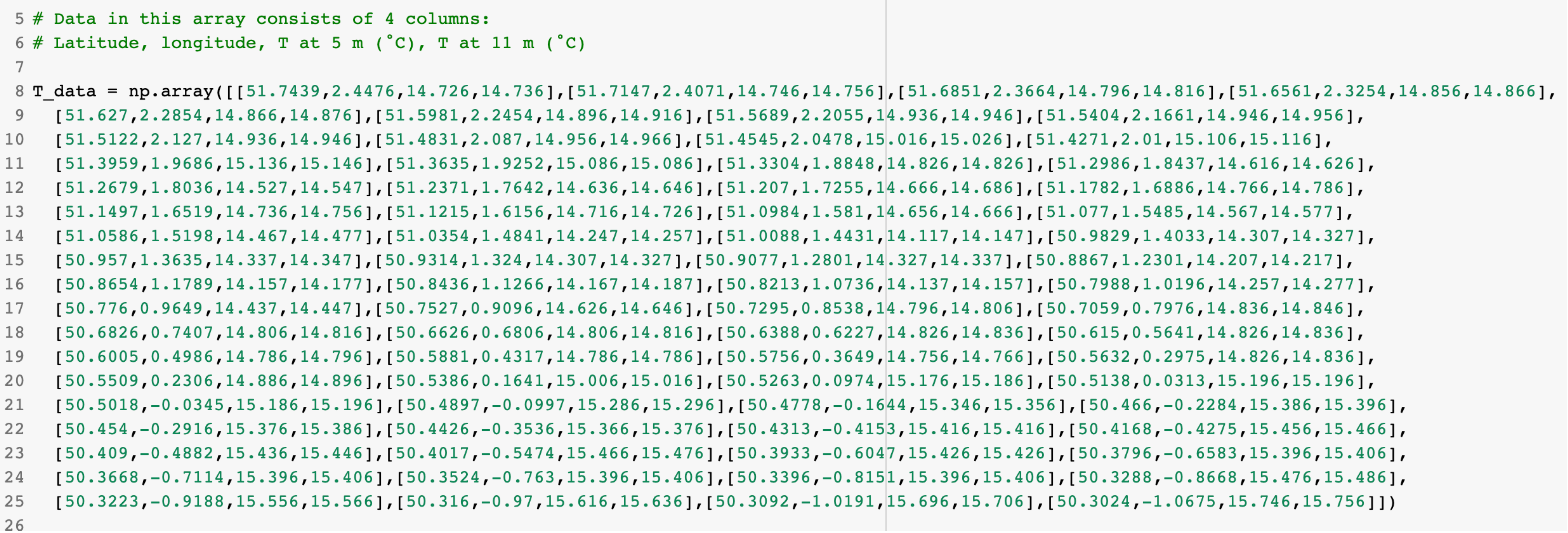
We could keep creating simple arrays…

```
np.array([[1,2,3,4],[5,6,7,8]])
```

But looking at real data is usually more interesting!
(and kind of the point of data science)

Using real data means having data files

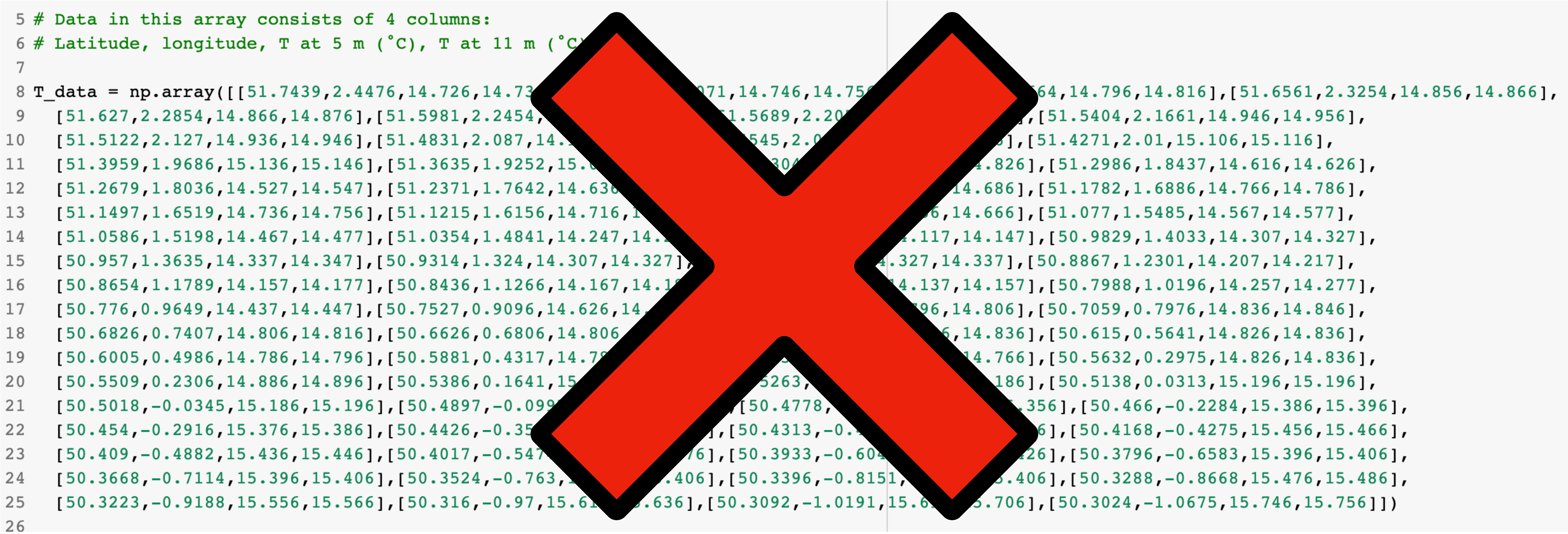
Assignment #2, Q4 - data numpy array



```
# Data in this array consists of 4 columns:
# Latitude, longitude, T at 5 m (˚C), T at 11 m (˚C)

T_data = np.array([[51.7439,2.4476,14.726,14.736],[51.7147,2.4071,14.746,14.756],[51.6851,2.3664,14.796,14.816],[51.6561,2.3254,14.856,14.866],
  [51.627,2.2854,14.866,14.876],[51.5981,2.2454,14.896,14.916],[51.5689,2.2055,14.936,14.946],[51.5404,2.1661,14.946,14.956],
  [51.5122,2.127,14.936,14.946],[51.4831,2.087,14.956,14.966],[51.4545,2.0478,15.016,15.026],[51.4271,2.01,15.106,15.116],
  [51.3959,1.9686,15.136,15.146],[51.3635,1.9252,15.086,15.086],[51.3304,1.8848,14.826,14.826],[51.2986,1.8437,14.616,14.626],
  [51.2679,1.8036,14.527,14.547],[51.2371,1.7642,14.636,14.646],[51.207,1.7255,14.666,14.686],[51.1782,1.6886,14.766,14.786],
  [51.1497,1.6519,14.736,14.756],[51.1215,1.6156,14.716,14.726],[51.0984,1.581,14.656,14.666],[51.077,1.5485,14.567,14.577],
  [51.0586,1.5198,14.467,14.477],[51.0354,1.4841,14.247,14.257],[51.0088,1.4431,14.117,14.147],[50.9829,1.4033,14.307,14.327],
  [50.957,1.3635,14.337,14.347],[50.9314,1.324,14.307,14.327],[50.9077,1.2801,14.327,14.337],[50.8867,1.2301,14.207,14.217],
  [50.8654,1.1789,14.157,14.177],[50.8436,1.1266,14.167,14.187],[50.8213,1.0736,14.137,14.157],[50.7988,1.0196,14.257,14.277],
  [50.776,0.9649,14.437,14.447],[50.7527,0.9096,14.626,14.646],[50.7295,0.8538,14.796,14.806],[50.7059,0.7976,14.836,14.846],
  [50.6826,0.7407,14.806,14.816],[50.6626,0.6806,14.806,14.816],[50.6388,0.6227,14.826,14.836],[50.615,0.5641,14.826,14.836],
  [50.6005,0.4986,14.786,14.796],[50.5881,0.4317,14.786,14.786],[50.5756,0.3649,14.756,14.766],[50.5632,0.2975,14.826,14.836],
  [50.5509,0.2306,14.886,14.896],[50.5386,0.1641,15.006,15.016],[50.5263,0.0974,15.176,15.186],[50.5138,0.0313,15.196,15.196],
  [50.5018,-0.0345,15.186,15.196],[50.4897,-0.0997,15.286,15.296],[50.4778,-0.1644,15.346,15.356],[50.466,-0.2284,15.386,15.396],
  [50.454,-0.2916,15.376,15.386],[50.4426,-0.3536,15.366,15.376],[50.4313,-0.4153,15.416,15.416],[50.4168,-0.4275,15.456,15.466],
  [50.409,-0.4882,15.436,15.446],[50.4017,-0.5474,15.466,15.476],[50.3933,-0.6047,15.426,15.426],[50.3796,-0.6583,15.396,15.406],
  [50.3668,-0.7114,15.396,15.406],[50.3524,-0.763,15.396,15.406],[50.3396,-0.8151,15.396,15.406],[50.3288,-0.8668,15.476,15.486],
  [50.3223,-0.9188,15.556,15.566],[50.316,-0.97,15.616,15.636],[50.3092,-1.0191,15.696,15.706],[50.3024,-1.0675,15.746,15.756]])
```

Using real data means having data files

Assignment #2, Q4 - data numpy array



```
# Data in this array consists of 4 columns:
# Latitude, longitude, T at 5 m (˚C), T at 11 m (˚C

T_data = np.array([[51.7439,2.4476,14.726,14.73 71,14.746,14.75 64,14.796,14.816],[51.6561,2.3254,14.856,14.866],
  [51.627,2.2854,14.866,14.876],[51.5981,2.2454, 1.5689,2.20 ],[51.5404,2.1661,14.946,14.956],
  [51.5122,2.127,14.936,14.946],[51.4831,2.087,14. 545,2.0 ],[51.4271,2.01,15.106,15.116],
  [51.3959,1.9686,15.136,15.146],[51.3635,1.9252,15.0 304 .826],[51.2986,1.8437,14.616,14.626],
  [51.2679,1.8036,14.527,14.547],[51.2371,1.7642,14.636 14.686],[51.1782,1.6886,14.766,14.786],
  [51.1497,1.6519,14.736,14.756],[51.1215,1.6156,14.716,1 6,14.666],[51.077,1.5485,14.567,14.577],
  [51.0586,1.5198,14.467,14.477],[51.0354,1.4841,14.247,14. .117,14.147],[50.9829,1.4033,14.307,14.327],
  [50.957,1.3635,14.337,14.347],[50.9314,1.324,14.307,14.327], .327,14.337],[50.8867,1.2301,14.207,14.217],
  [50.8654,1.1789,14.157,14.177],[50.8436,1.1266,14.167,14.1 4.137,14.157],[50.7988,1.0196,14.257,14.277],
  [50.776,0.9649,14.437,14.447],[50.7527,0.9096,14.626,14. 96,14.806],[50.7059,0.7976,14.836,14.846],
  [50.6826,0.7407,14.806,14.816],[50.6626,0.6806,14.806 6,14.836],[50.615,0.5641,14.826,14.836],
  [50.6005,0.4986,14.786,14.796],[50.5881,0.4317,14.7 14.766],[50.5632,0.2975,14.826,14.836],
  [50.5509,0.2306,14.886,14.896],[50.5386,0.1641,15 5263, 186],[50.5138,0.0313,15.196,15.196],
  [50.5018,-0.0345,15.186,15.196],[50.4897,-0.099 ],[50.4778, .356],[50.466,-0.2284,15.386,15.396],
  [50.454,-0.2916,15.376,15.386],[50.4426,-0.35 ],[50.4313,-0. 6],[50.4168,-0.4275,15.456,15.466],
  [50.409,-0.4882,15.436,15.446],[50.4017,-0.547 76],[50.3933,-0.604 26],[50.3796,-0.6583,15.396,15.406],
  [50.3668,-0.7114,15.396,15.406],[50.3524,-0.763,1 .406],[50.3396,-0.8151, .406],[50.3288,-0.8668,15.476,15.486],
  [50.3223,-0.9188,15.556,15.566],[50.316,-0.97,15.61 .636],[50.3092,-1.0191,15.6 5.706],[50.3024,-1.0675,15.746,15.756]])
```

Instead of having the data hard-coded into your notebooks, we will now learn how to read data files

Using real data means having data files

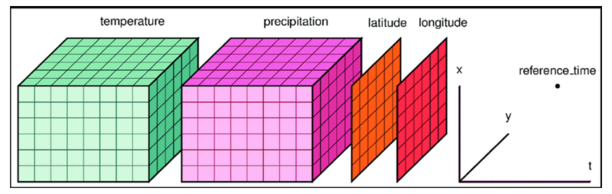
Most common data file types

Covered in this class

.txt (ASCII text)

.csv (comma separated values)

.xlsx (Microsoft Excel)



.nc (NetCDF)

Not covered in this class (probably)

.json (JavaScript object notation)

.jpg (JPEG)

.avi (audio-visual interleave)

Using data files in Colab notebooks

Google Colab runs on the Cloud so files that are stored on your computer (locally) are not accessible. There are options for loading data files:

1) Upload local files to a runtime

Pros:

- Can keep your files offline/doesn't take space on Google drive
- Is good for a fast look at a file to see what is in it

Cons:

- Removes access files after your runtime is over (sometimes)
- Manually uploading files every time you re-open the notebook can take a lot of time

2) Mount your Google Drive

Pros:

- Your data files are accessible from any machine, every time you open the notebook because the are on Drive
- Is good for sharing data and code with others

Cons:

- Have to upload files to Cloud and navigate Google Drive file structure
- Requires internet to even look at the data

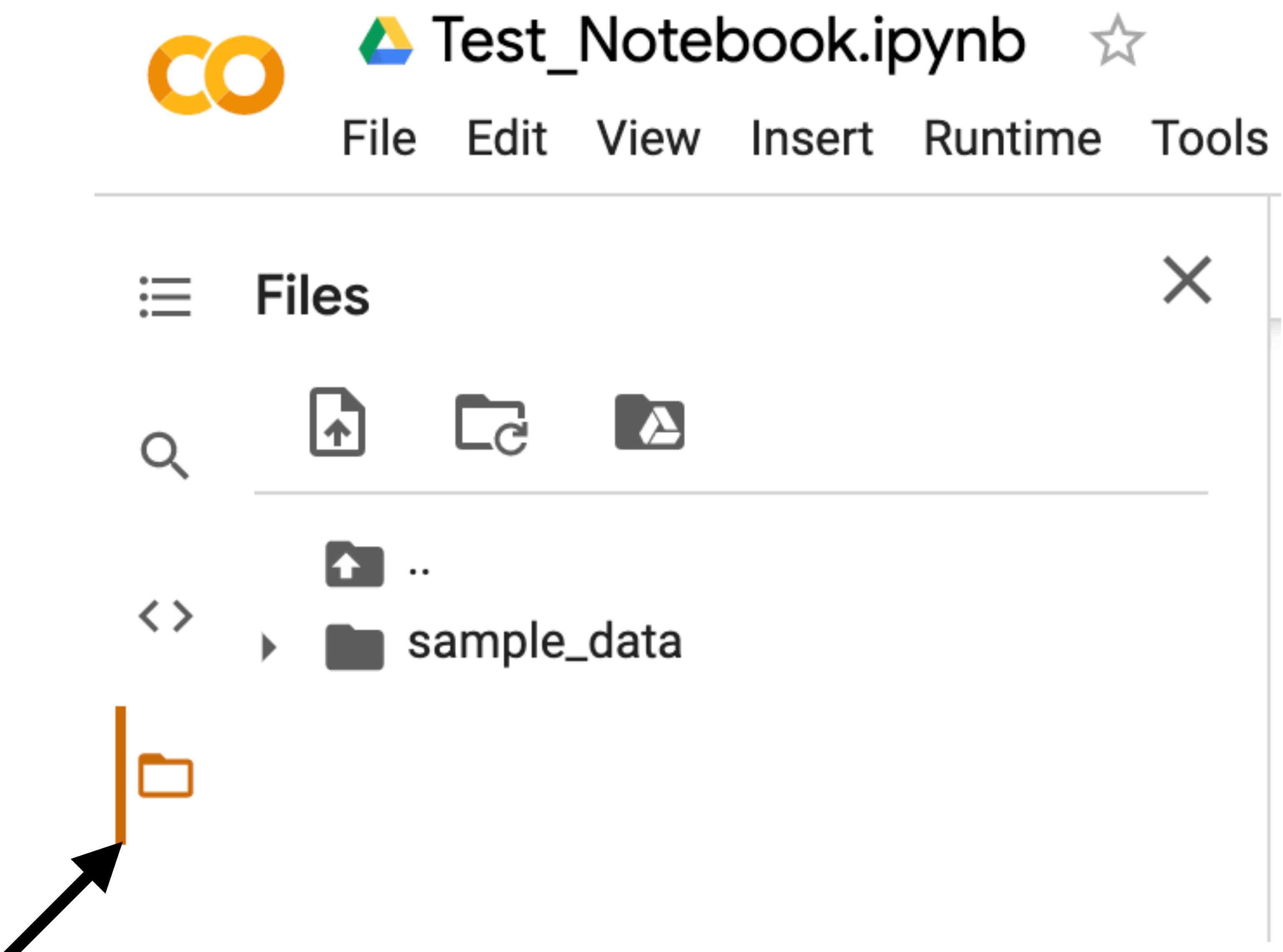
Uploading local files every runtime

User Interface (UI)

In coding cells

Uploading local files every runtime

User Interface (UI)



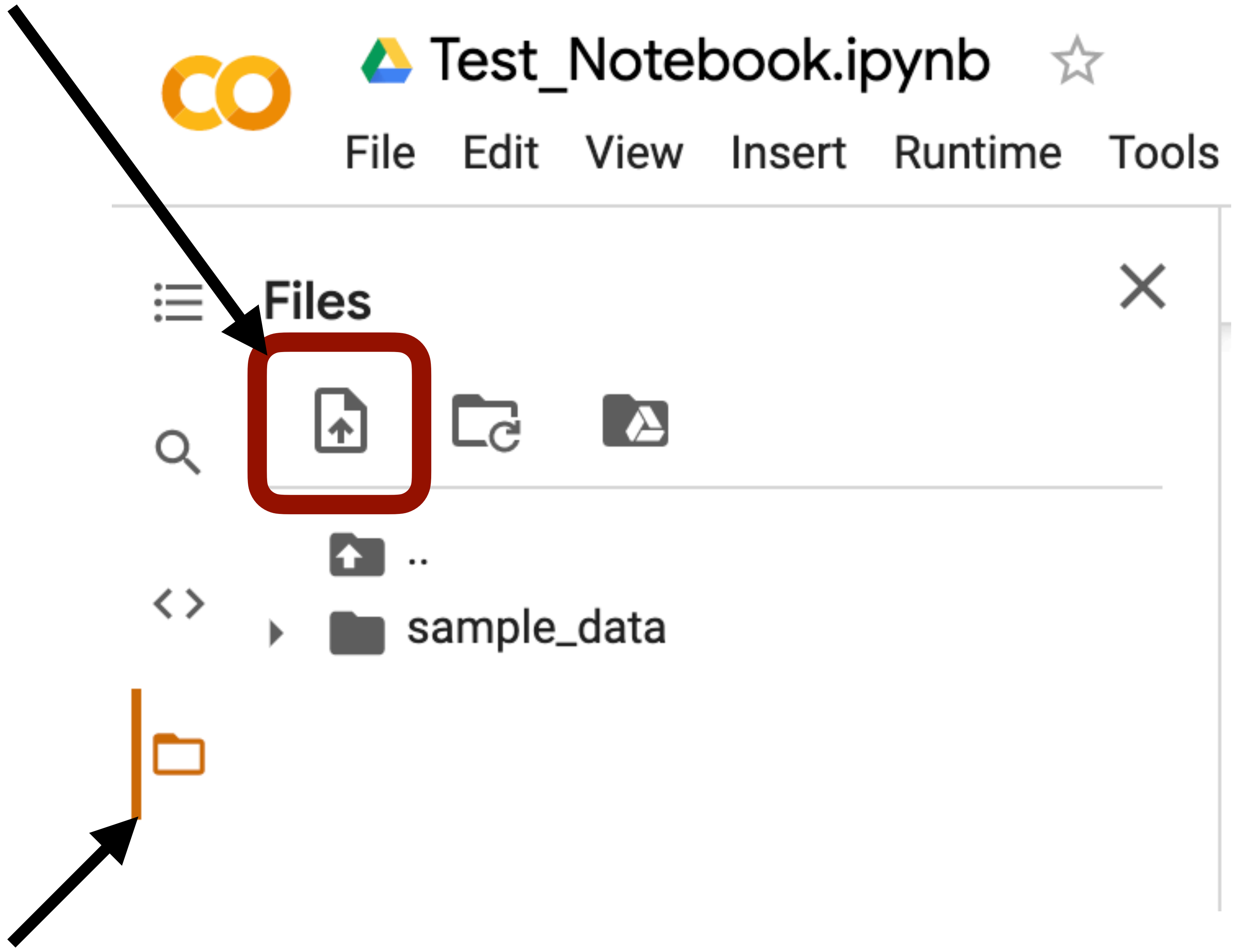
This is the sidebar menu for managing files

In coding cells

Uploading local files every runtime

User Interface (UI)

Click here and select the file
(or files, using ctrl/⌘ + click)



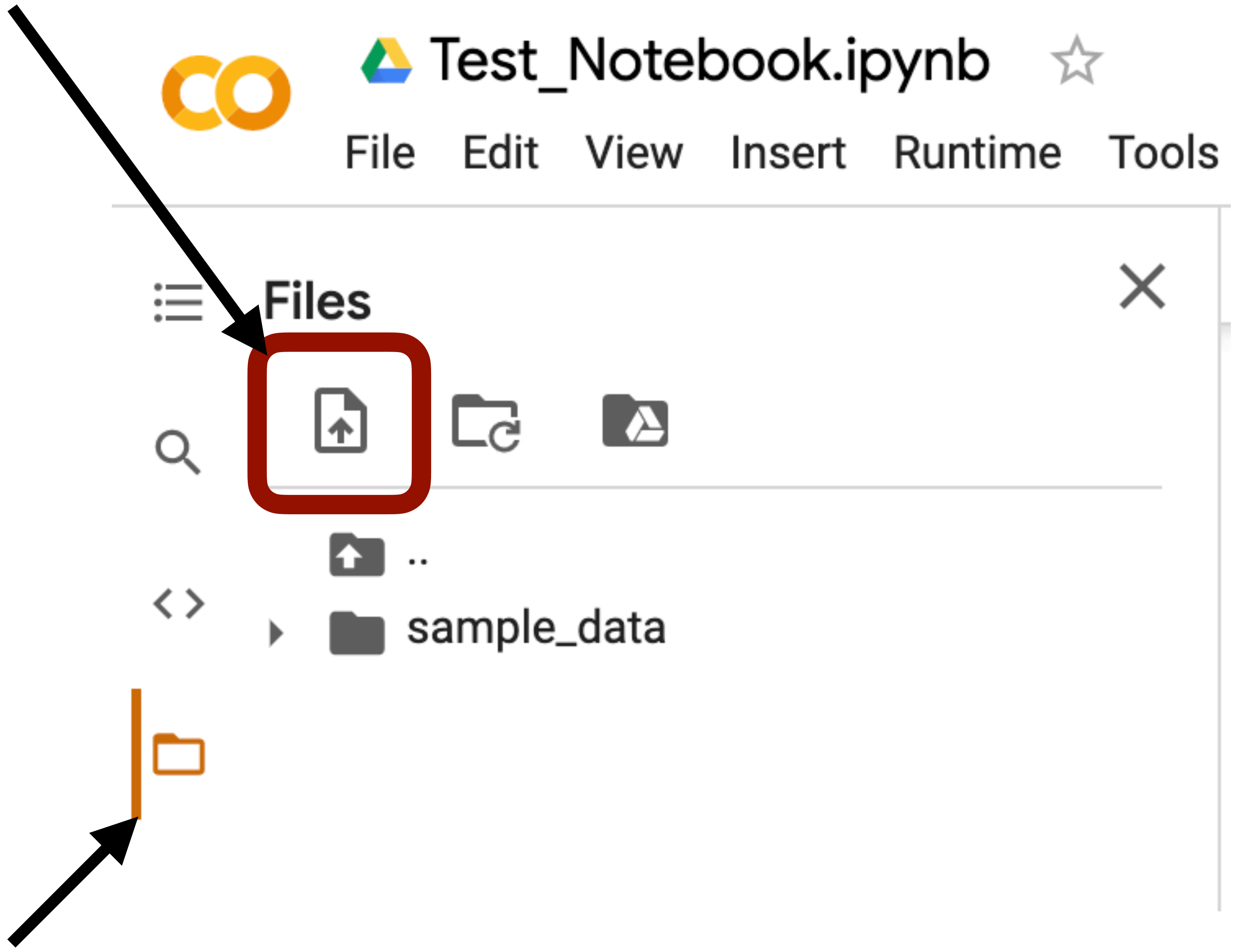
This is the sidebar menu
for managing files

In coding cells

Uploading local files every runtime

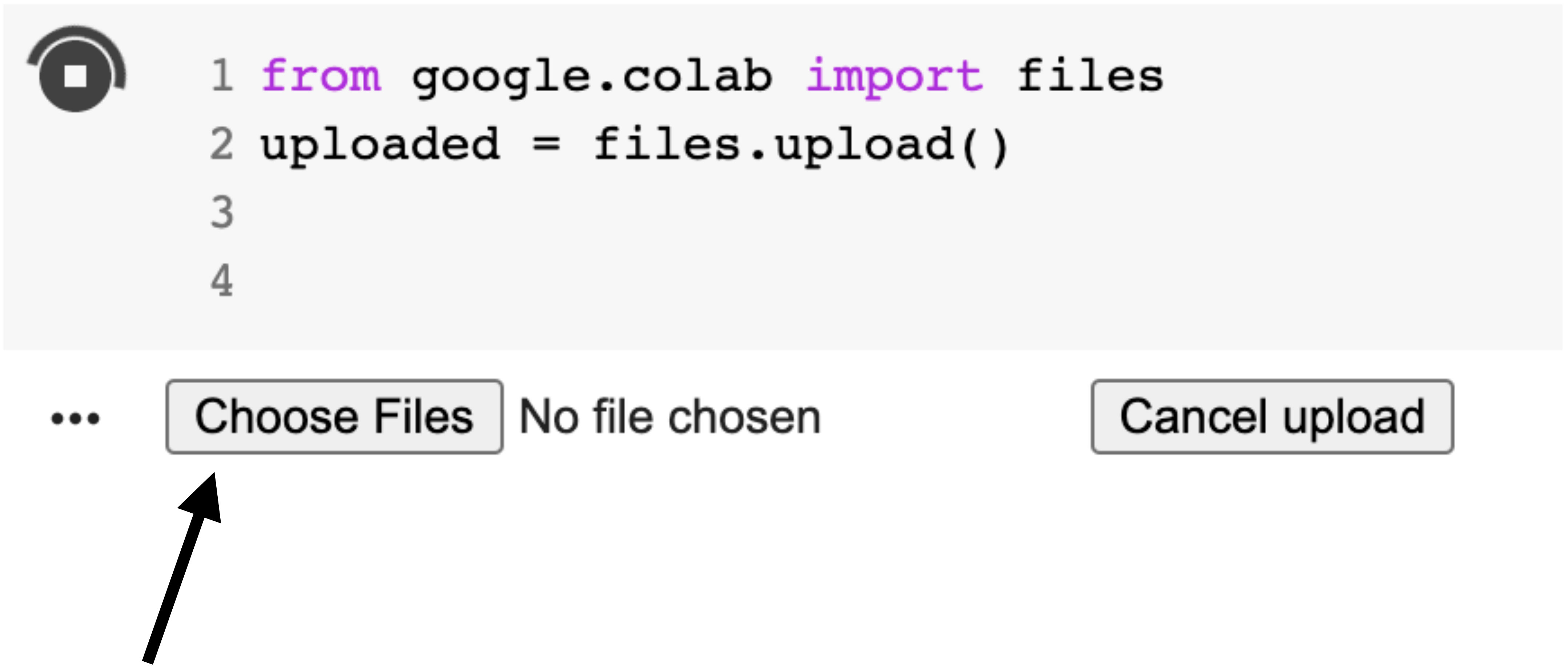
User Interface (UI)

Click here and select the file
(or files, using ctrl/⌘ + click)



This is the sidebar menu
for managing files

In coding cells



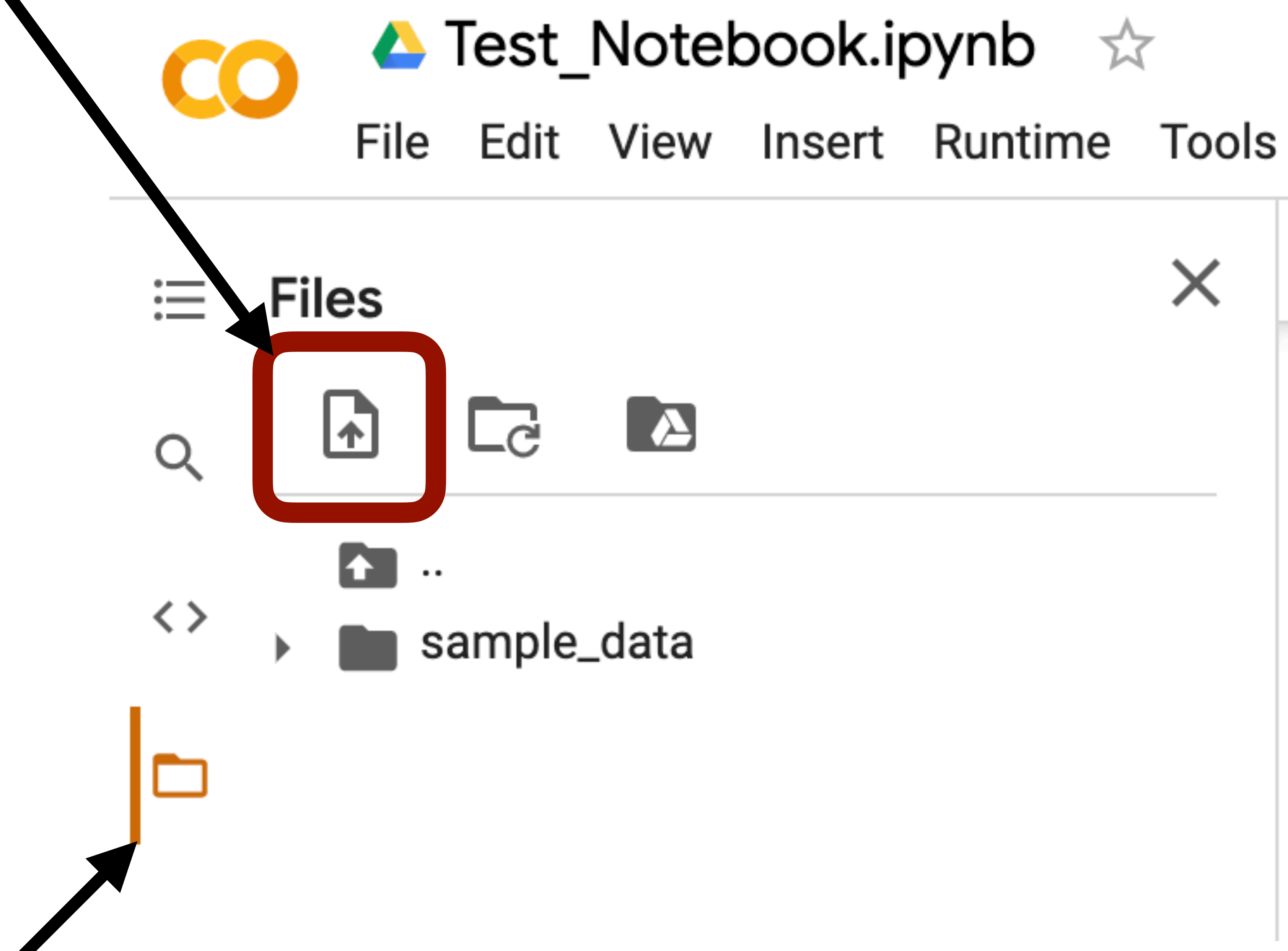
```
from google.colab import files
uploaded = files.upload()


```

Click here and select the files
(or files, using ctrl/⌘ + click)

Uploading local files every runtime

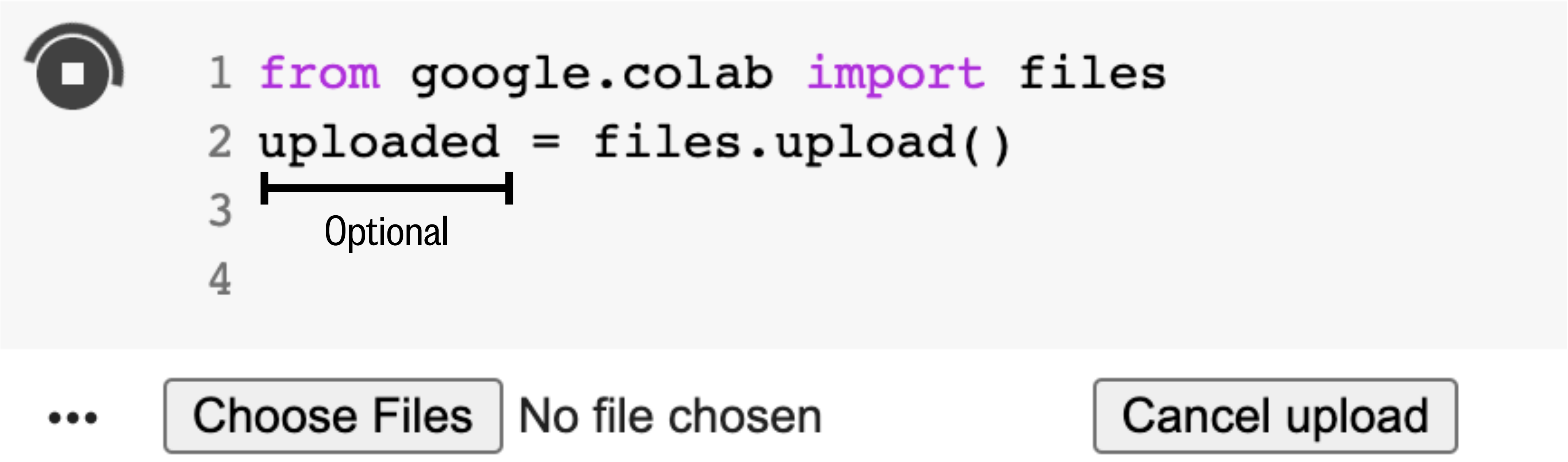
User Interface (UI)

Click here and select the file (or files, using ctrl/⌘ + click)



This is the sidebar menu for managing files

In coding cells



```
from google.colab import files
uploaded = files.upload()


```

Optional

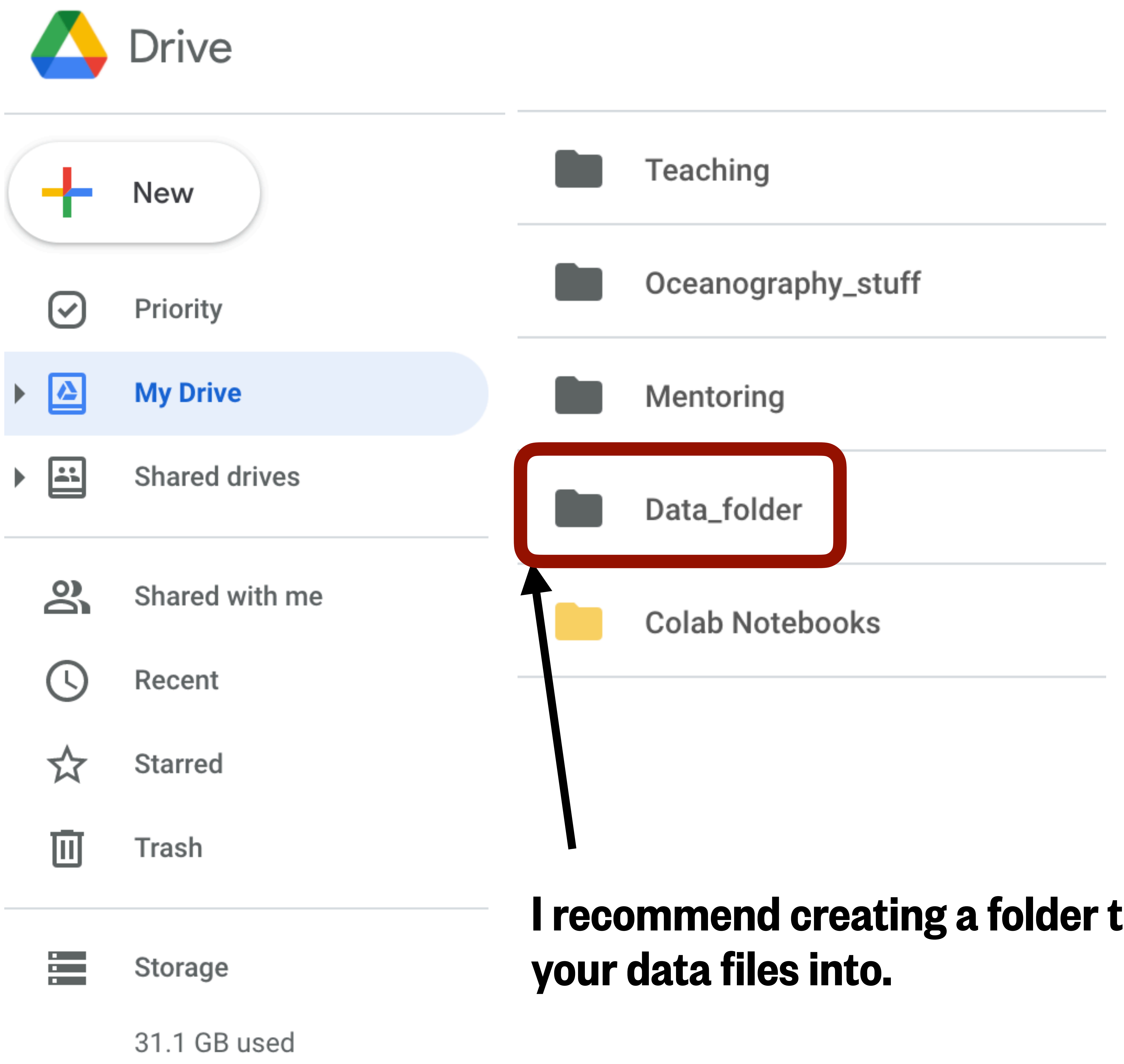
Click here and select the files (or files, using ctrl/⌘ + click)

The output of this is a Python dictionary, with each file name as a key and the file contents as its corresponding value.

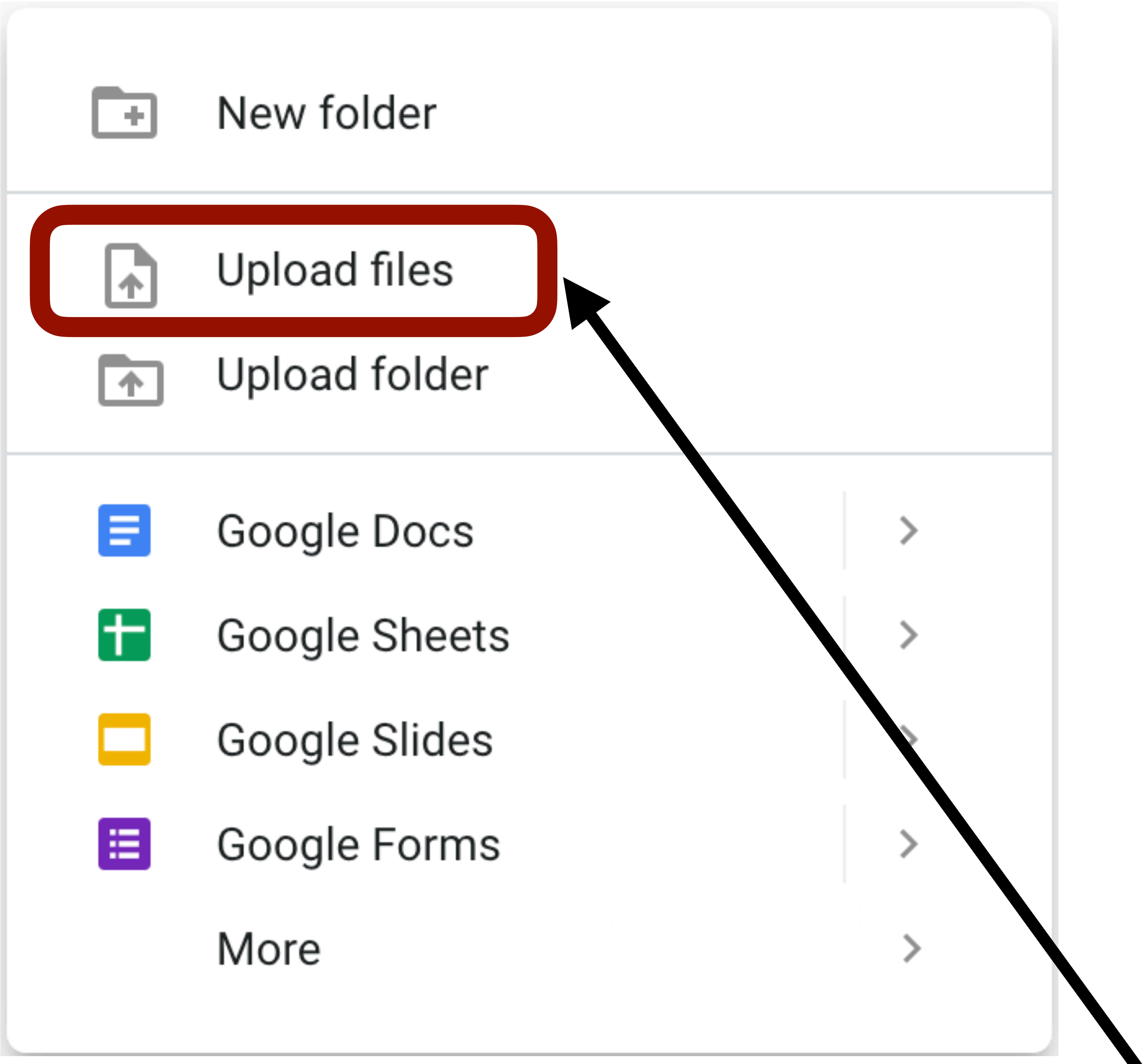
Both of these options require you to manually select the files!

Using Google Drive - uploading your files

drive.google.com



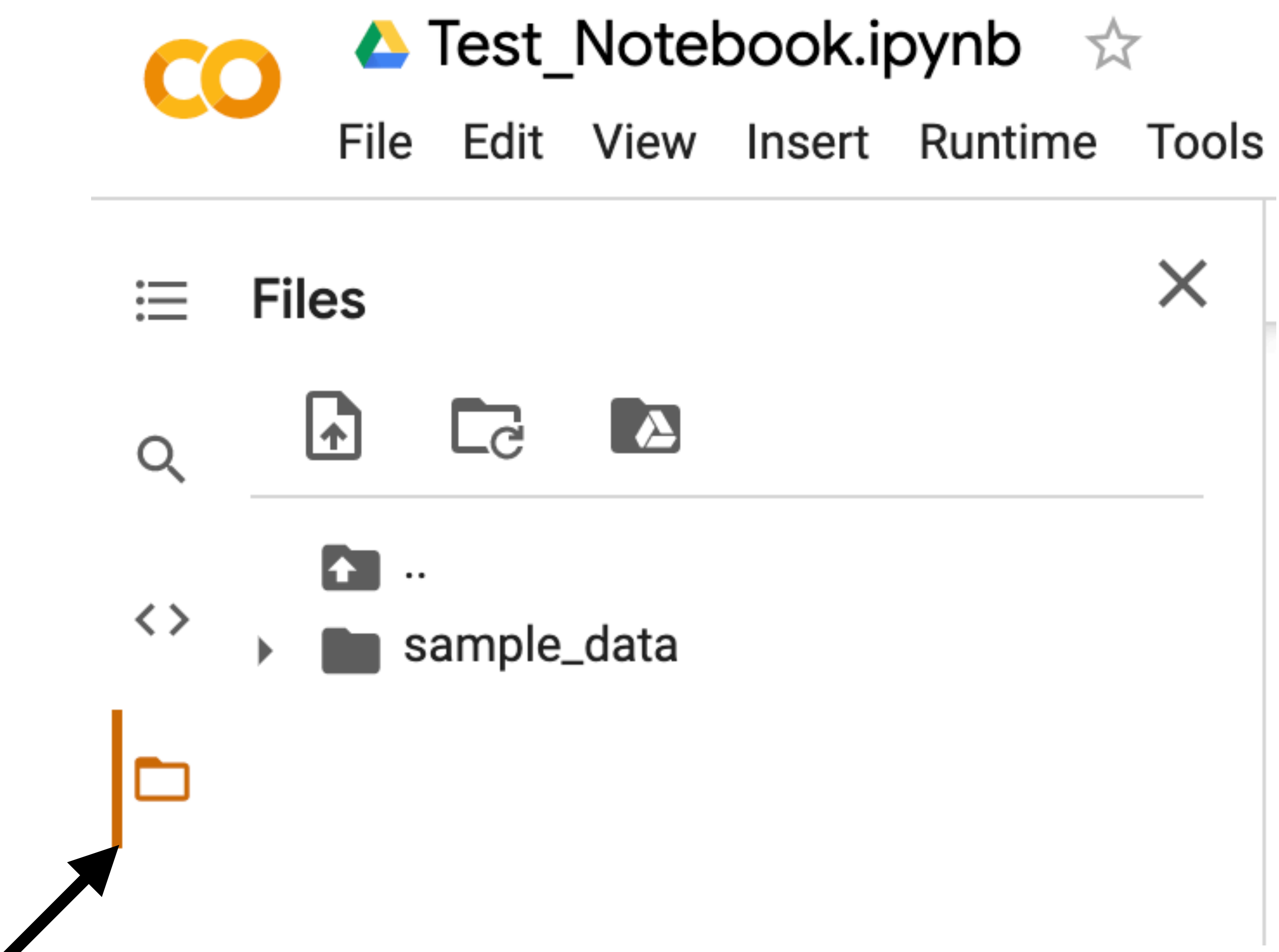
Right click to get this menu



I recommend creating a folder to put your data files into.

Click here and select the file (or files, using ctrl/⌘ + click)

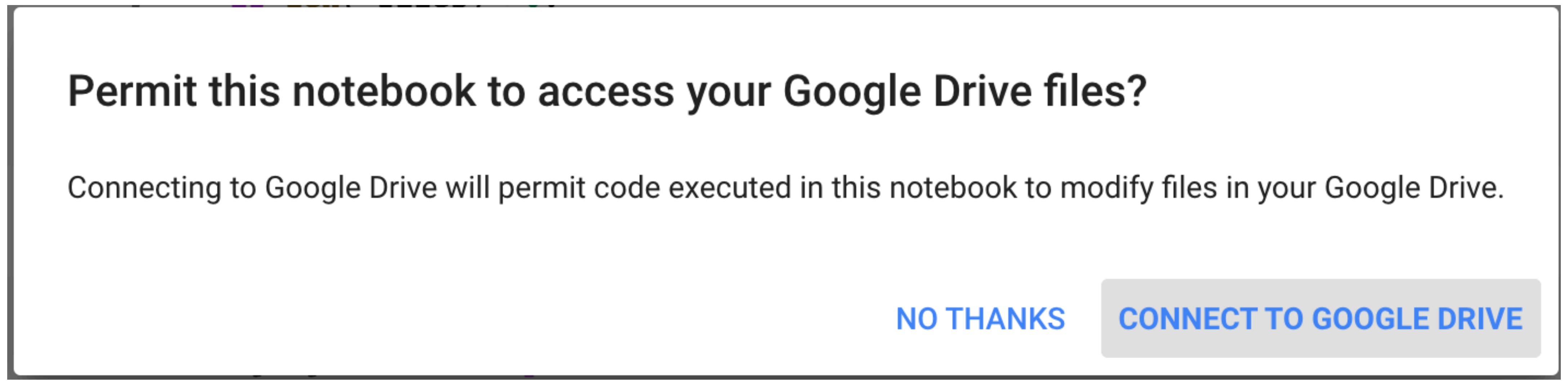
Mount your Google Drive to Colab (User Interface - UI)



This is the sidebar menu for managing files

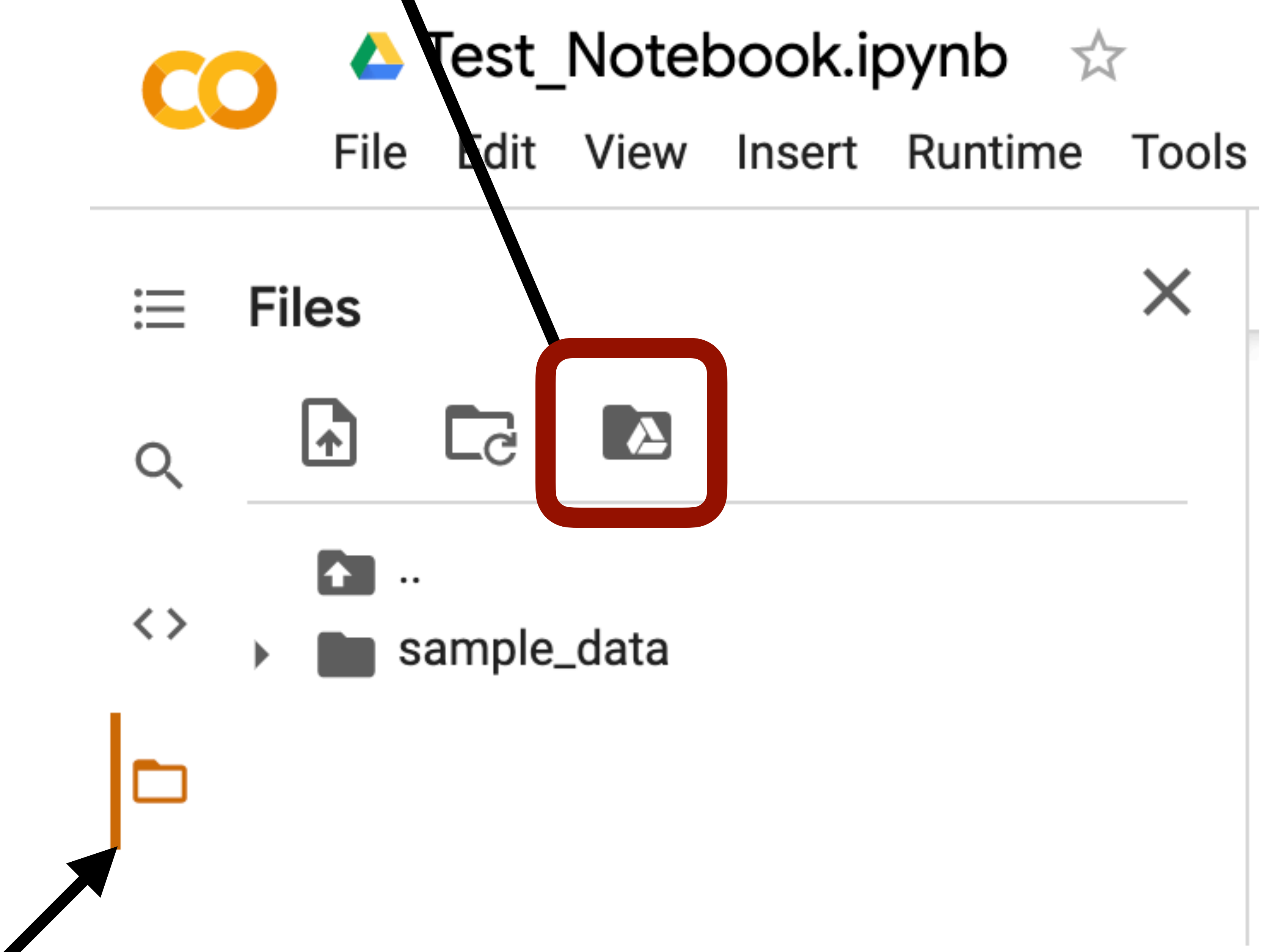
Mount your Google Drive to Colab (User Interface - UI)

Click here for a pop-up to open



Permit this notebook to access your Google Drive files?

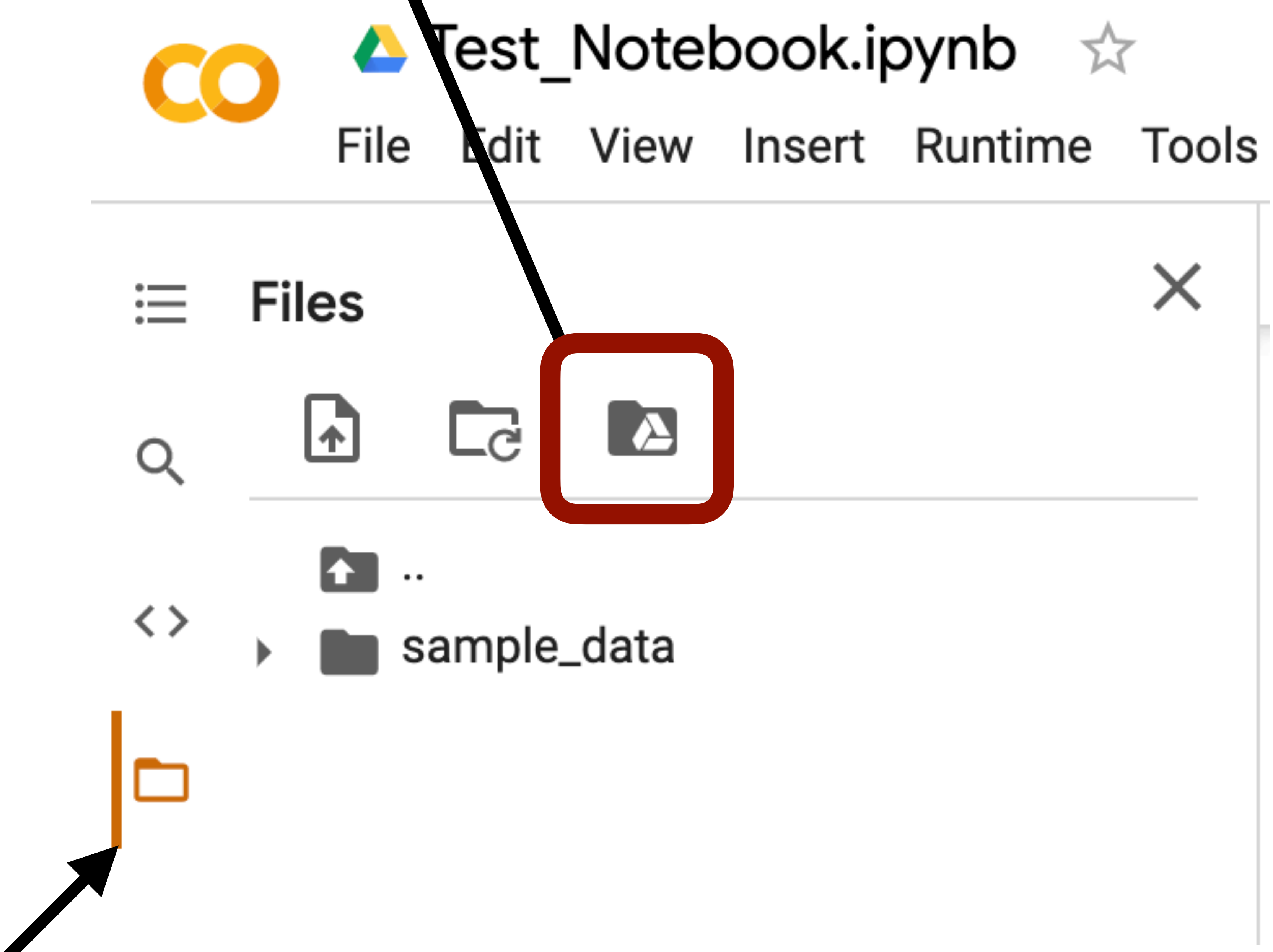
Connecting to Google Drive will permit code executed in this notebook to modify files in your Google Drive.



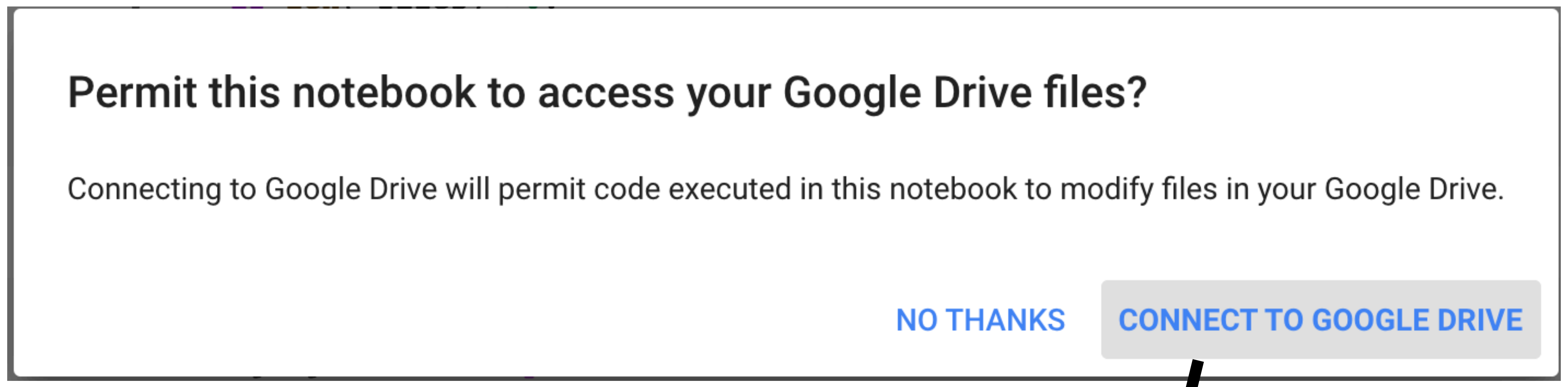
This is the sidebar menu for managing files

Mount your Google Drive to Colab (User Interface - UI)

Click here for a pop-up to open



This is the sidebar menu for managing files

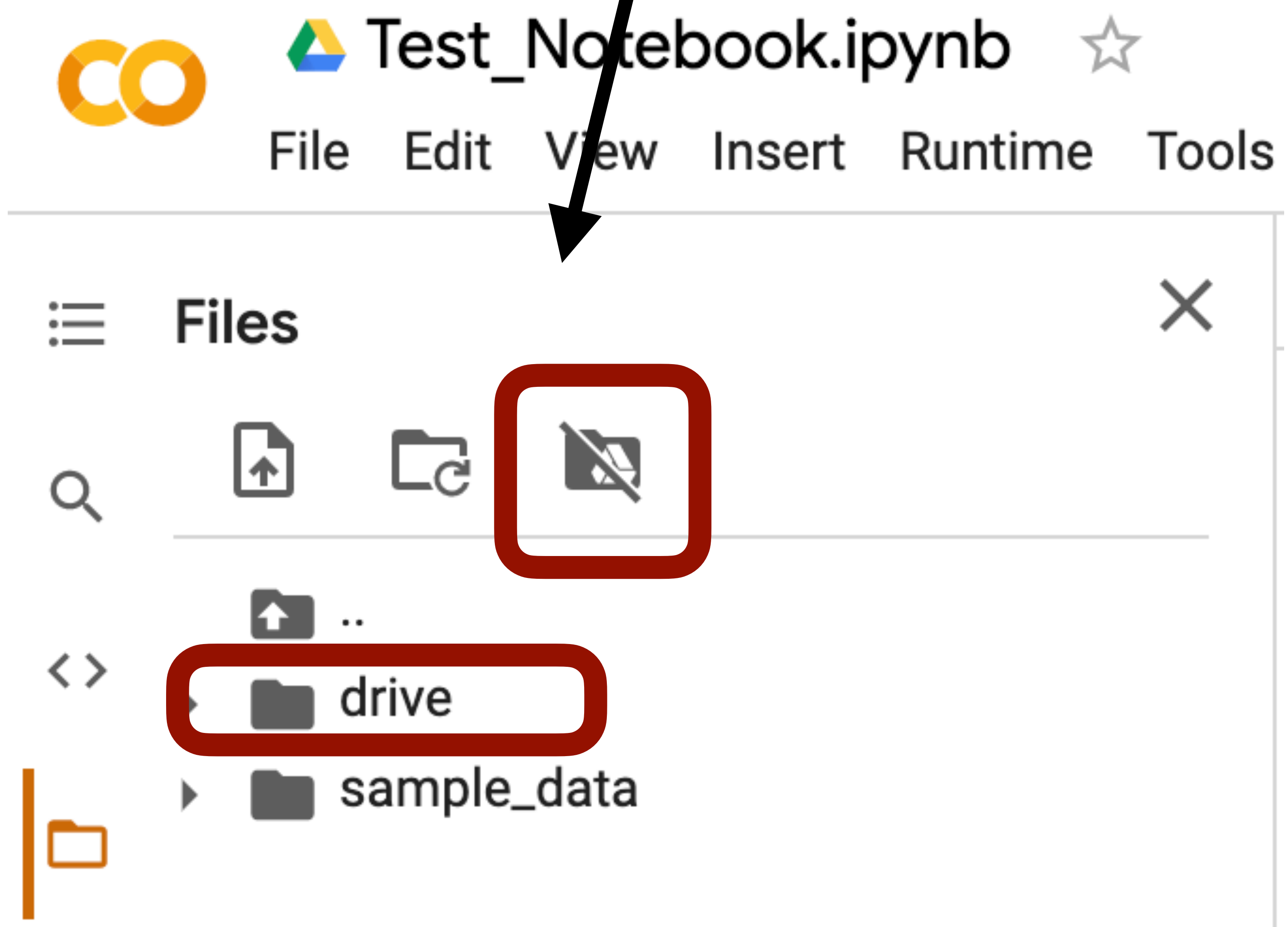


Permit this notebook to access your Google Drive files?

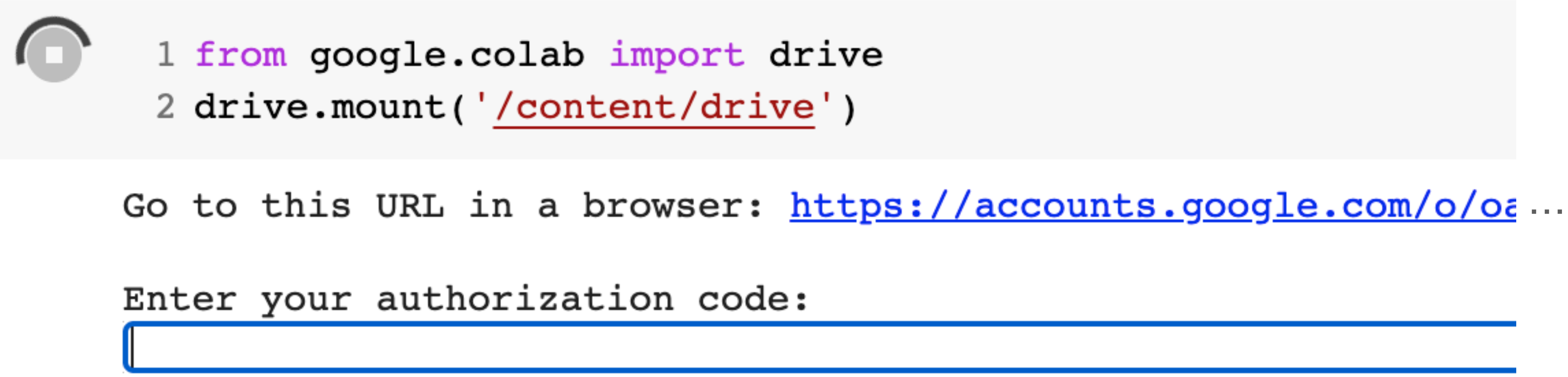
Connecting to Google Drive will permit code executed in this notebook to modify files in your Google Drive.

NO THANKS CONNECT TO GOOGLE DRIVE

This method only works if you are the only editor on a notebook, but doing it this way means you don't have to re-mount Google Drive every runtime



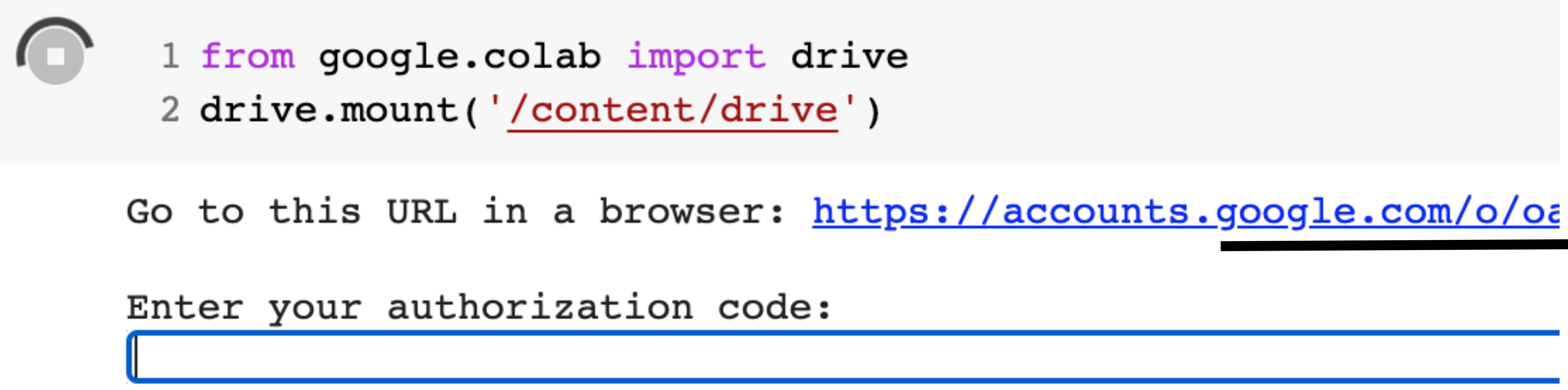
Mount your Google Drive to Colab (code)



```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oa…

Enter your authorization code:
```

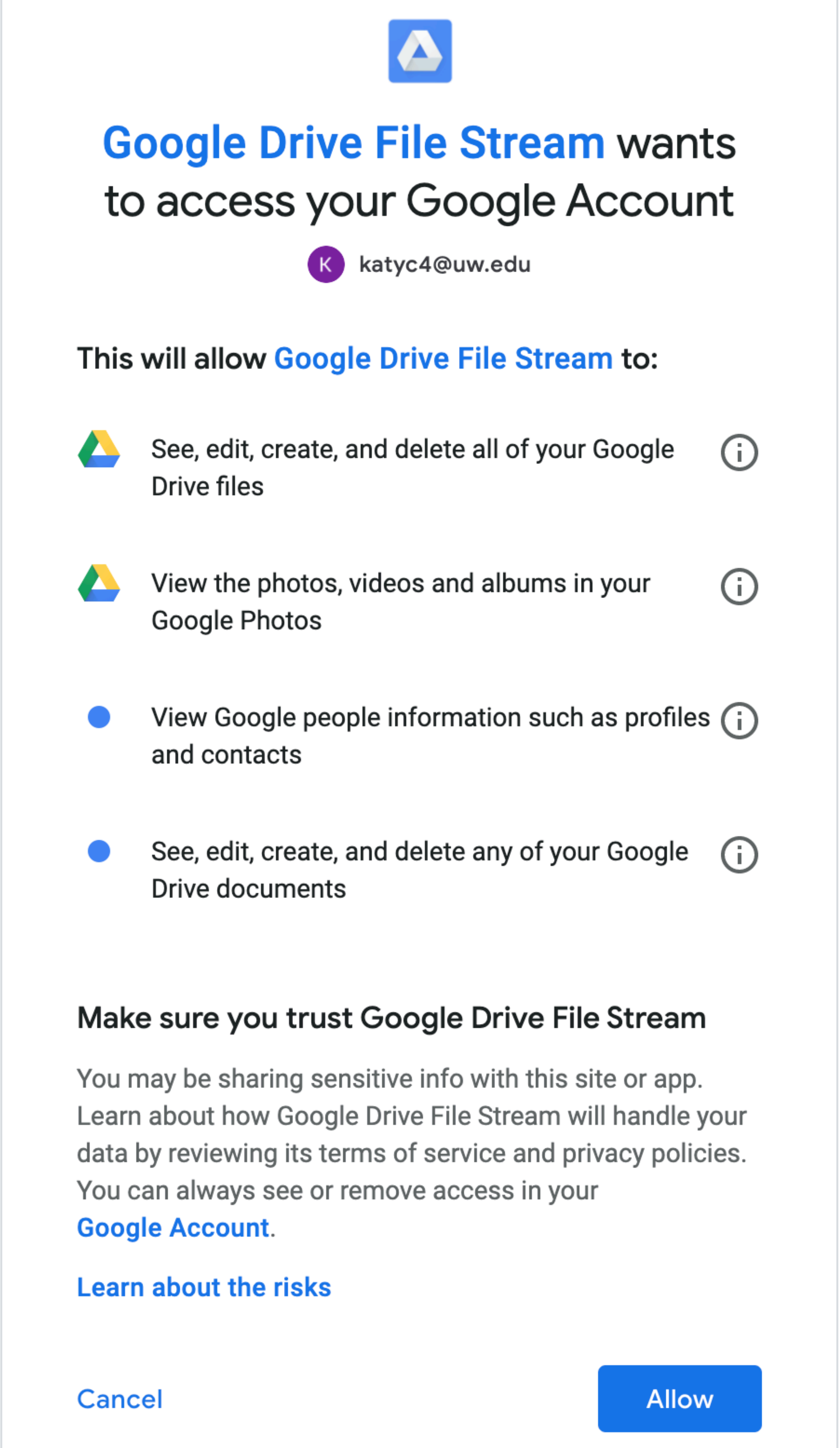
Mount your Google Drive to Colab (code)



```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oa

Enter your authorization code:
```

Click the link that appears after running the cell, a new tab will open



Google Drive File Stream wants to access your Google Account

katyc4@uw.edu

This will allow **Google Drive File Stream** to:

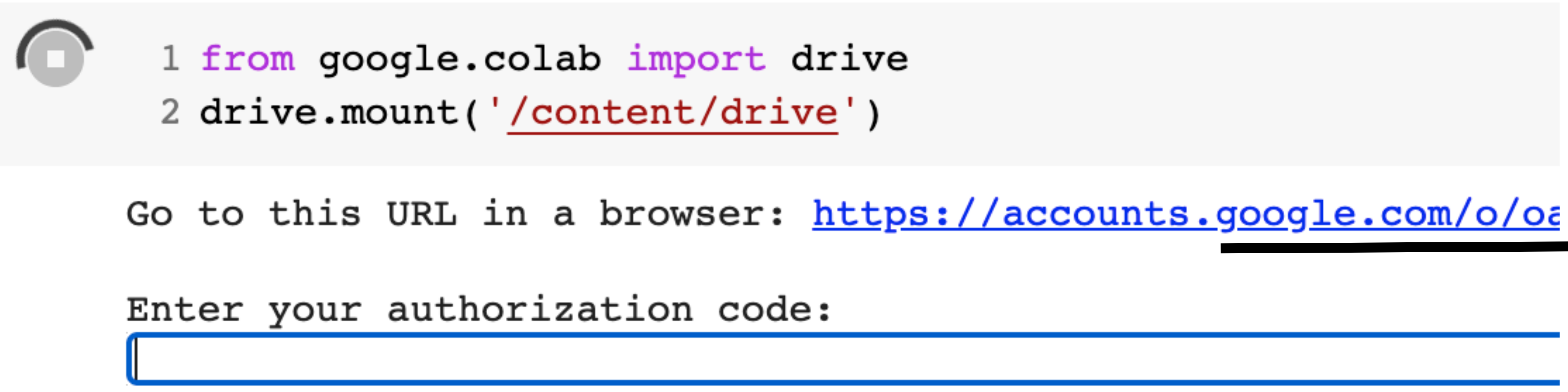
- See, edit, create, and delete all of your Google Drive files
- View the photos, videos and albums in your Google Photos
- View Google people information such as profiles and contacts
- See, edit, create, and delete any of your Google Drive documents

Make sure you trust Google Drive File Stream

You may be sharing sensitive info with this site or app. Learn about how Google Drive File Stream will handle your data by reviewing its terms of service and privacy policies. You can always see or remove access in your Google Account.

Learn about the risks

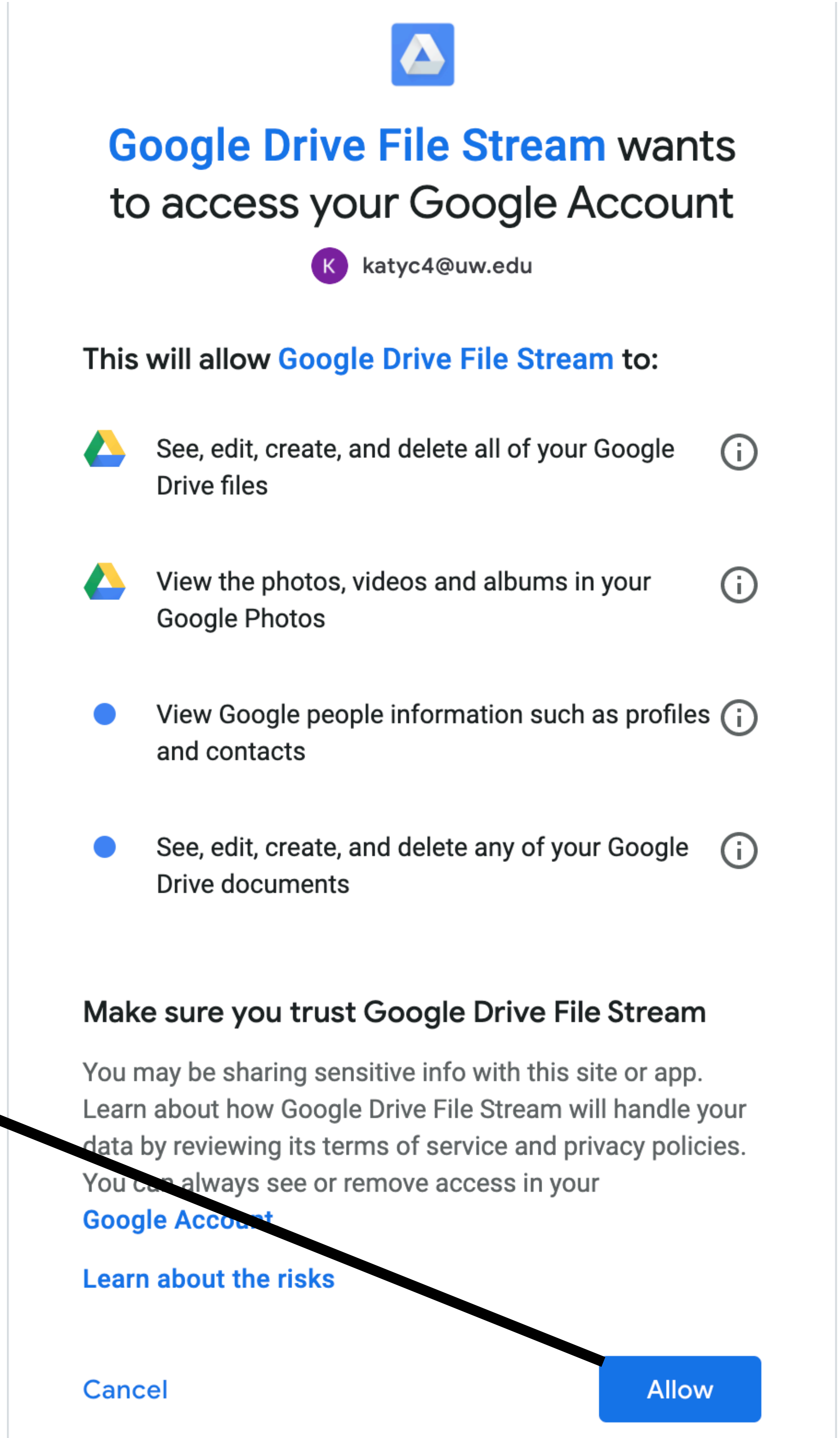
Mount your Google Drive to Colab (code)



```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oa

Enter your authorization code:
```

Click the link that appears after running the cell, a new tab will open



Google Drive File Stream wants to access your Google Account

katyc4@uw.edu

This will allow **Google Drive File Stream** to:

- See, edit, create, and delete all of your Google Drive files
- View the photos, videos and albums in your Google Photos
- View Google people information such as profiles and contacts
- See, edit, create, and delete any of your Google Drive documents

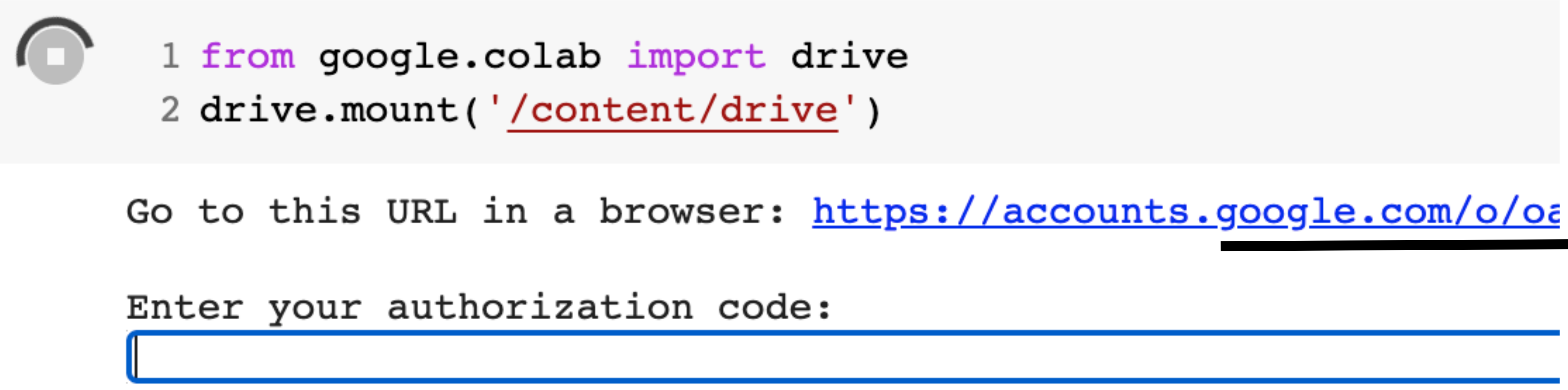
Make sure you trust Google Drive File Stream

You may be sharing sensitive info with this site or app. Learn about how Google Drive File Stream will handle your data by reviewing its terms of service and privacy policies. You can always see or remove access in your Google Account

Learn about the risks

Clicking **Allow** brings you to a new page with an authorization code.
Copy and past it into the notebook.

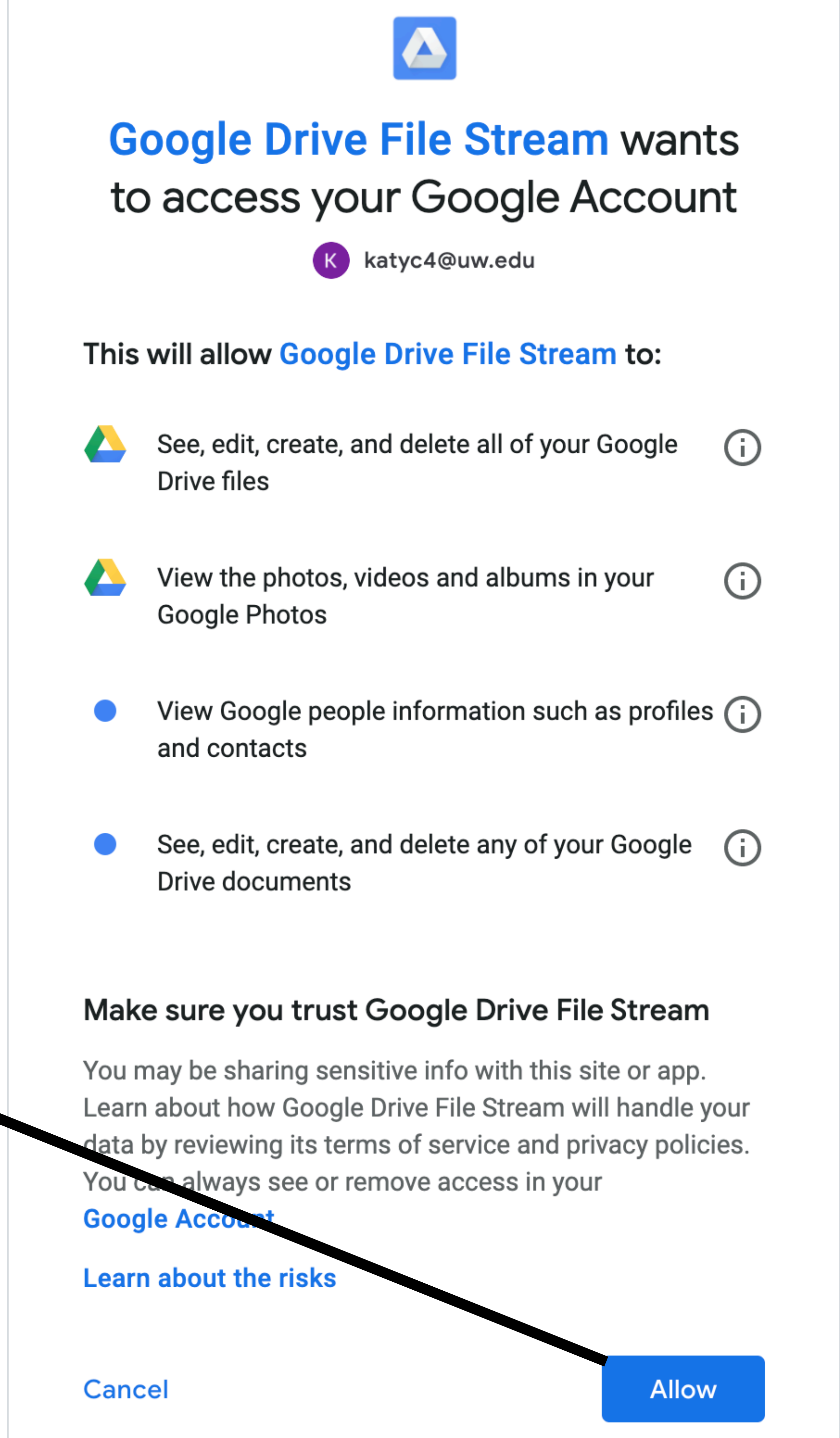
Mount your Google Drive to Colab (code)



```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oa

Enter your authorization code:
```

Click the link that appears after running the cell, a new tab will open



Google Drive File Stream wants to access your Google Account

katyc4@uw.edu

This will allow **Google Drive File Stream** to:

- See, edit, create, and delete all of your Google Drive files
- View the photos, videos and albums in your Google Photos
- View Google people information such as profiles and contacts
- See, edit, create, and delete any of your Google Drive documents

Make sure you trust Google Drive File Stream

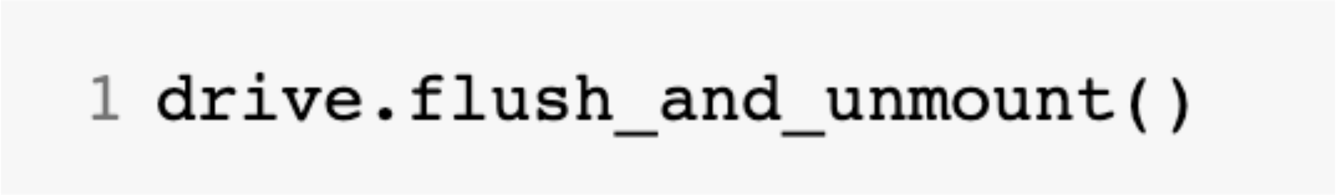
You may be sharing sensitive info with this site or app. Learn about how Google Drive File Stream will handle your data by reviewing its terms of service and privacy policies. You can always see or remove access in your **Google Account**

Learn about the risks

Clicking **Allow** brings you to a new page with an authorization code.
Copy and past it into the notebook.

Load the data from the files into Python here!

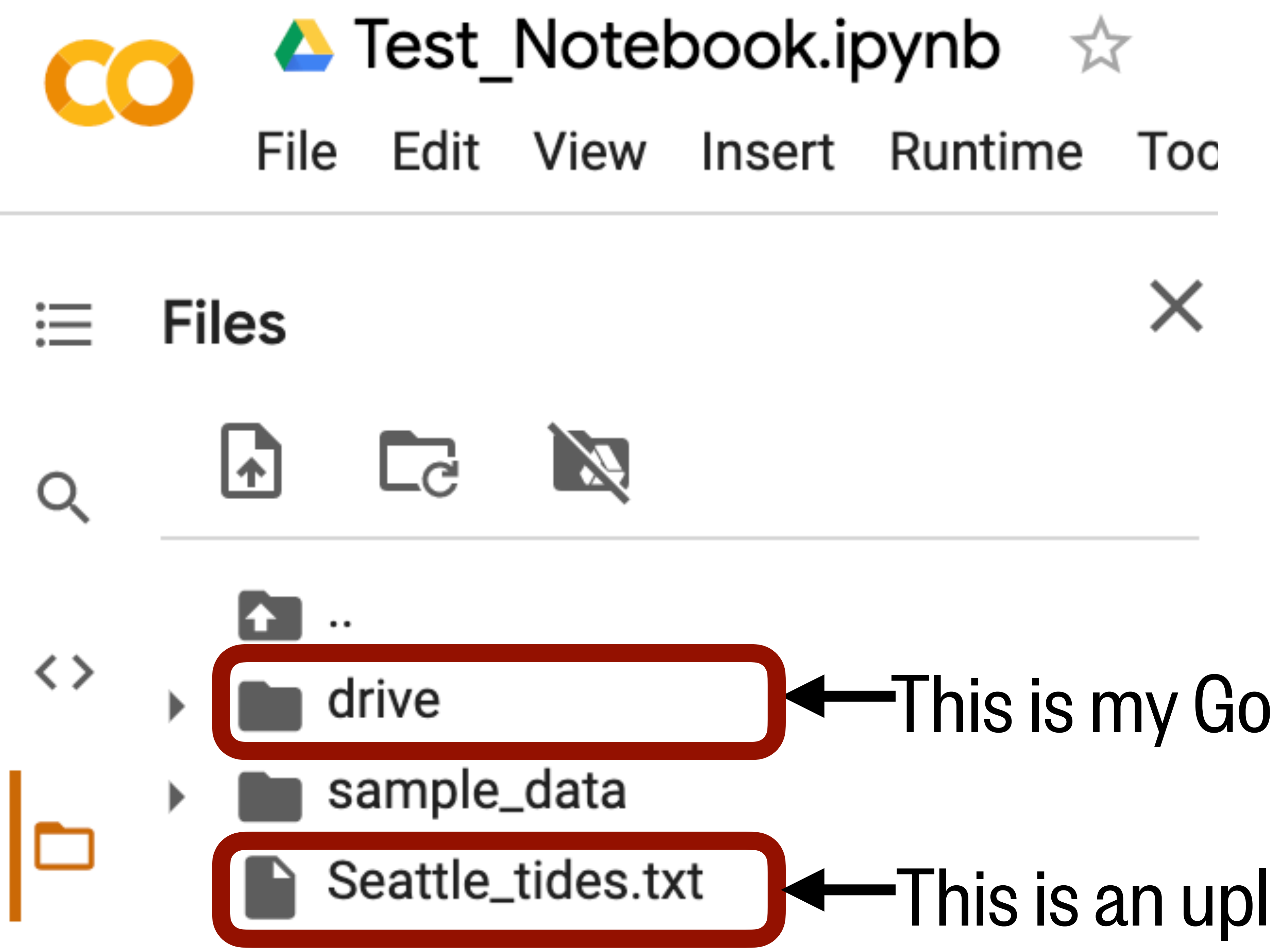
Un-mounting the Google Drive once you have loaded your data is preferred.



```
drive.flush_and_unmount()
```

A note about file paths

After mounting your drive or uploading your files, they should appear in your sidebar for **Files**



This is my Google Drive

This is an uploaded file

When you want to access those files (to load their data), you will use its **path**

Path for uploaded files:
a string containing the file name

```
filepath = 'Seattle_tides.txt'
```

Path in Google Drive:
a string containing the file name, preceded by its folders and separated by /

```
filepath = 'drive/My Drive/Data_folder/Seattle_tides.txt'
```

These are the folders where you put your data file in your Google Drive

What we'll cover in this lesson

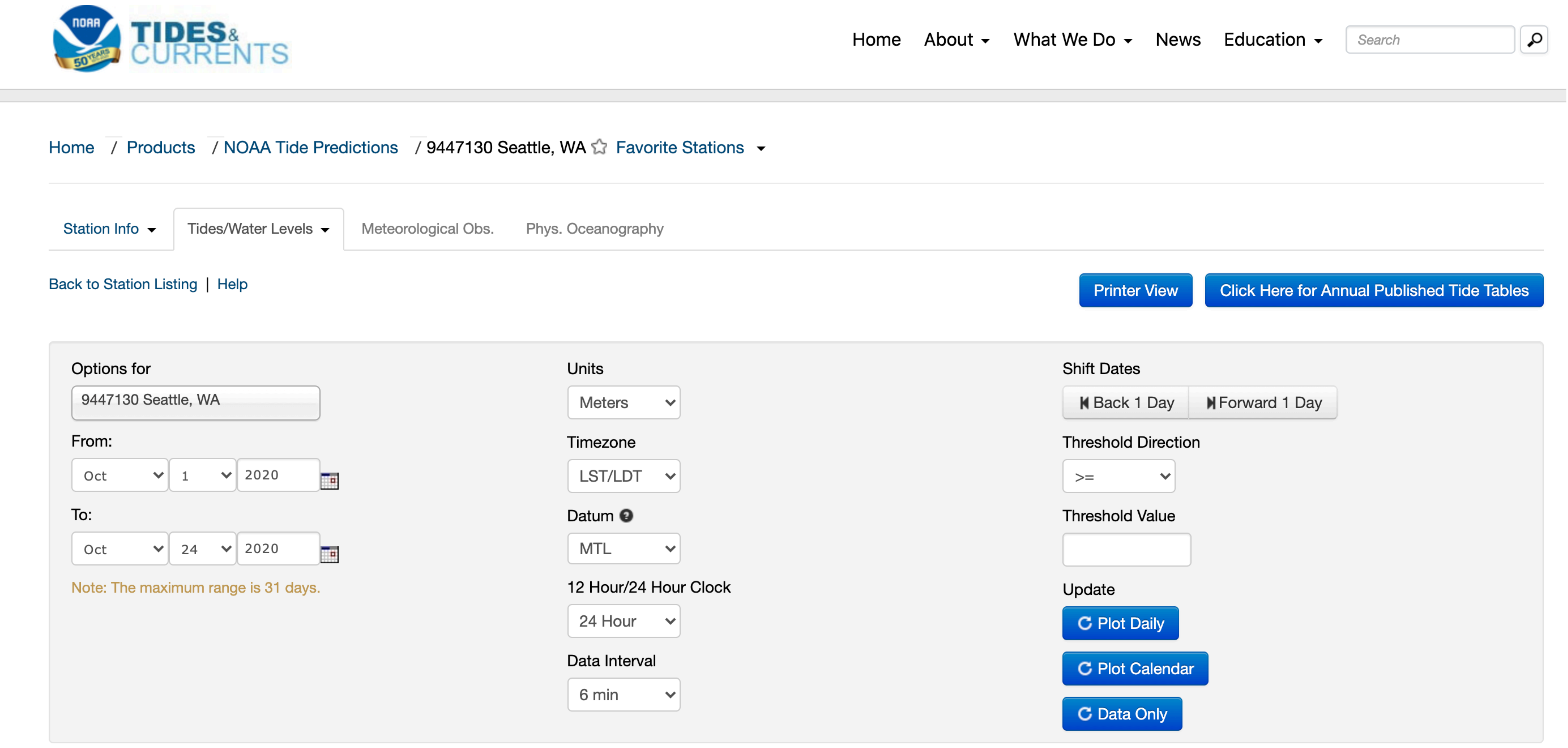
1. Loading and saving files to Google Colab

2. **Loading data using readlines and numpy**

3. Intro to plotting

Sample data - Seattle tidal record

Data source: https://tidesandcurrents.noaa.gov/noaatidepredictions.html?id=9447130&units=metric&bdate=20201001&edate=20201024&timezone=LST/LDT&clock=24hour&datum=MTL&interval=6&action=data



Options for: 9447130 Seattle, WA

From: Oct 1 2020

To: Oct 24 2020

Note: The maximum range is 31 days.

Units: Meters

Timezone: LST/LDT

Datum: MTL

12 Hour/24 Hour Clock: 24 Hour

Data Interval: 6 min

Shift Dates

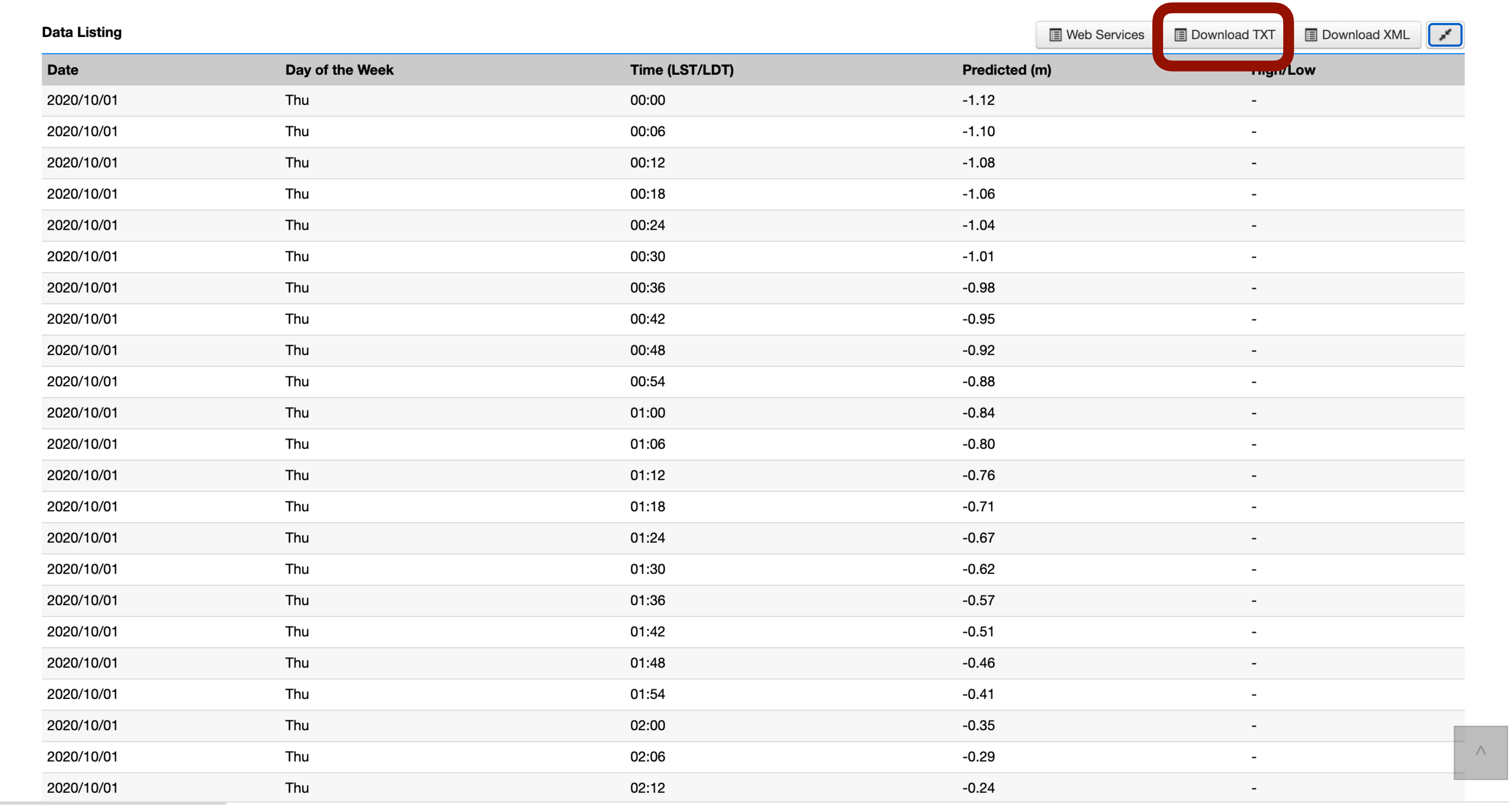
Threshold Direction: >=

Threshold Value:

Update

Sample data - Seattle tidal record

Data source: https://tidesandcurrents.noaa.gov/noaatidepredictions.html?id=9447130&units=metric&bdate=20201001&edate=20201024&timezone=LST/LDT&clock=24hour&datum=MTL&interval=6&action=data



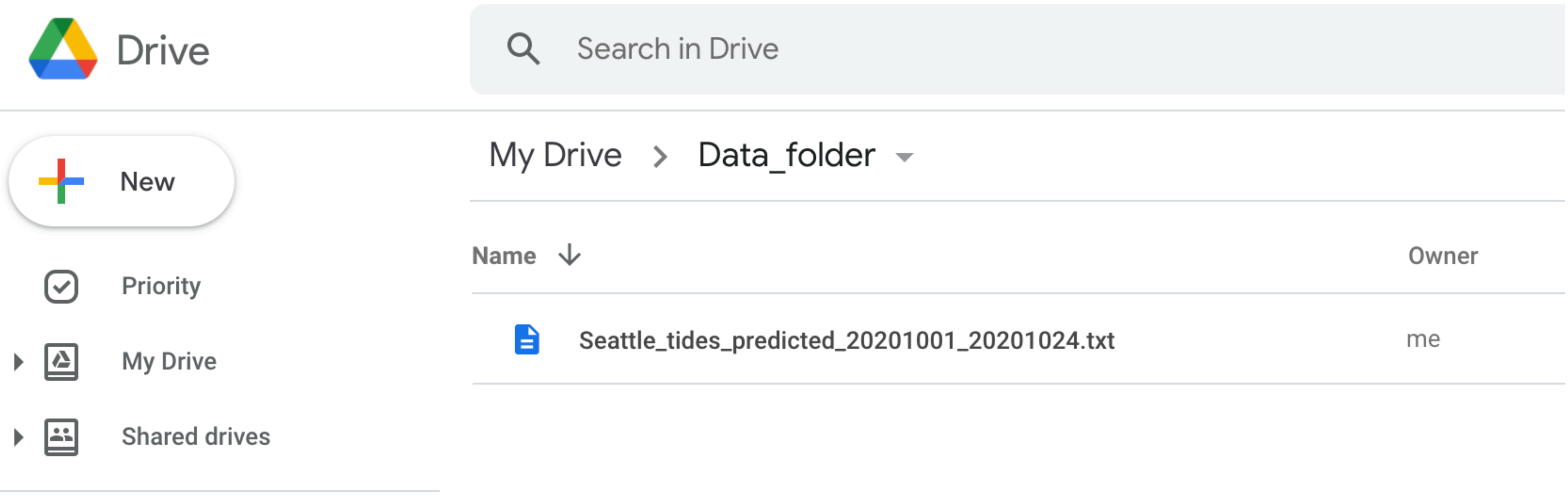
Data Listing

Date	Day of the Week	Time (LST/LDT)	Predicted (m)	High/Low
2020/10/01	Thu	00:00	-1.12	-
2020/10/01	Thu	00:06	-1.10	-
2020/10/01	Thu	00:12	-1.08	-
2020/10/01	Thu	00:18	-1.06	-
2020/10/01	Thu	00:24	-1.04	-
2020/10/01	Thu	00:30	-1.01	-
2020/10/01	Thu	00:36	-0.98	-
2020/10/01	Thu	00:42	-0.95	-
2020/10/01	Thu	00:48	-0.92	-
2020/10/01	Thu	00:54	-0.88	-
2020/10/01	Thu	01:00	-0.84	-
2020/10/01	Thu	01:06	-0.80	-
2020/10/01	Thu	01:12	-0.76	-
2020/10/01	Thu	01:18	-0.71	-
2020/10/01	Thu	01:24	-0.67	-
2020/10/01	Thu	01:30	-0.62	-
2020/10/01	Thu	01:36	-0.57	-
2020/10/01	Thu	01:42	-0.51	-
2020/10/01	Thu	01:48	-0.46	-
2020/10/01	Thu	01:54	-0.41	-
2020/10/01	Thu	02:00	-0.35	-
2020/10/01	Thu	02:06	-0.29	-
2020/10/01	Thu	02:12	-0.24	-

Sample data - Seattle tidal record

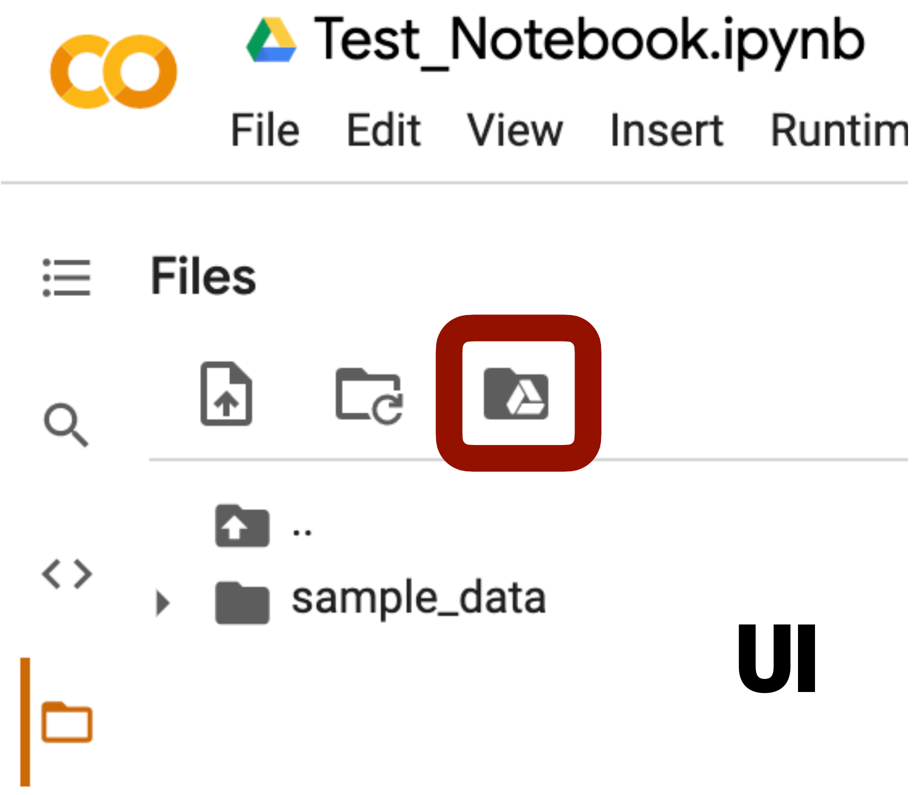
Data source: https://tidesandcurrents.noaa.gov/noaatidepredictions.html?id=9447130&units=metric&bdate=20201001&edate=20201024&timezone=LST/LDT&clock=24hour&datum=MTL&interval=6&action=data

Upload the resulting .txt file to your Google Drive data folder...

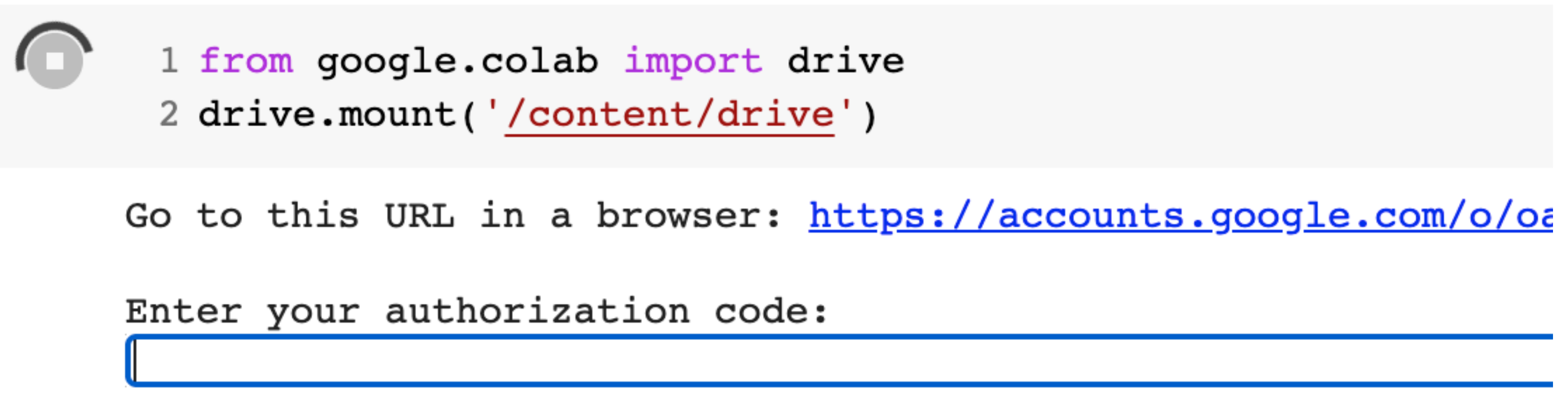


Name	Owner
Seattle_tides_predicted_20201001_20201024.txt	me

Then mount your Google Drive.



UI

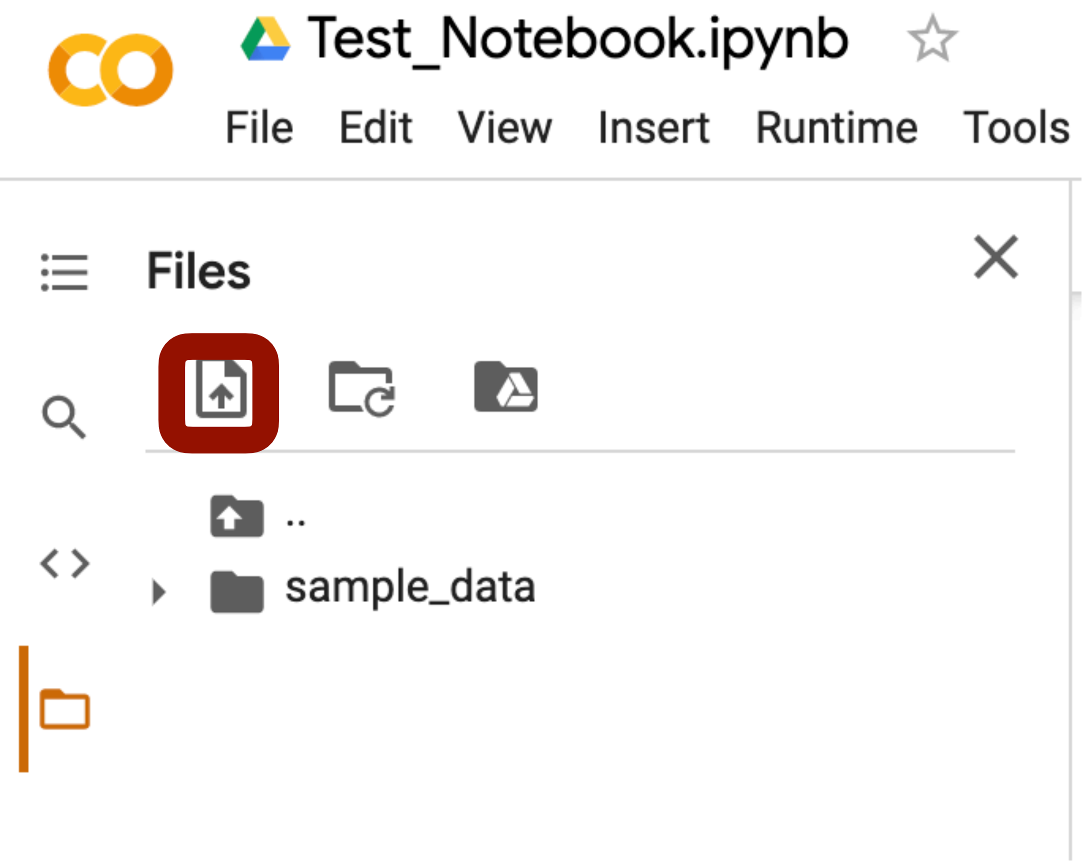


```
from google.colab import drive
drive.mount('/content/drive')
```

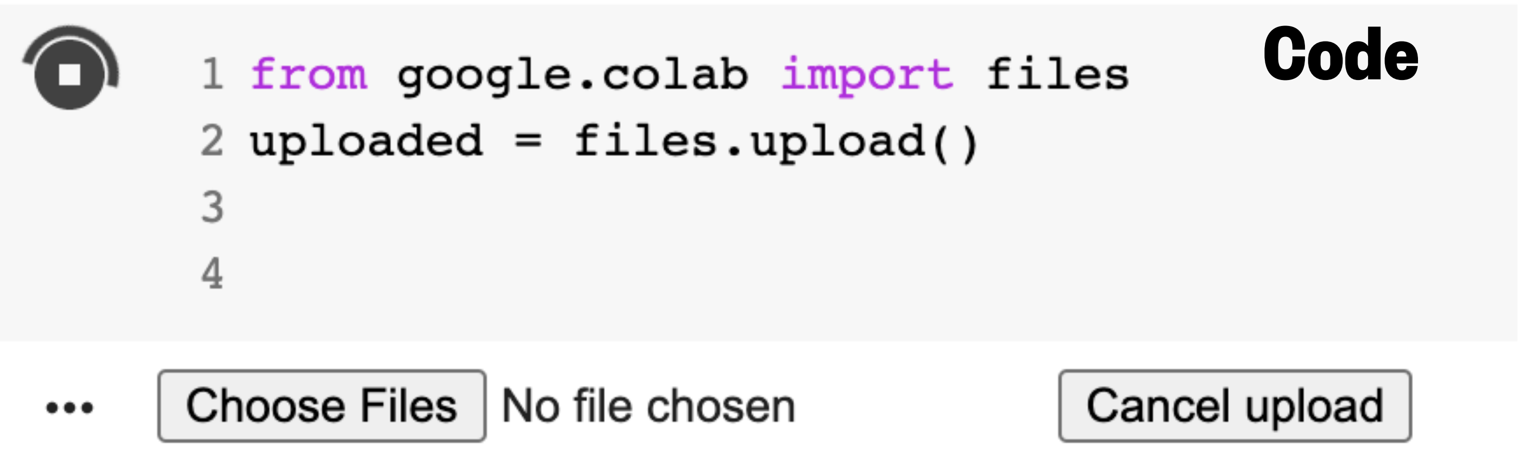
```
Go to this URL in a browser: https://accounts.google.com/o/oa

Enter your authorization code:
```

Code

Or upload directly to Google Colab.



UI



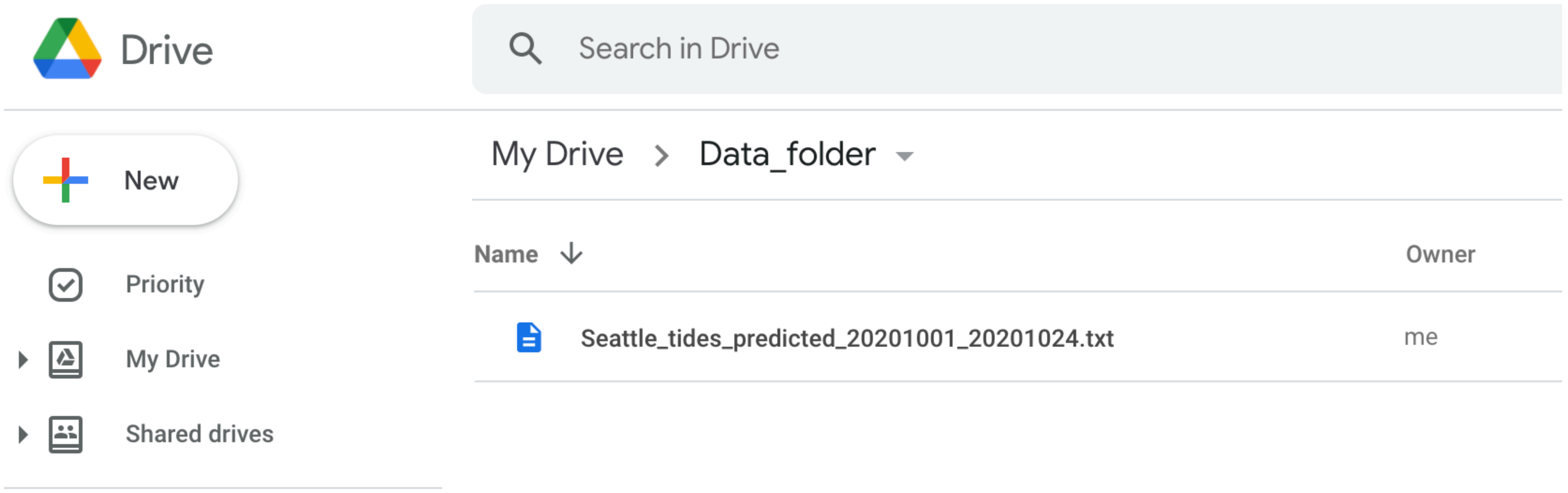
```
from google.colab import files
uploaded = files.upload()
```

Code

Sample data - Seattle tidal record

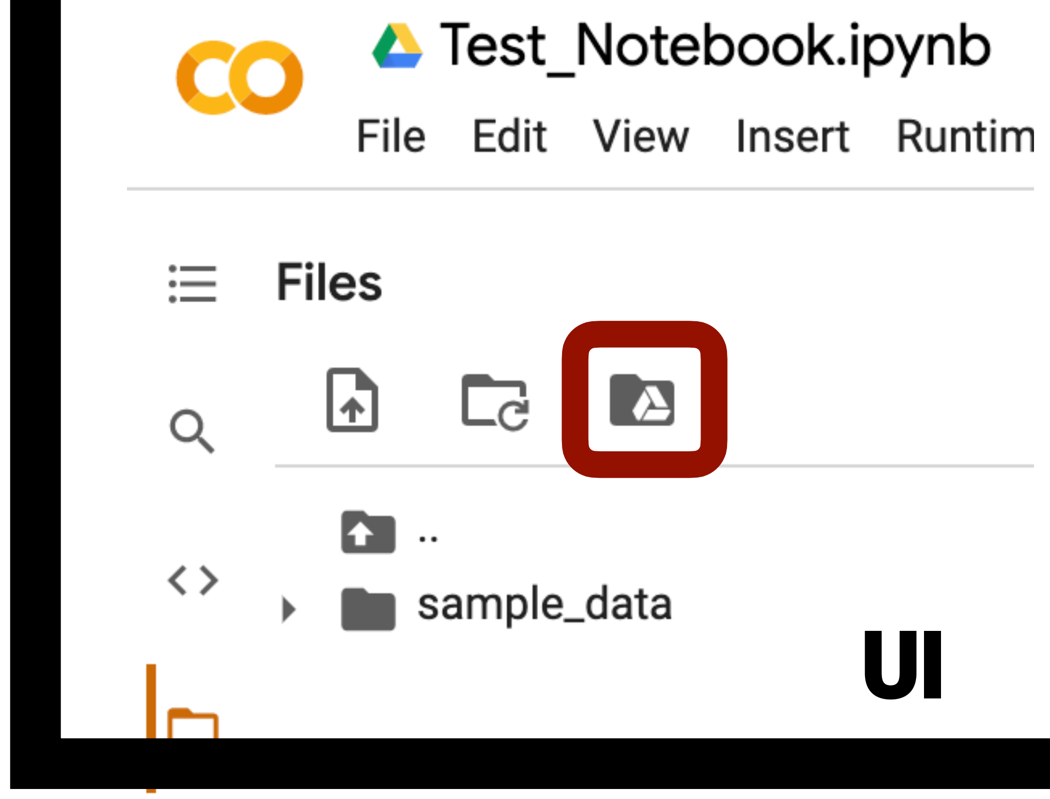
Data source: https://tidesandcurrents.noaa.gov/noaatidepredictions.html?id=9447130&units=metric&bdate=20201001&edate=20201024&timezone=LST/LDT&clock=24hour&datum=MTL&interval=6&action=data

Upload the resulting .txt file to your Google Drive data folder…

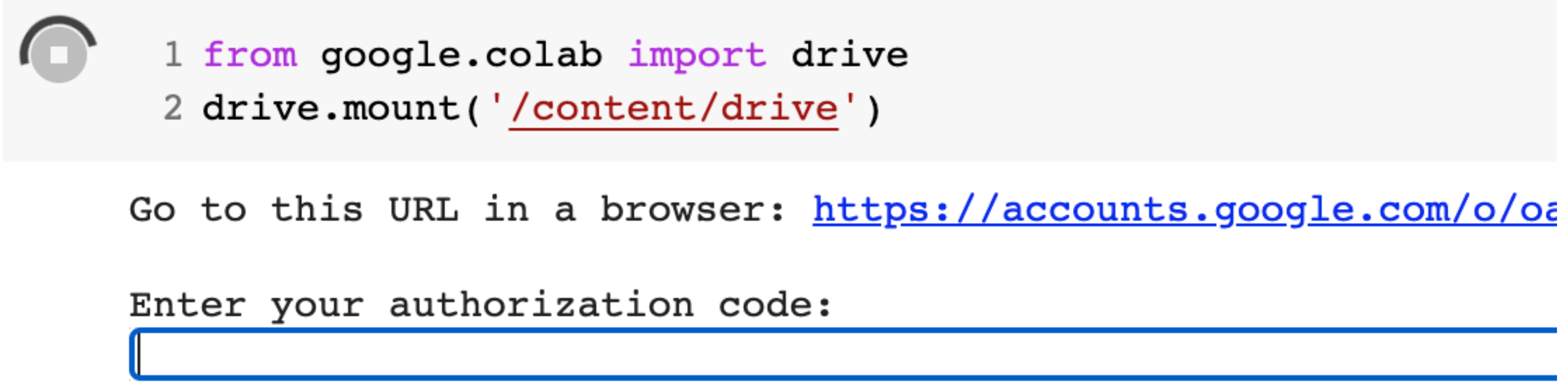


Name	Owner
Seattle_tides_predicted_20201001_20201024.txt	me

Then mount your Google Drive.



UI

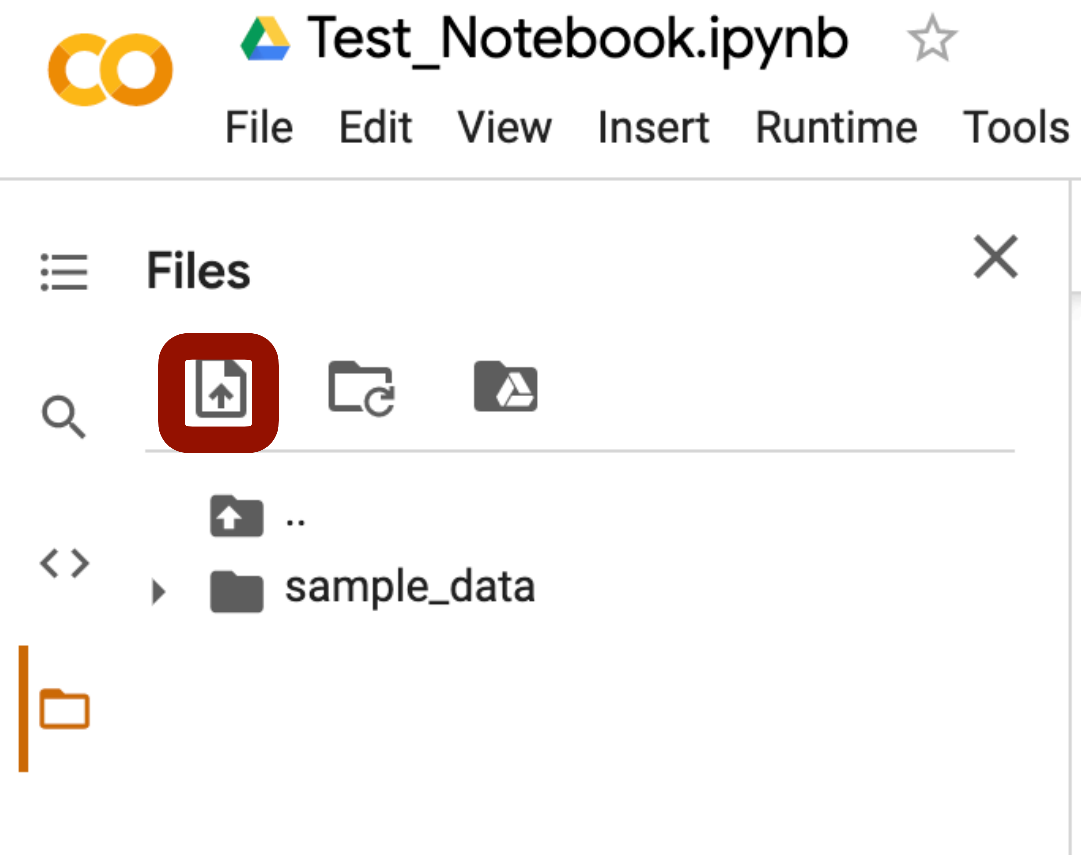


```
from google.colab import drive
drive.mount('/content/drive')
```

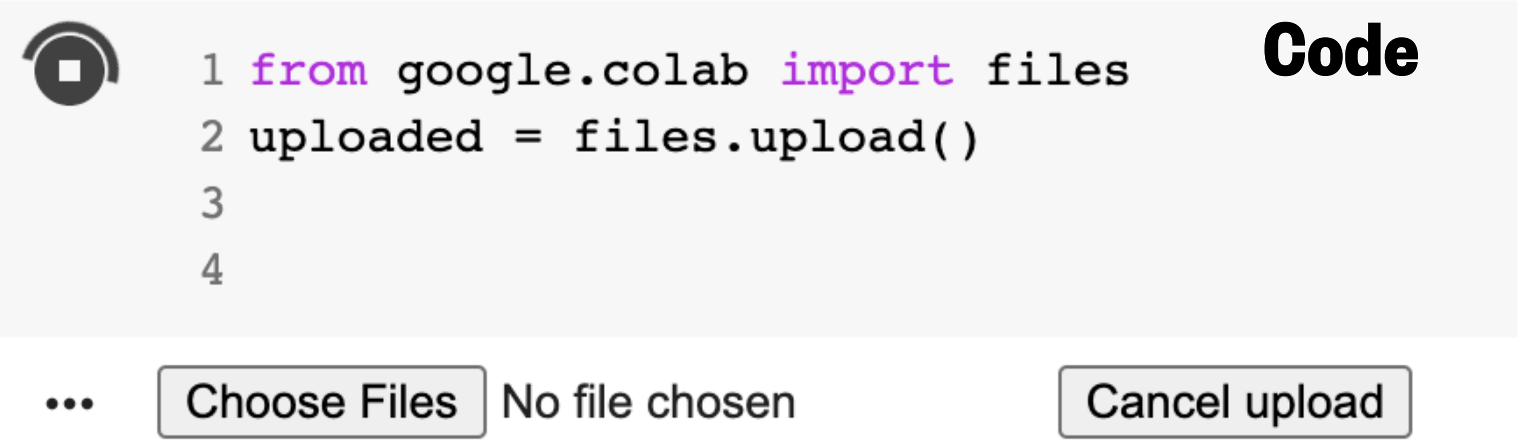
```
Go to this URL in a browser: https://accounts.google.com/o/oa

Enter your authorization code:
```

Code

Or upload directly to Google Colab.



UI

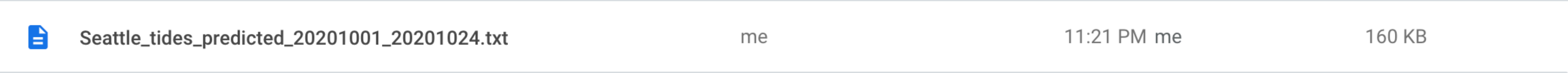


```
from google.colab import files
uploaded = files.upload()


```

Code

Getting to know your data

Our data file can tell us a little…

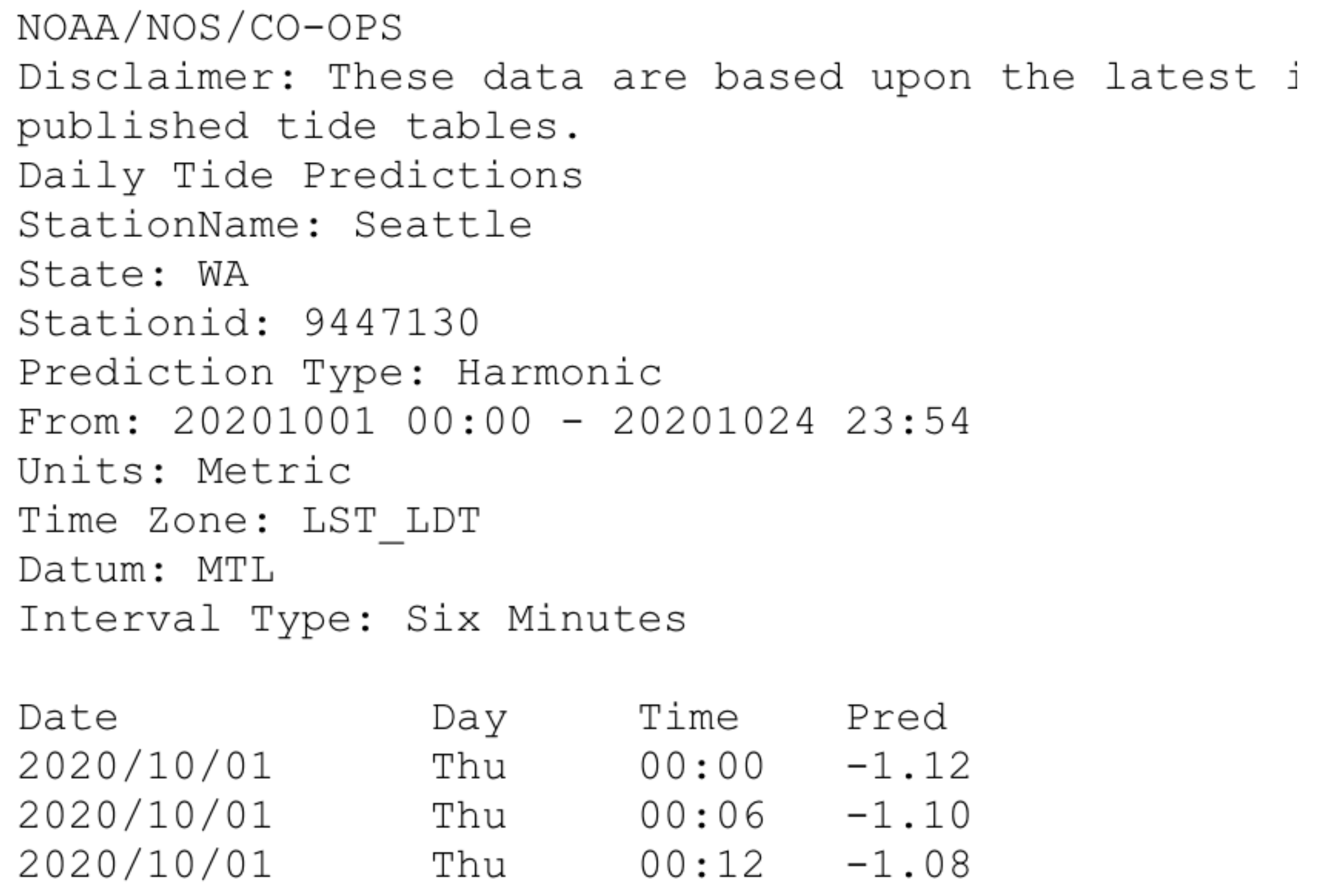


Seattle_tides_predicted_20201001_20201024.txt	me	11:21 PM me	160 KB

But not what the inside looks like. Look inside by:

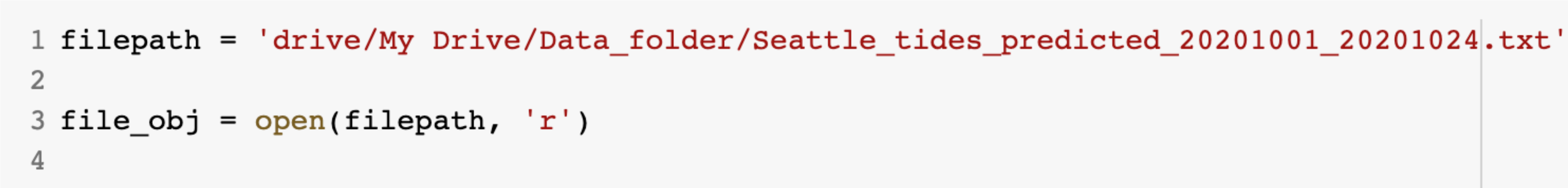
MS Word is NOT a text editor!

1) Opening the file using a text editor



```
NOAA/NOS/CO-OPS
Disclaimer: These data are based upon the latest i
published tide tables.
Daily Tide Predictions
StationName: Seattle
State: WA
Stationid: 9447130
Prediction Type: Harmonic
From: 20201001 00:00 - 20201024 23:54
Units: Metric
Time Zone: LST_LDT
Datum: MTL
Interval Type: Six Minutes

Date            Day     Time     Pred
2020/10/01      Thu     00:00    -1.12
2020/10/01      Thu     00:06    -1.10
2020/10/01      Thu     00:12    -1.08
```

2) Opening the file using Python

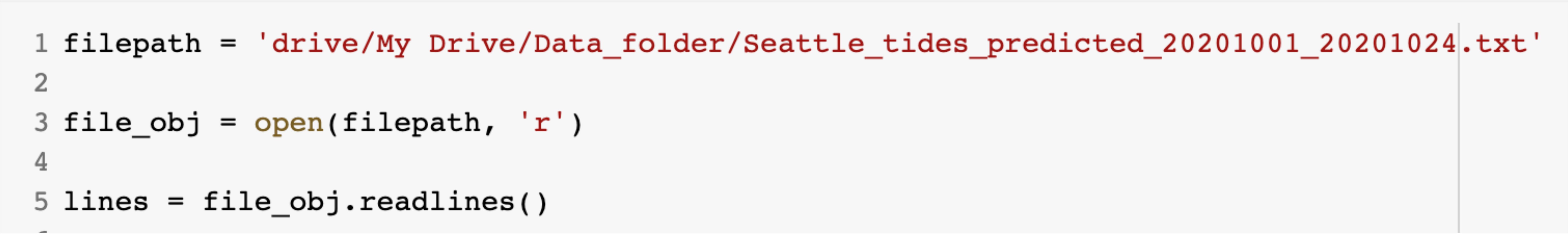


```python
filepath = 'drive/My Drive/Data_folder/Seattle_tides_predicted_20201001_20201024.txt'

file_obj = open(filepath, 'r')

```

Using **open** does not read the file. Instead, it creates a file object that can be read later. Think of it like opening a book…

readlines( )

To read the file after opening, use the function **readlines( )**

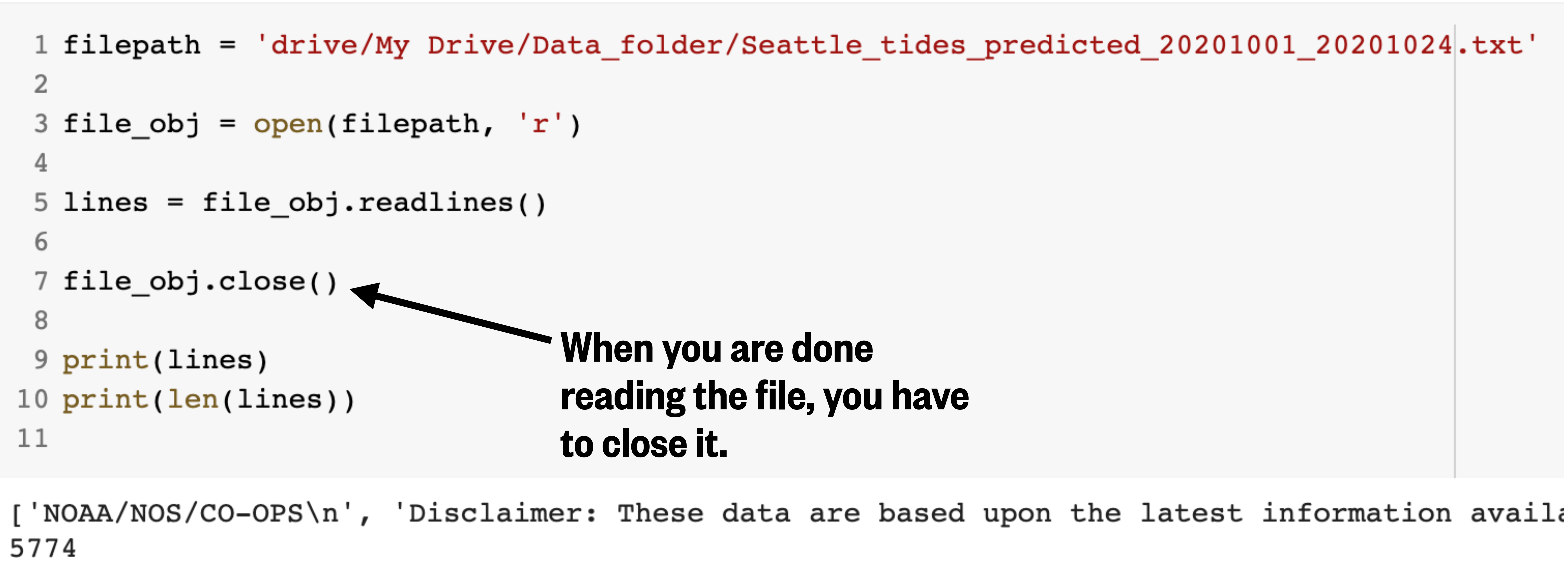


```python
filepath = 'drive/My Drive/Data_folder/Seattle_tides_predicted_20201001_20201024.txt'

file_obj = open(filepath, 'r')

lines = file_obj.readlines()
```

This function loads the entire file into memory and will return a list object containing each of the lines in your file as items.

readlines( )

To read the file after opening, use the function **readlines( )**



```
filepath = 'drive/My Drive/Data_folder/Seattle_tides_predicted_20201001_20201024.txt'

file_obj = open(filepath, 'r')

lines = file_obj.readlines()

file_obj.close()

print(lines)
print(len(lines))

```

```
['NOAA/NOS/CO-OPS\n', 'Disclaimer: These data are based upon the latest information availa
5774
```

When you are done reading the file, you have to close it.

When you print the list, it is not very easy to look at.
The **len( )** function gives you the total number of lines.

readlines( )

To read the file after opening, use the function **readlines( )**

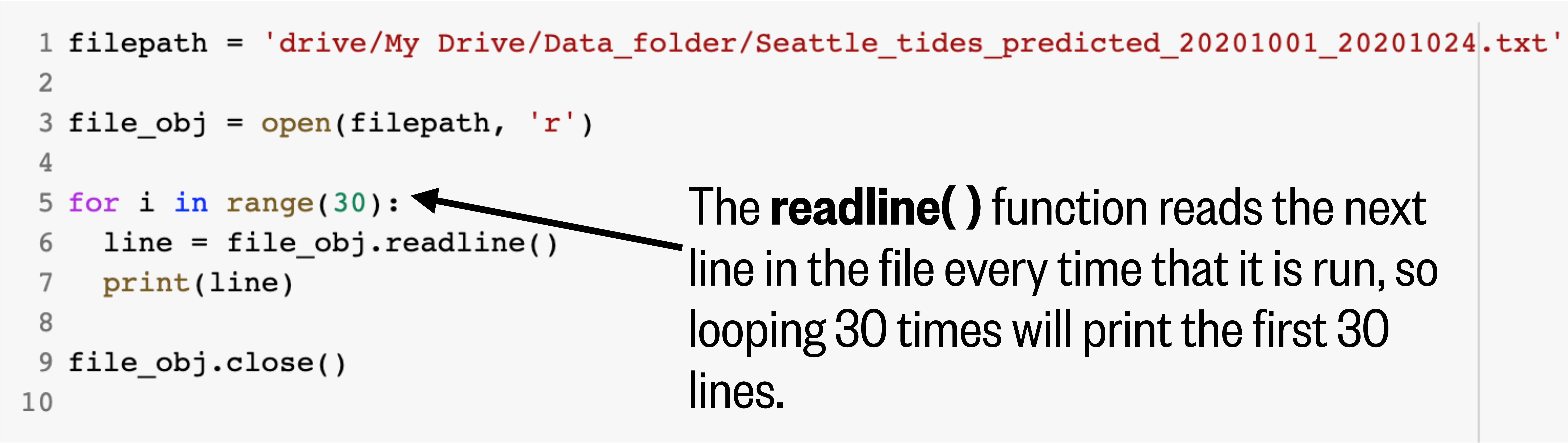
```python
filepath = 'drive/My Drive/Data_folder/Seattle_tides_predicted_20201001_20201024.txt'

file_obj = open(filepath, 'r')

lines = file_obj.readlines()

file_obj.close()

print(lines)
print(len(lines))

```

```
['NOAA/NOS/CO-OPS\n', 'Disclaimer: These data are based upon the latest information available as of the date of your request, and may differ from the publis
5774
```

The **len( )** function gives you the total number of lines.
When you print the list, it is not very easy to look at. Plus, loading files that are large can cause your code to slow down.

readline( )

Instead of reading the whole file at once with **readlines( )**, read each line as you go using **readline( )** and a for loop.

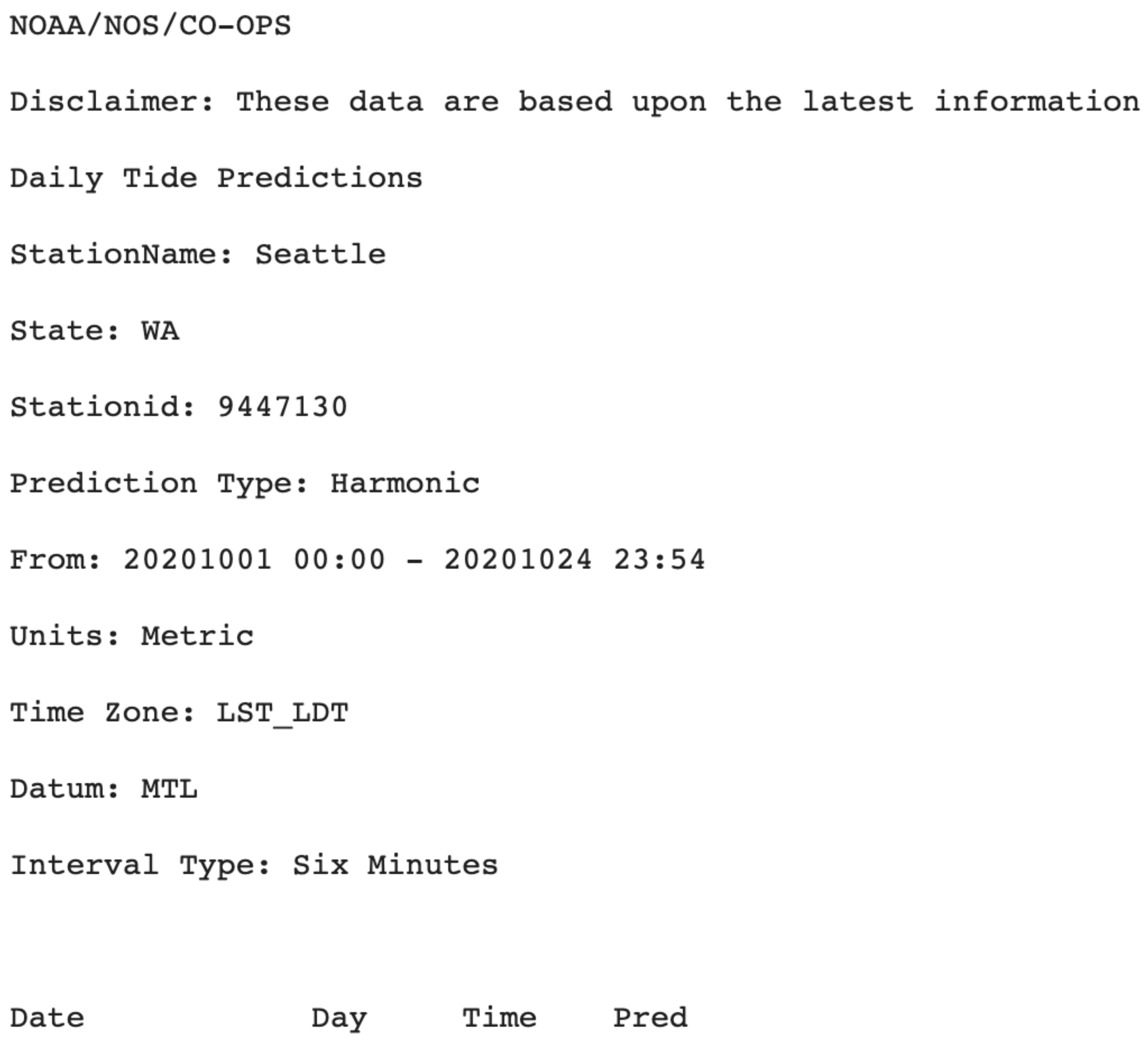


```
filepath = 'drive/My Drive/Data_folder/Seattle_tides_predicted_20201001_20201024.txt'

file_obj = open(filepath, 'r')

for i in range(30):
  line = file_obj.readline()
  print(line)

file_obj.close()

```

The **readline( )** function reads the next line in the file every time that it is run, so looping 30 times will print the first 30 lines.

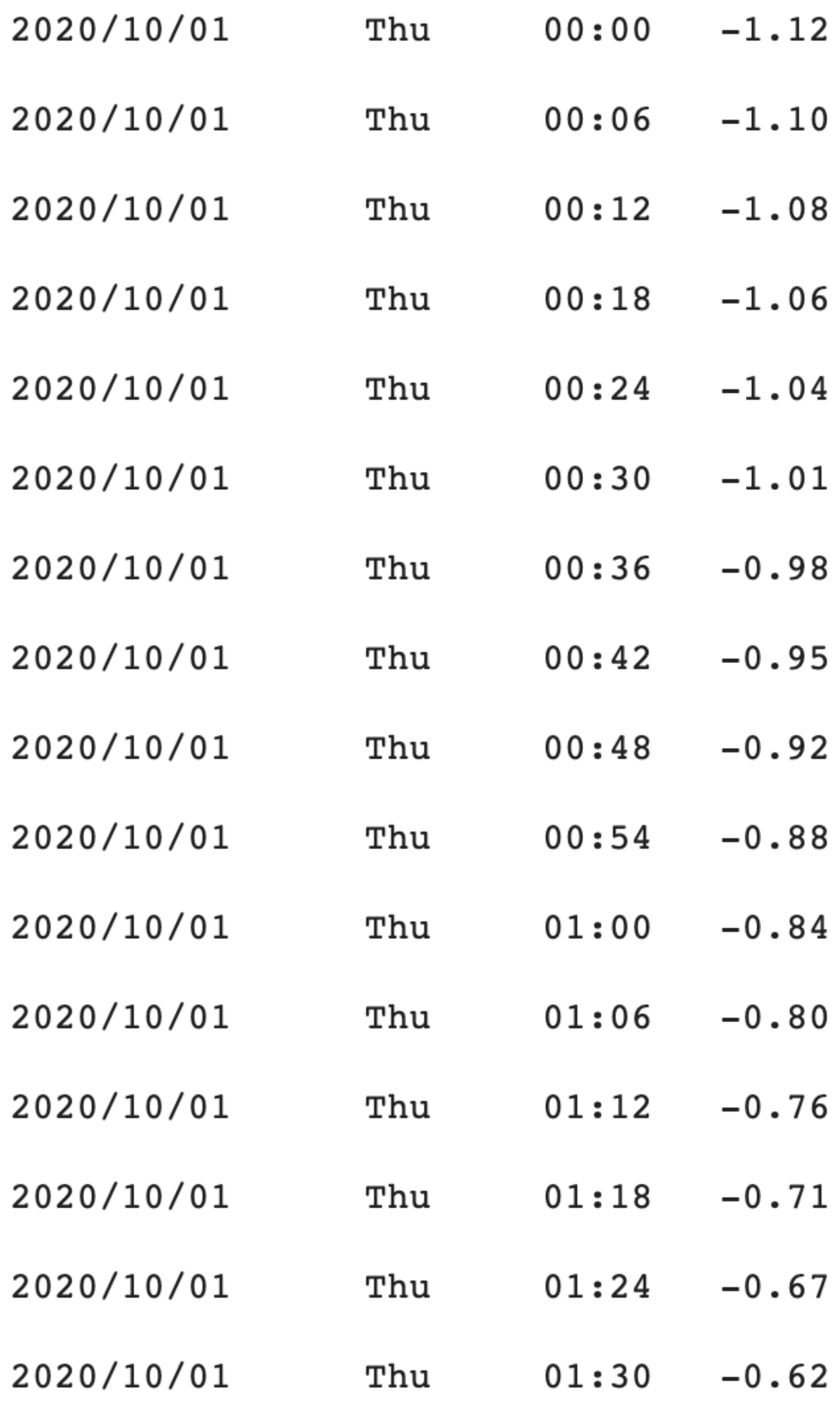
readline( )

Header



```
NOAA/NOS/CO-OPS

Disclaimer: These data are based upon the latest information

Daily Tide Predictions

StationName: Seattle

State: WA

Stationid: 9447130

Prediction Type: Harmonic

From: 20201001 00:00 - 20201024 23:54

Units: Metric

Time Zone: LST_LDT

Datum: MTL

Interval Type: Six Minutes


Date             Day      Time     Pred
```

Data



```
2020/10/01      Thu      00:00    -1.12
2020/10/01      Thu      00:06    -1.10
2020/10/01      Thu      00:12    -1.08
2020/10/01      Thu      00:18    -1.06
2020/10/01      Thu      00:24    -1.04
2020/10/01      Thu      00:30    -1.01
2020/10/01      Thu      00:36    -0.98
2020/10/01      Thu      00:42    -0.95
2020/10/01      Thu      00:48    -0.92
2020/10/01      Thu      00:54    -0.88
2020/10/01      Thu      01:00    -0.84
2020/10/01      Thu      01:06    -0.80
2020/10/01      Thu      01:12    -0.76
2020/10/01      Thu      01:18    -0.71
2020/10/01      Thu      01:24    -0.67
2020/10/01      Thu      01:30    -0.62
```

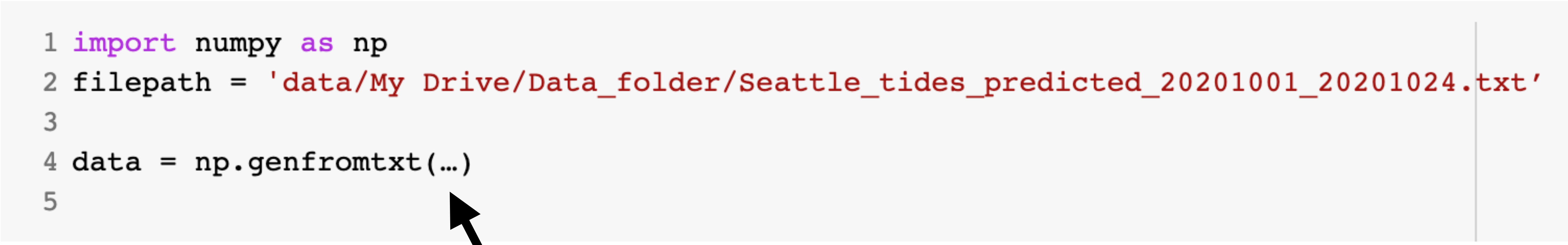
Here is what we know about our file now:

1) Our file path on the Google Drive
2) There are 14 lines of header information
   - Station, state, units, interval/frequency
3) Columns 0, 1, 2 are date information
4) Column 3 has floats
5) The columns are separated by white space

Extracting the data

Now that we know what the file structure is, we can load the data using the numpy function, **np.genfromtxt( )**

This function takes a file and puts its data elements into a numpy array. We have to carefully consider the file structure to properly load the data.



```python
import numpy as np
filepath = 'data/My Drive/Data_folder/Seattle_tides_predicted_20201001_20201024.txt'

data = np.genfromtxt(…)

```

We start building our arguments for loading our data.

Here is what we know about our file now:

1) Our file path on the Google Drive
2) There are 14 lines of header information
    - Station, state, units, interval/frequency
3) Columns 0, 1, 2 are date information
4) Column 3 has floats
5) The columns are separated by white space

np.genfromtxt( )

```
data = np.genfromtxt(…)
```

Here is what we know about our file now:

1) Our file path on the Google Drive
2) There are 14 lines of header information
   - Station, state, units, interval/frequency
3) Columns 0, 1, 2 are date information
4) Column 3 has floats
5) The columns are separated by white space

np.genfromtxt( )

```
data = np.genfromtxt(…)
```

filepath

Here is what we know about our file now:

1) Our file path on the Google Drive
2) There are 14 lines of header information
   - Station, state, units, interval/frequency
3) Columns 0, 1, 2 are date information
4) Column 3 has floats
5) The columns are separated by white space

np.genfromtxt( )

```
data = np.genfromtxt(…)
```

```
filepath
skip_header = 14
```

Here is what we know about our file now:

1) Our file path on the Google Drive
2) There are 14 lines of header information
   - Station, state, units, interval/frequency
3) Columns 0, 1, 2 are date information
4) Column 3 has floats
5) The columns are separated by white space

np.genfromtxt( )

```
data = np.genfromtxt(…)

filepath
skip_header = 14
usecols = 3
dtype = float
```

Here is what we know about our file now:

1) Our file path on the Google Drive

2) There are 14 lines of header information

   - Station, state, units, interval/frequency

3) Columns 0, 1, 2 are date information

4) Column 3 has floats

5) The columns are separated by white space

np.genfromtxt( )

```
data = np.genfromtxt(…)

        filepath
  skip_header = 14
     usecols = 3
    dtype = float
 (delimiter = None)
```

Here is what we know about our file now:

1) Our file path on the Google Drive

2) There are 14 lines of header information

   - Station, state, units, interval/frequency

3) Columns 0, 1, 2 are date information

4) Column 3 has floats

5) The columns are separated by white space

np.genfromtxt( )

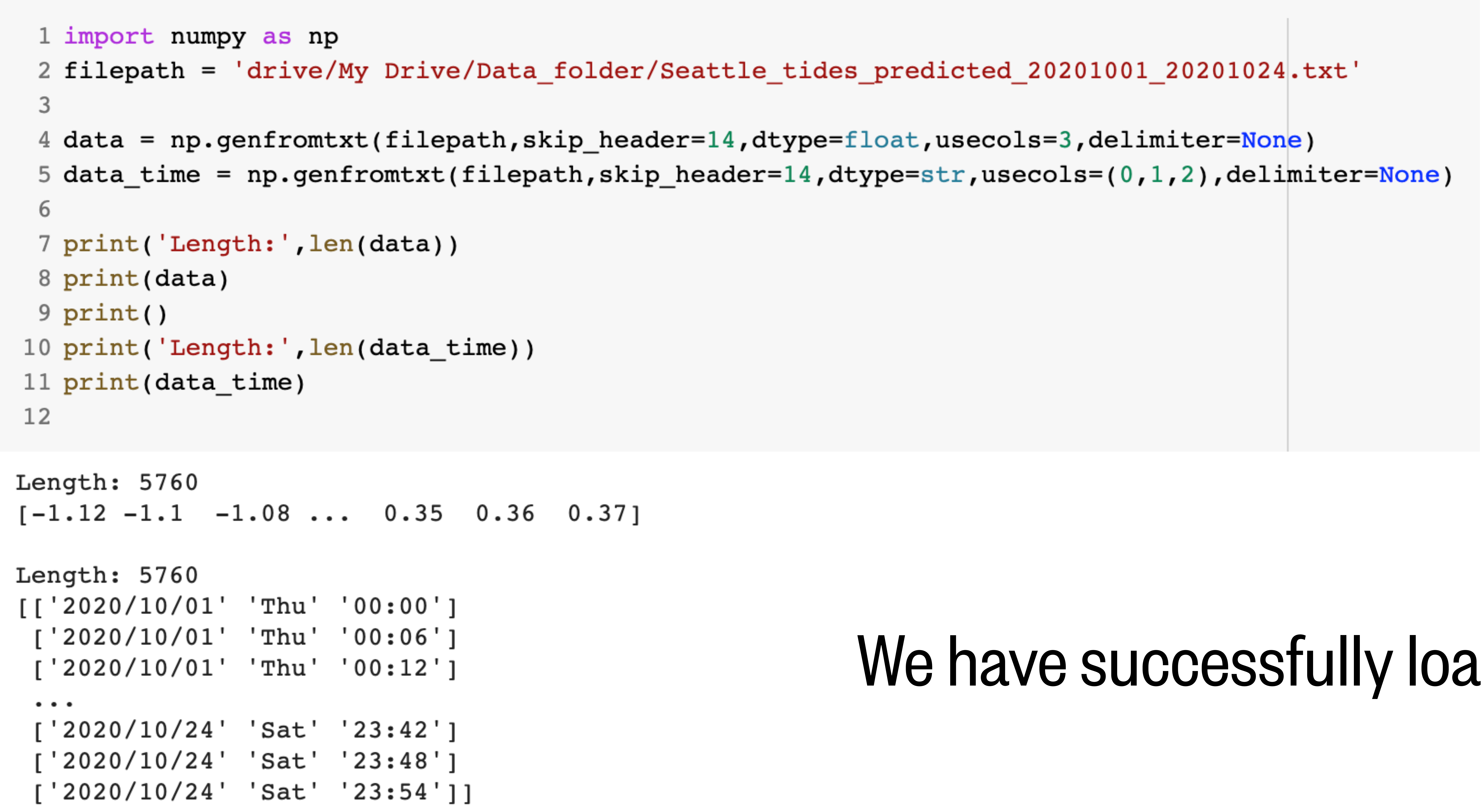
```
data_time = np.genfromtxt(…)
```

```
filepath
skip_header = 14
usecols = (0,1,2)
dtype = str
(delimiter = None)
```

Here is what we know about our file now:

1) Our file path on the Google Drive
2) There are 14 lines of header information
   - Station, state, units, interval/frequency
3) Columns 0, 1, 2 are date information
4) Column 3 has floats
5) The columns are separated by white space

np.genfromtxt( )



```python
import numpy as np
filepath = 'drive/My Drive/Data_folder/Seattle_tides_predicted_20201001_20201024.txt'

data = np.genfromtxt(filepath,skip_header=14,dtype=float,usecols=3,delimiter=None)
data_time = np.genfromtxt(filepath,skip_header=14,dtype=str,usecols=(0,1,2),delimiter=None)

print('Length:',len(data))
print(data)
print()
print('Length:',len(data_time))
print(data_time)

```

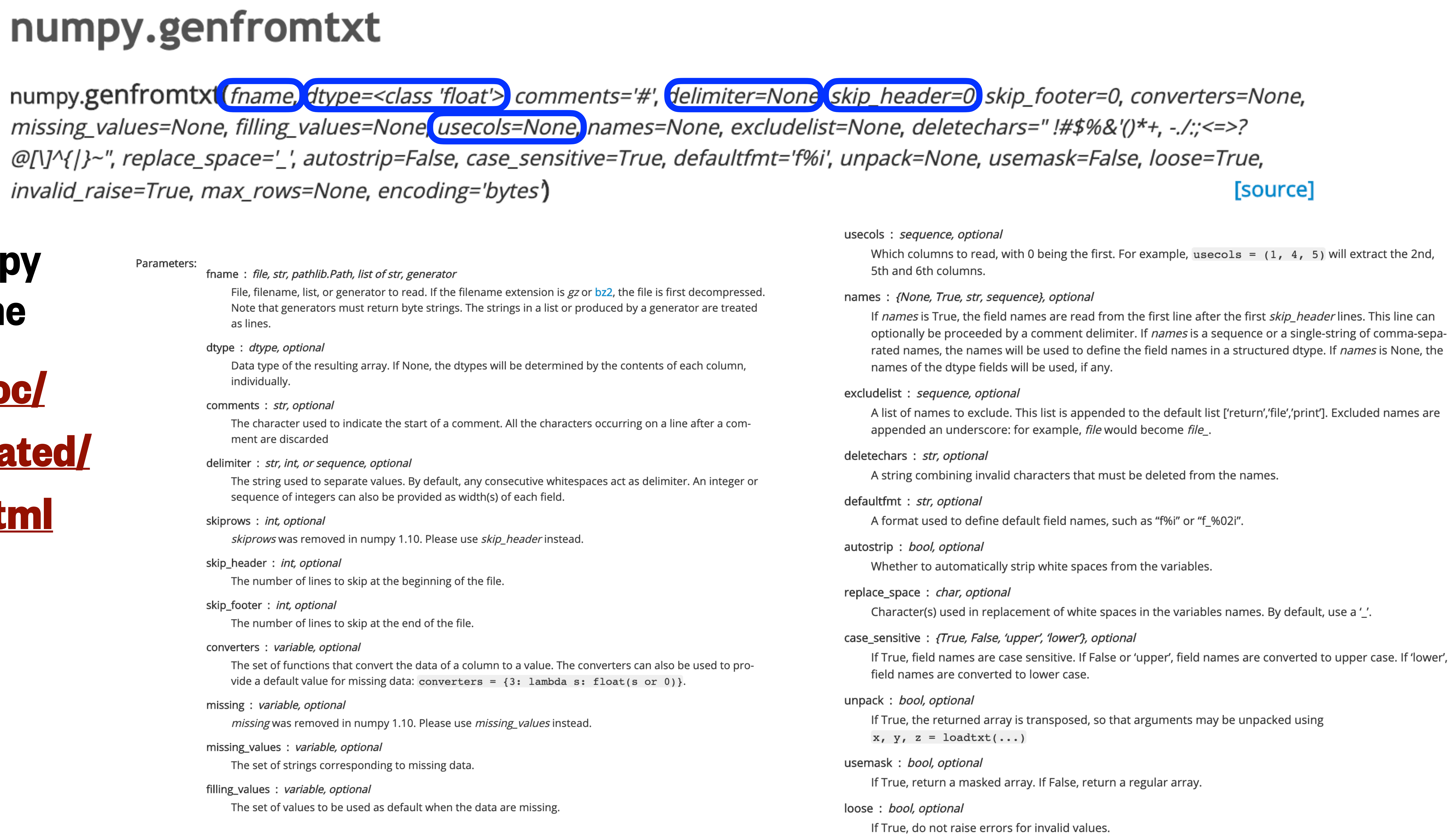
```
Length: 5760
[-1.12 -1.1  -1.08 ...  0.35  0.36  0.37]

Length: 5760
[['2020/10/01' 'Thu' '00:00']
 ['2020/10/01' 'Thu' '00:06']
 ['2020/10/01' 'Thu' '00:12']
 ...
 ['2020/10/24' 'Sat' '23:42']
 ['2020/10/24' 'Sat' '23:48']
 ['2020/10/24' 'Sat' '23:54']]
```

We have successfully loaded data!

Formatting function arguments

From the official numpy documentation online

https://numpy.org/doc/stable/reference/generated/numpy.genfromtxt.html



numpy.genfromtxt

numpy.genfromtxt(*fname*, *dtype=<class 'float'>*, *comments='#'*, *delimiter=None*, *skip_header=0*, *skip_footer=0*, *converters=None*, *missing_values=None*, *filling_values=None*, *usecols=None*, *names=None*, *excludelist=None*, *deletechars=" !#$%&'()*+, -./:;<=>?@[\]^{|}~"*, *replace_space='_'*, *autostrip=False*, *case_sensitive=True*, *defaultfmt='f%i'*, *unpack=None*, *usemask=False*, *loose=True*, *invalid_raise=True*, *max_rows=None*, *encoding='bytes'*) [source]

Parameters:

fname : *file, str, pathlib.Path, list of str, generator*

File, filename, list, or generator to read. If the filename extension is *gz* or bz2, the file is first decompressed. Note that generators must return byte strings. The strings in a list or produced by a generator are treated as lines.

dtype : *dtype, optional*

Data type of the resulting array. If None, the dtypes will be determined by the contents of each column, individually.

comments : *str, optional*

The character used to indicate the start of a comment. All the characters occurring on a line after a comment are discarded

delimiter : *str, int, or sequence, optional*

The string used to separate values. By default, any consecutive whitespaces act as delimiter. An integer or sequence of integers can also be provided as width(s) of each field.

skiprows : *int, optional*

skiprows was removed in numpy 1.10. Please use *skip_header* instead.

skip_header : *int, optional*

The number of lines to skip at the beginning of the file.

skip_footer : *int, optional*

The number of lines to skip at the end of the file.

converters : *variable, optional*

The set of functions that convert the data of a column to a value. The converters can also be used to provide a default value for missing data: `converters = {3: lambda s: float(s or 0)}`.

missing : *variable, optional*

missing was removed in numpy 1.10. Please use *missing_values* instead.

missing_values : *variable, optional*

The set of strings corresponding to missing data.

filling_values : *variable, optional*

The set of values to be used as default when the data are missing.

usecols : *sequence, optional*

Which columns to read, with 0 being the first. For example, `usecols = (1, 4, 5)` will extract the 2nd, 5th and 6th columns.

names : *{None, True, str, sequence}, optional*

If *names* is True, the field names are read from the first line after the first *skip_header* lines. This line can optionally be proceeded by a comment delimiter. If *names* is a sequence or a single-string of comma-separated names, the names will be used to define the field names in a structured dtype. If *names* is None, the names of the dtype fields will be used, if any.

excludelist : *sequence, optional*

A list of names to exclude. This list is appended to the default list ['return','file','print']. Excluded names are appended an underscore: for example, *file* would become *file_*.

deletechars : *str, optional*

A string combining invalid characters that must be deleted from the names.

defaultfmt : *str, optional*

A format used to define default field names, such as "f%i" or "f_%02i".

autostrip : *bool, optional*

Whether to automatically strip white spaces from the variables.

replace_space : *char, optional*

Character(s) used in replacement of white spaces in the variables names. By default, use a '_'.

case_sensitive : *{True, False, 'upper', 'lower'}, optional*

If True, field names are case sensitive. If False or 'upper', field names are converted to upper case. If 'lower', field names are converted to lower case.

unpack : *bool, optional*

If True, the returned array is transposed, so that arguments may be unpacked using `x, y, z = loadtxt(...)`

usemask : *bool, optional*

If True, return a masked array. If False, return a regular array.

loose : *bool, optional*

If True, do not raise errors for invalid values.

What we'll cover in this lesson

1. Loading and saving files to Google Colab

2. Loading data using readlines and numpy

3. **Intro to plotting**

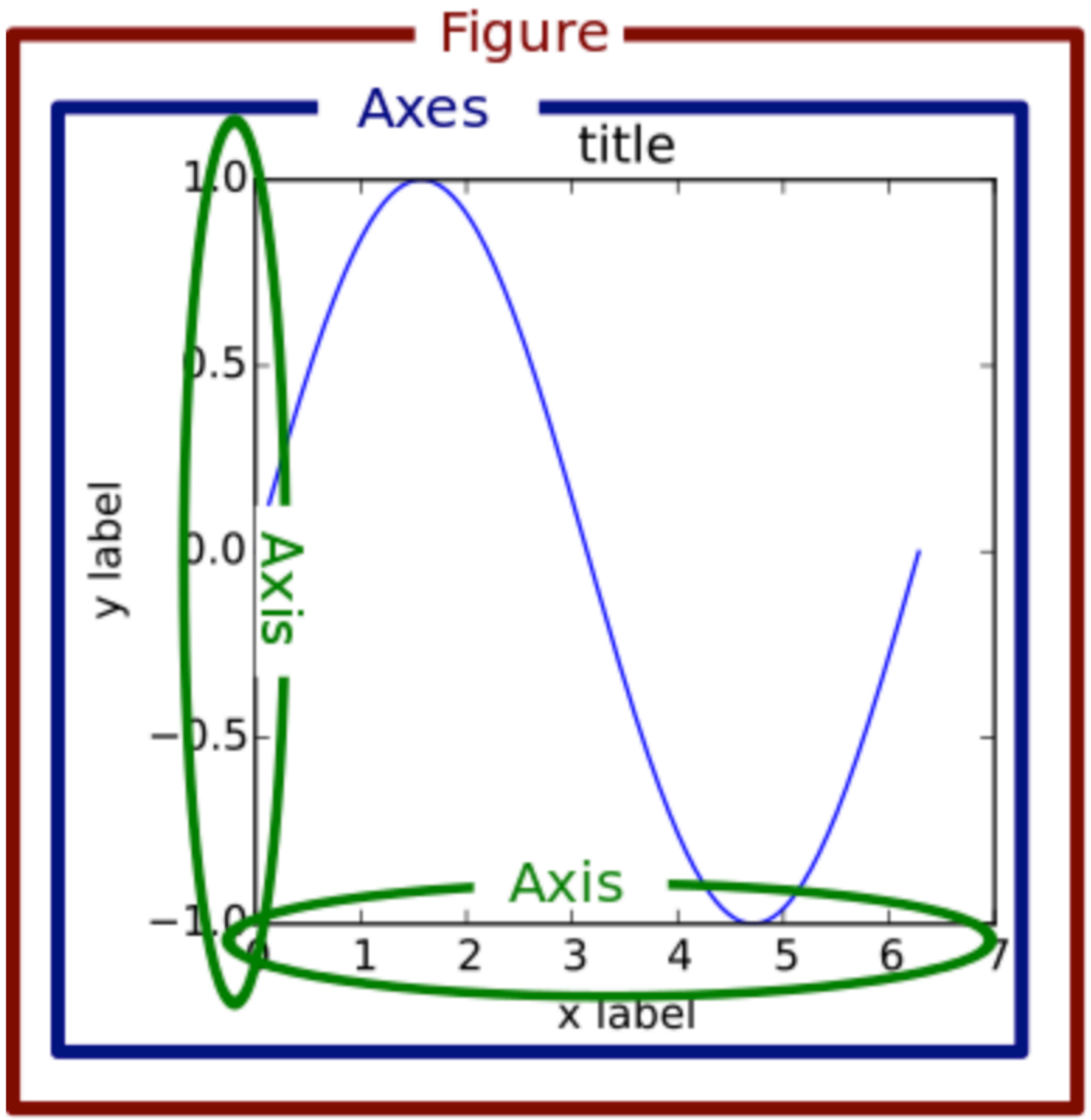
Importing matplotlib

This is a shortcut;
you can choose any name
but `plt` is most common

```python
import matplotlib.pyplot as plt
```

This part is
technically optional

Matplotlib objects



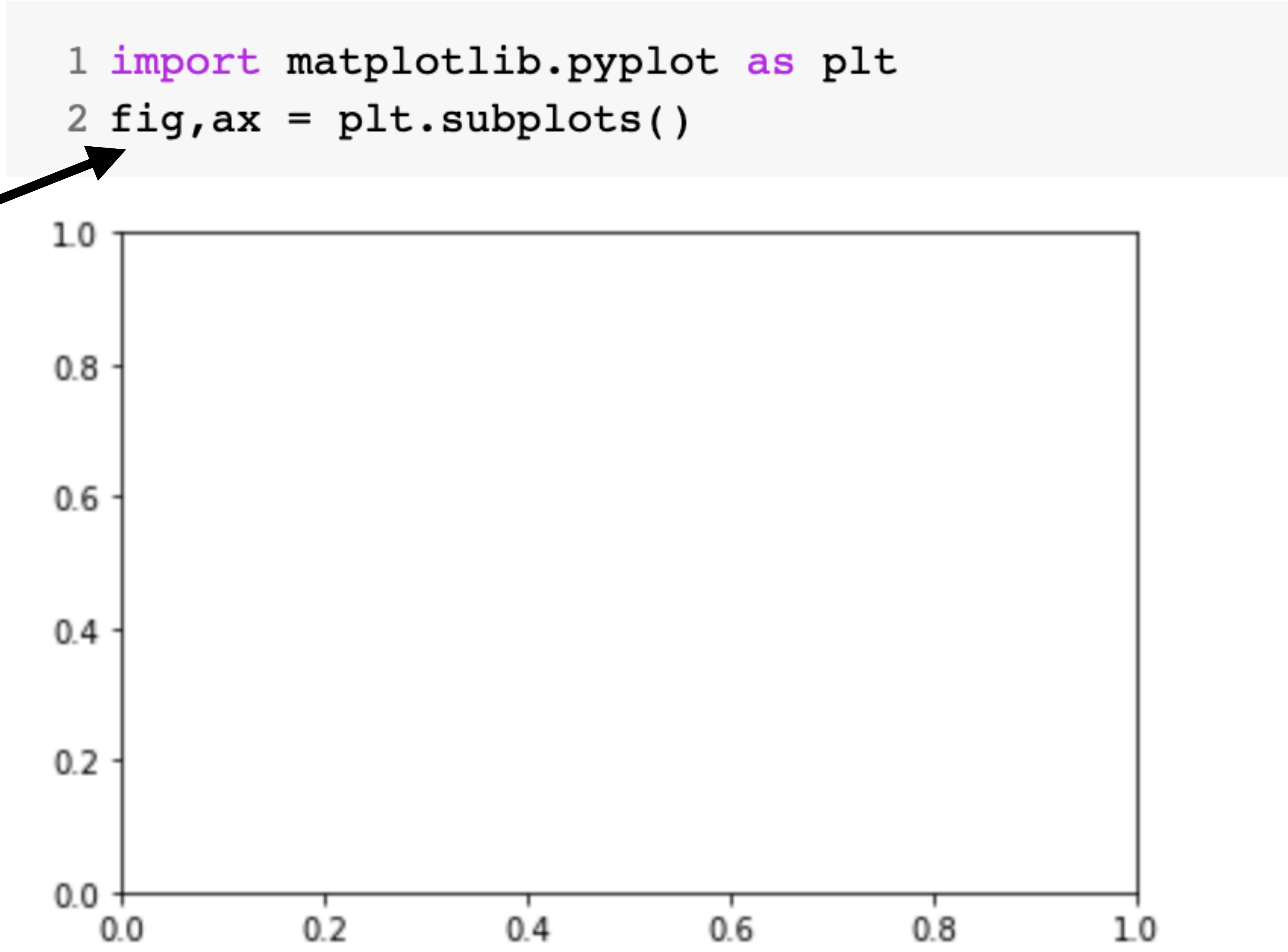
https://realpython.com/python-matplotlib-guide/

Main matplotlib objects:

1) Figure: this is outer container for plotting
2) Axes: this is an individual graph
3) Axis (and smaller...): these are the small formatting to refine your plot

Creating figures

Creating a figure with a blank axes object:



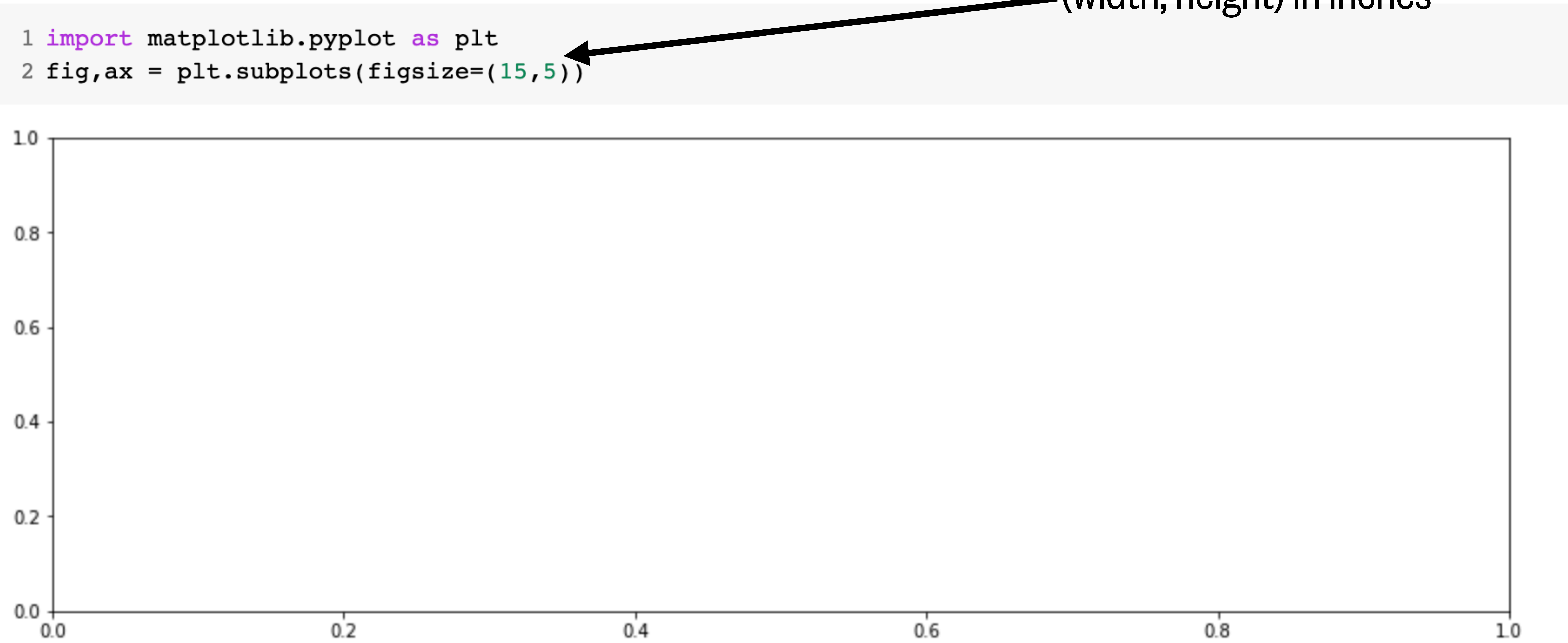
```
import matplotlib.pyplot as plt
fig,ax = plt.subplots()
```

These become the variable names for the figure and axes objects, respectively.

Creating figures

Creating a figure with a blank axes object of custom size:

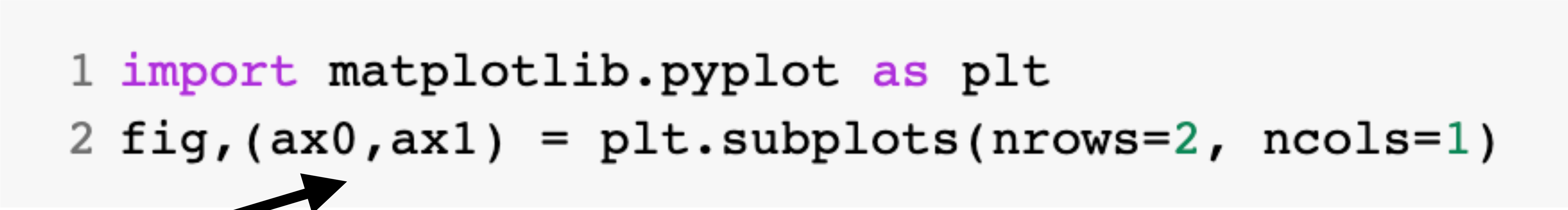
(width, height) in inches



```python
import matplotlib.pyplot as plt
fig,ax = plt.subplots(figsize=(15,5))
```

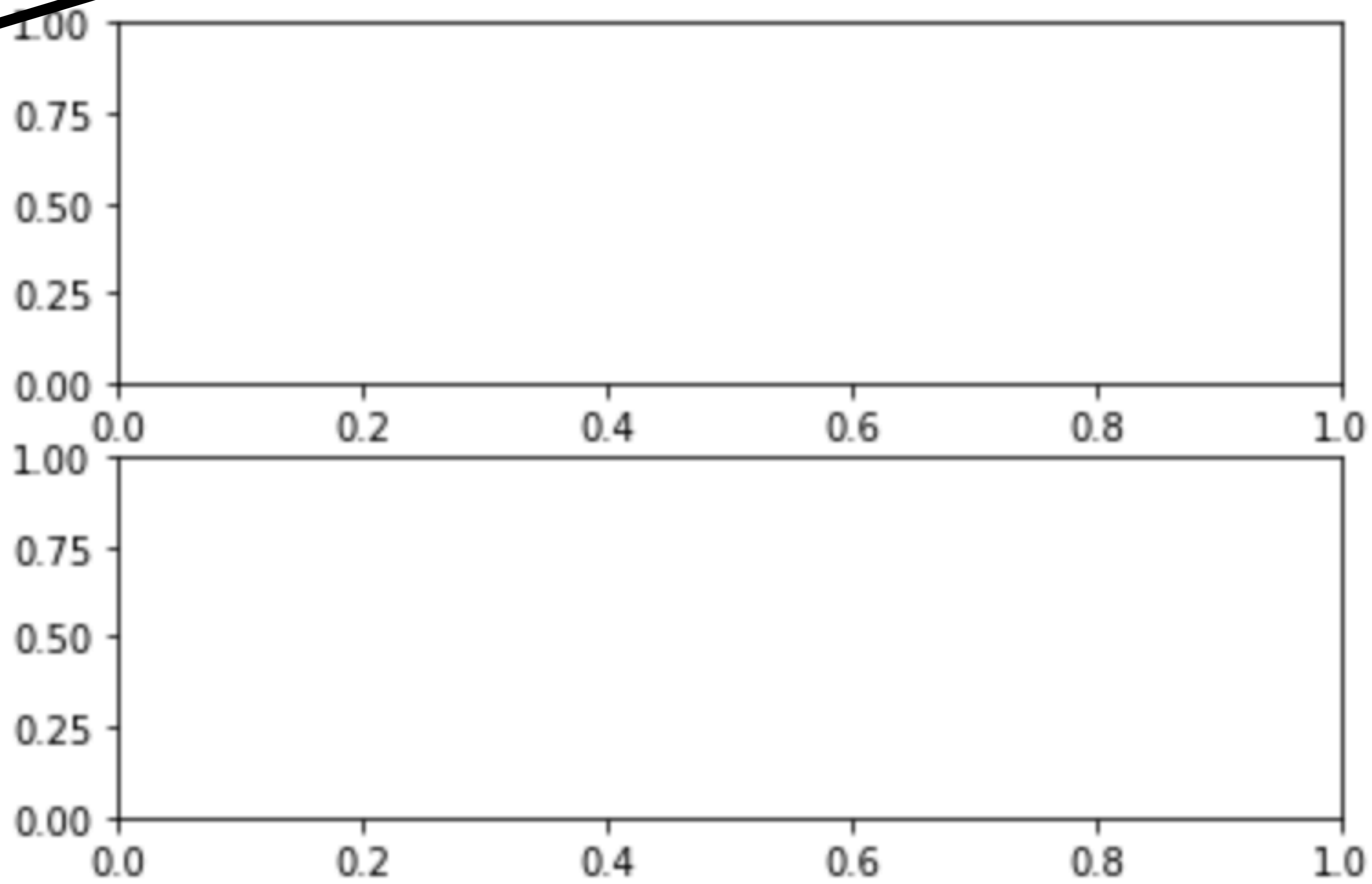
Creating figures

Creating a figure with multiple axes objects:



```python
import matplotlib.pyplot as plt
fig,(ax0,ax1) = plt.subplots(nrows=2, ncols=1)
```

This is so that each axes has a variable name

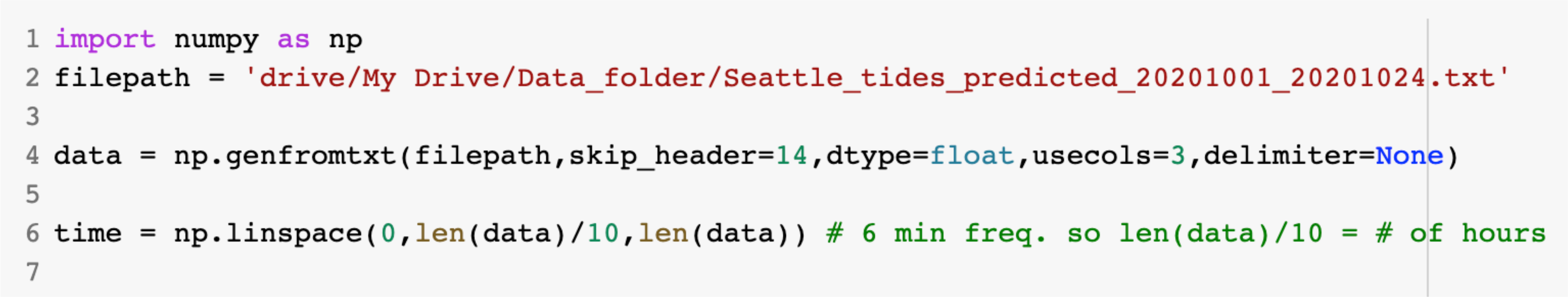


ax0

ax1

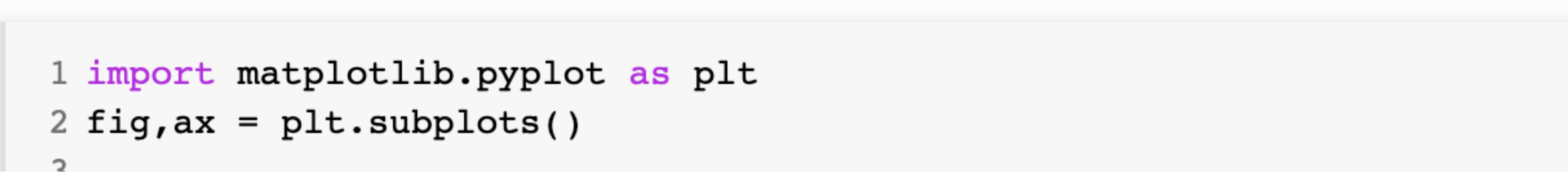
Simple line plot

Our data:

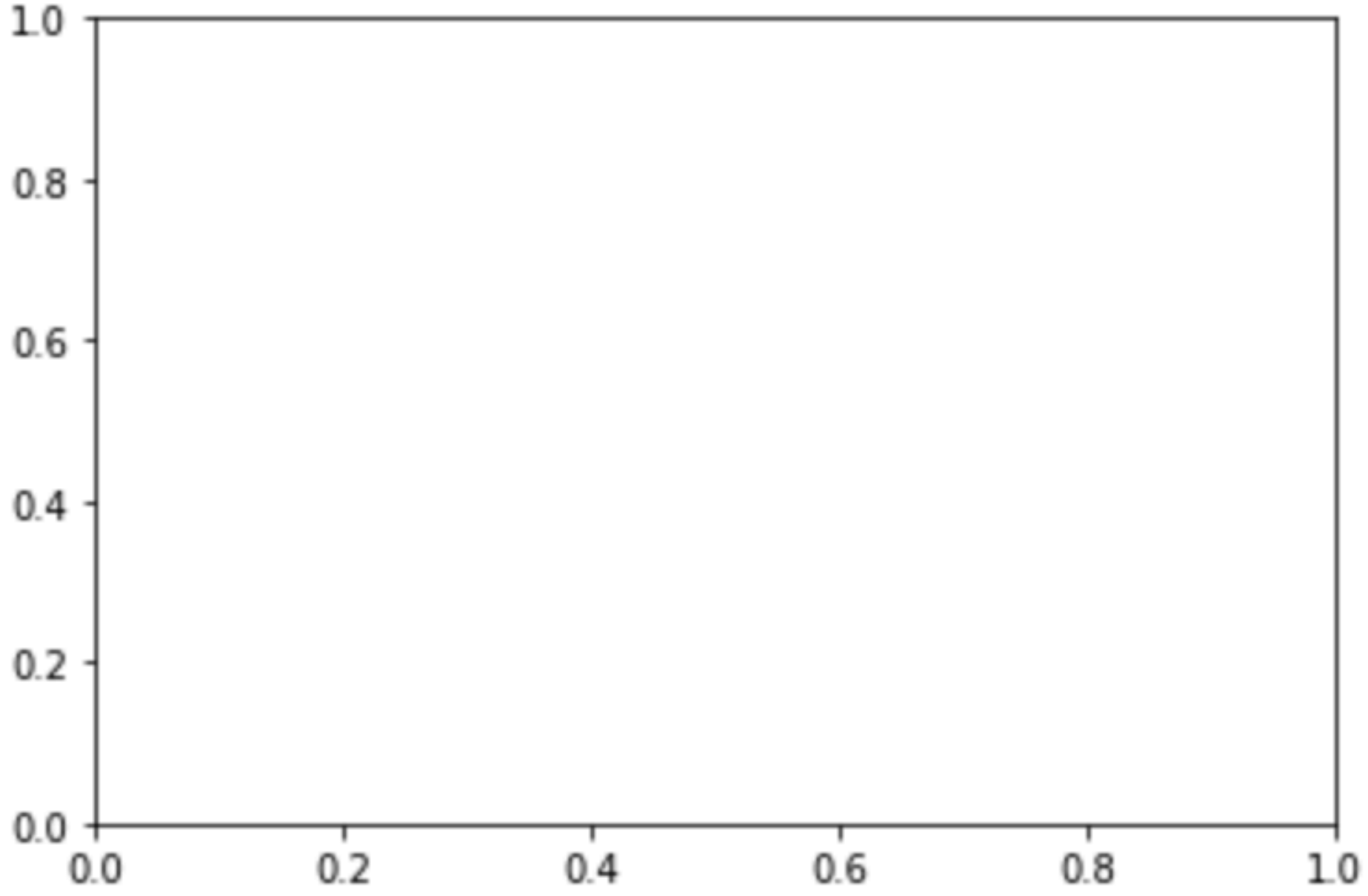


```python
import numpy as np
filepath = 'drive/My Drive/Data_folder/Seattle_tides_predicted_20201001_20201024.txt'

data = np.genfromtxt(filepath,skip_header=14,dtype=float,usecols=3,delimiter=None)

time = np.linspace(0,len(data)/10,len(data)) # 6 min freq. so len(data)/10 = # of hours

```

Start by creating a figure with an empty axes object:

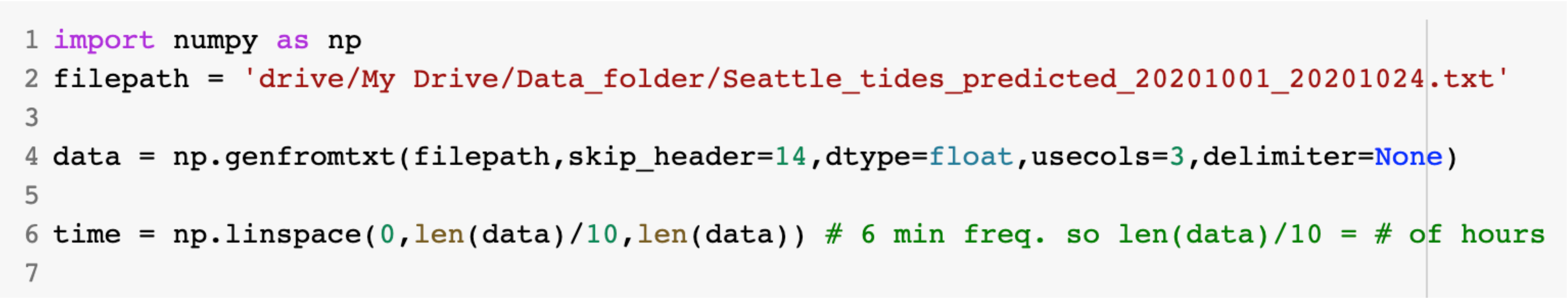


```python
import matplotlib.pyplot as plt
fig,ax = plt.subplots()
```



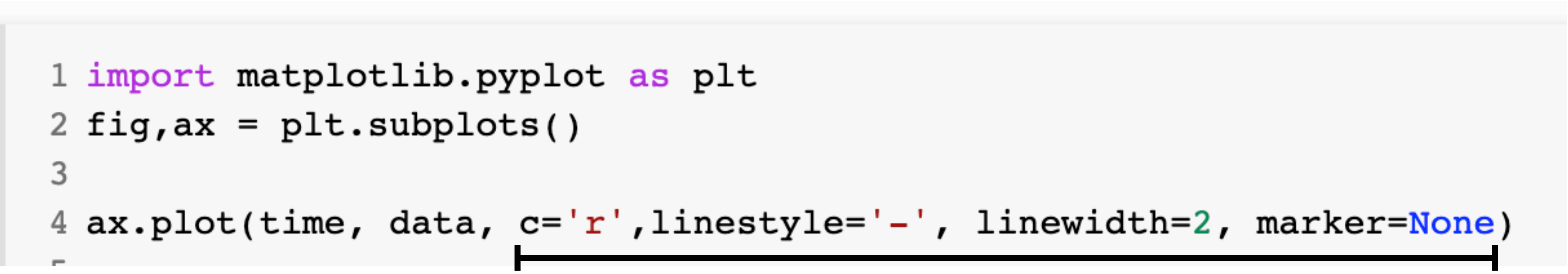
Simple line plot

Our data:



```
import numpy as np
filepath = 'drive/My Drive/Data_folder/Seattle_tides_predicted_20201001_20201024.txt'

data = np.genfromtxt(filepath,skip_header=14,dtype=float,usecols=3,delimiter=None)

time = np.linspace(0,len(data)/10,len(data)) # 6 min freq. so len(data)/10 = # of hours

```

Plot our data on the axis object:

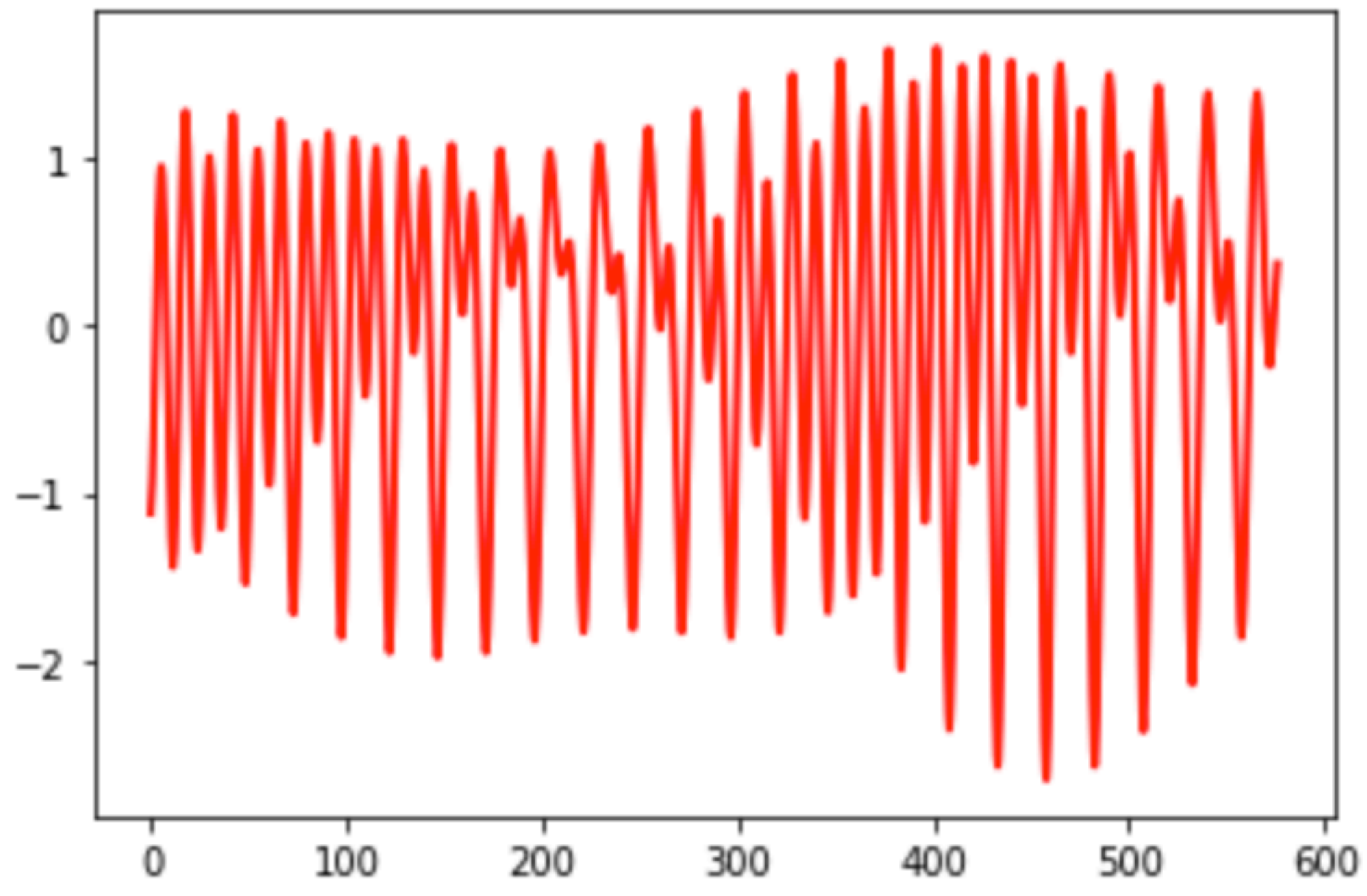


```
import matplotlib.pyplot as plt
fig,ax = plt.subplots()

ax.plot(time, data, c='r',linestyle='-', linewidth=2, marker=None)
```

x-axis, y-axis

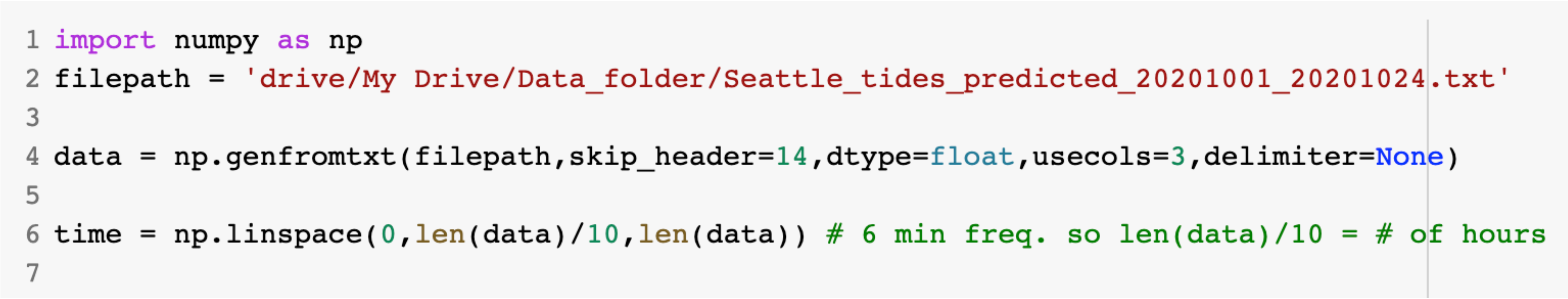
(c=color)

These are optional arguments, but they make the figure more appealing.



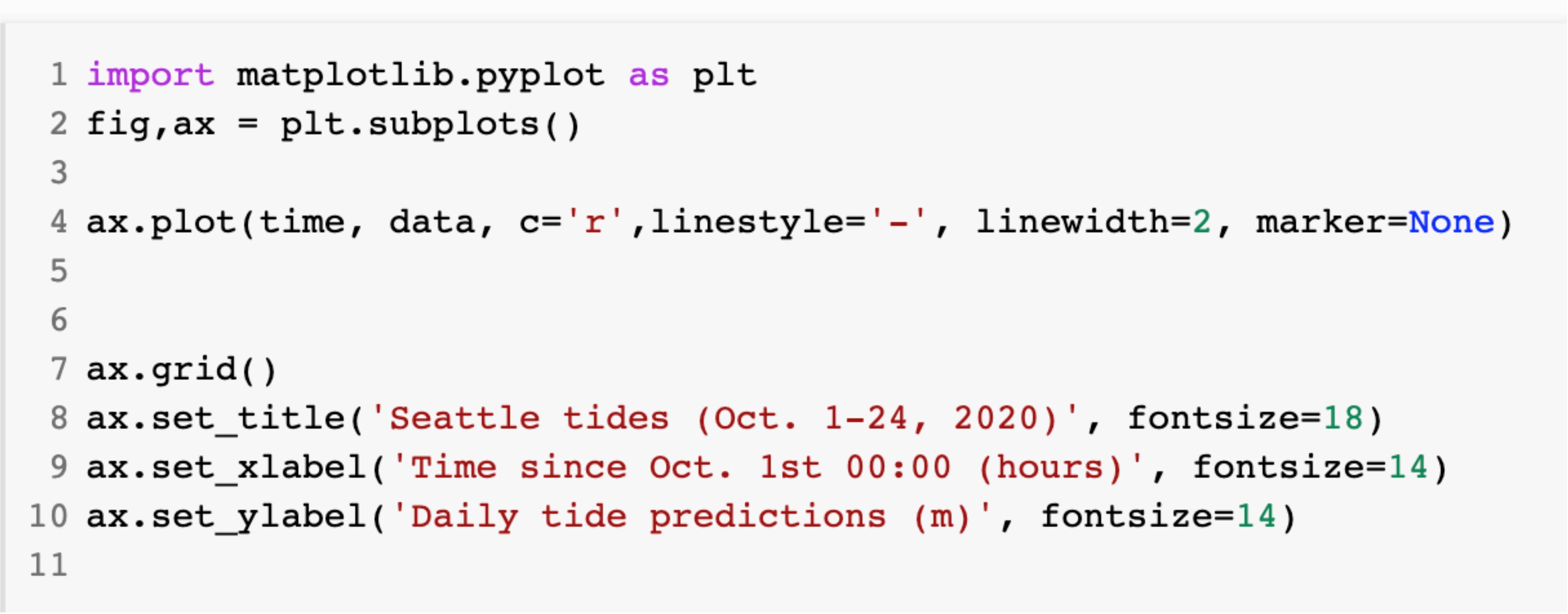
Simple line plot

Our data:

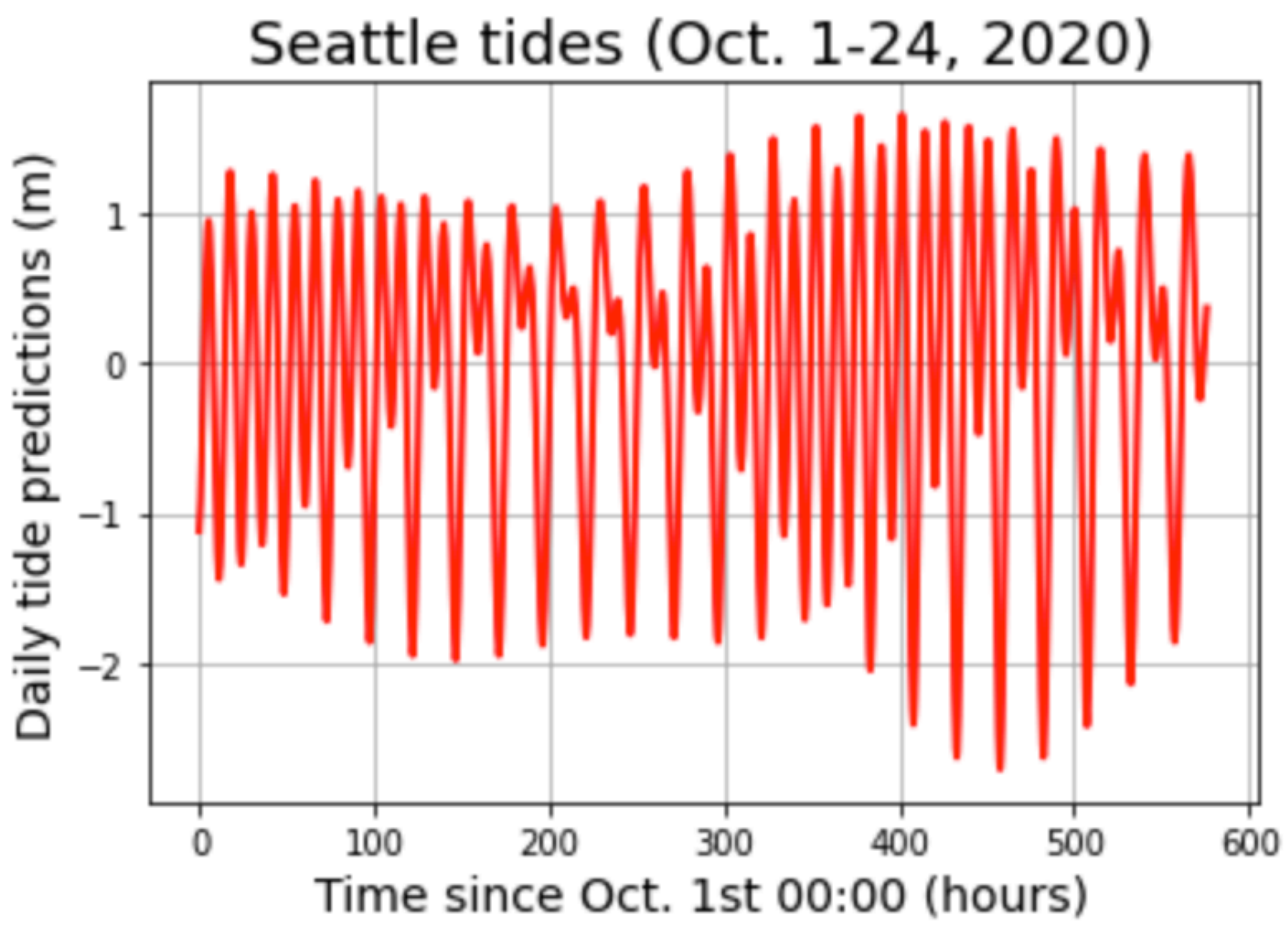


```python
import numpy as np
filepath = 'drive/My Drive/Data_folder/Seattle_tides_predicted_20201001_20201024.txt'

data = np.genfromtxt(filepath,skip_header=14,dtype=float,usecols=3,delimiter=None)

time = np.linspace(0,len(data)/10,len(data)) # 6 min freq. so len(data)/10 = # of hours

```

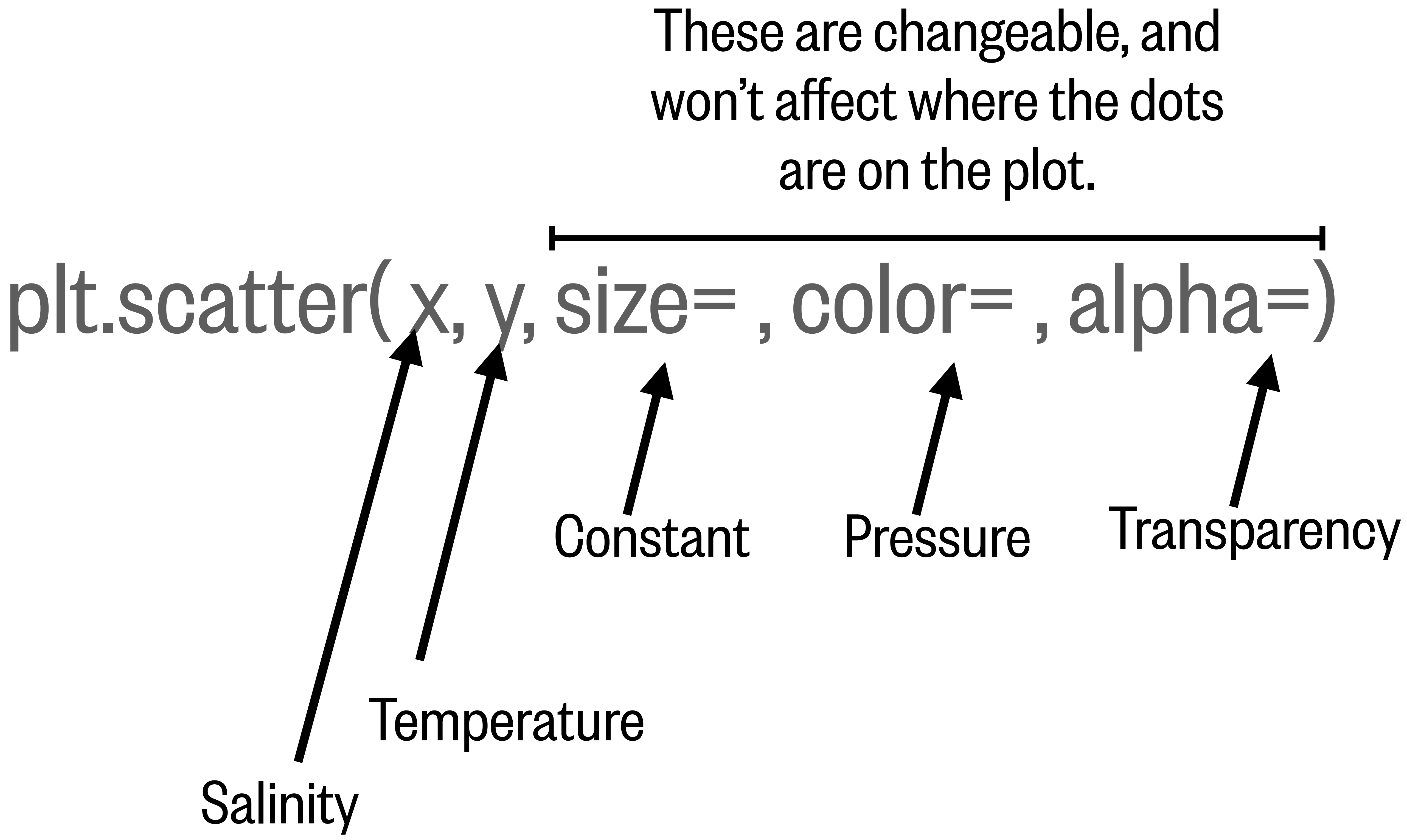
Create a title, labels, and figure formatting:



```python
import matplotlib.pyplot as plt
fig,ax = plt.subplots()

ax.plot(time, data, c='r',linestyle='-', linewidth=2, marker=None)


ax.grid()
ax.set_title('Seattle tides (Oct. 1-24, 2020)', fontsize=18)
ax.set_xlabel('Time since Oct. 1st 00:00 (hours)', fontsize=14)
ax.set_ylabel('Daily tide predictions (m)', fontsize=14)

```



Scatter plot



These are changeable, and won’t affect where the dots are on the plot.

```
plt.scatter( x, y, size= , color= , alpha=)
```

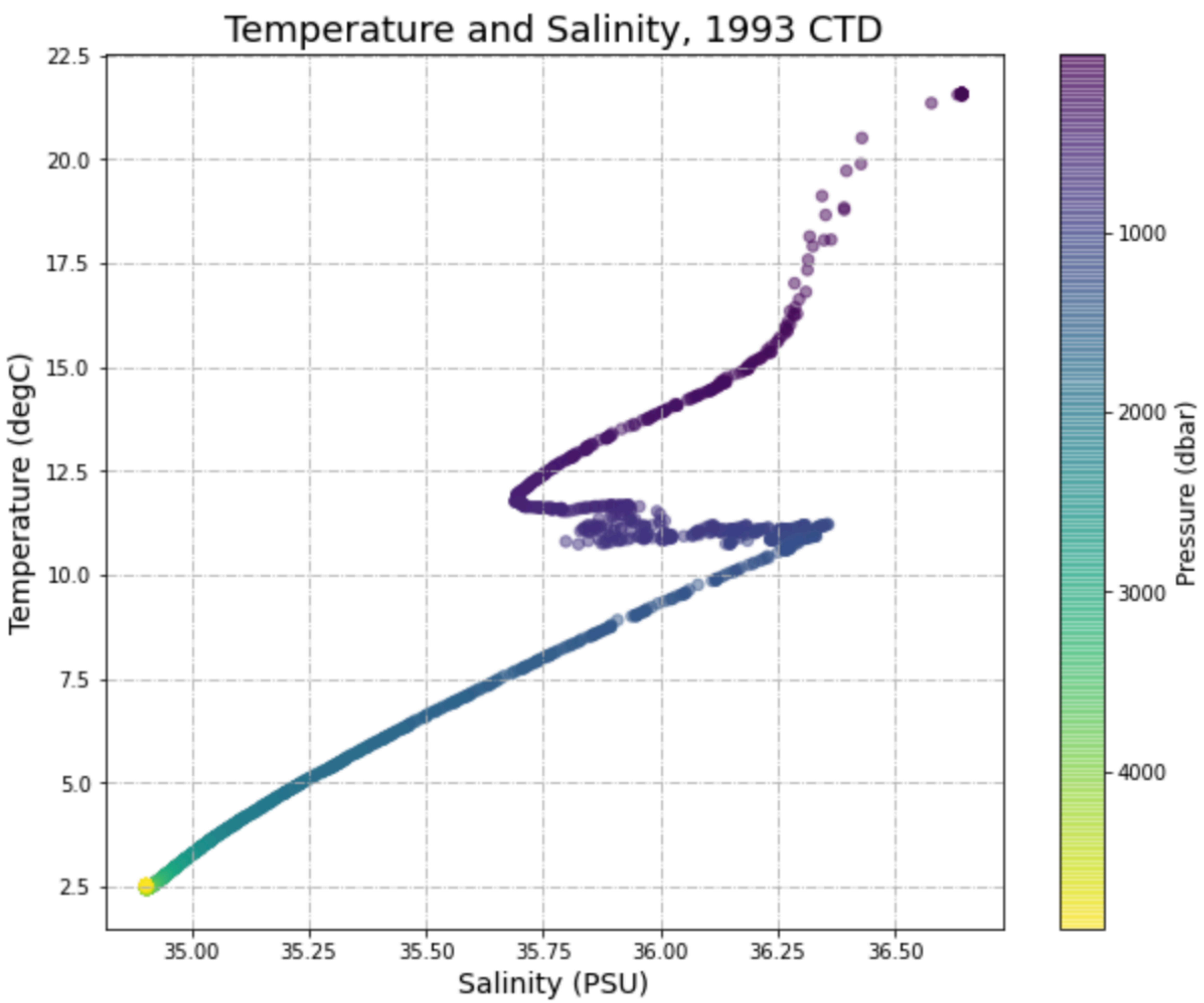
Salinity

Temperature

Constant

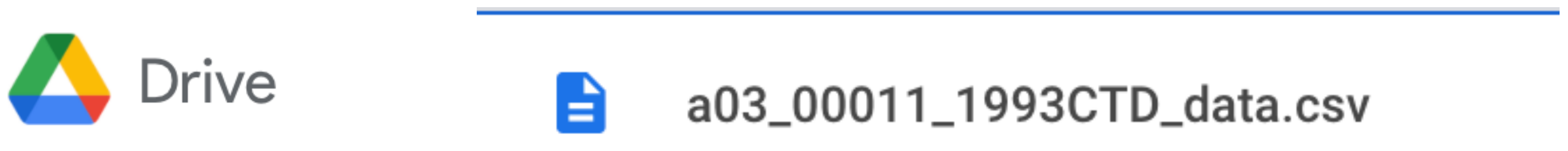
Pressure

Transparency

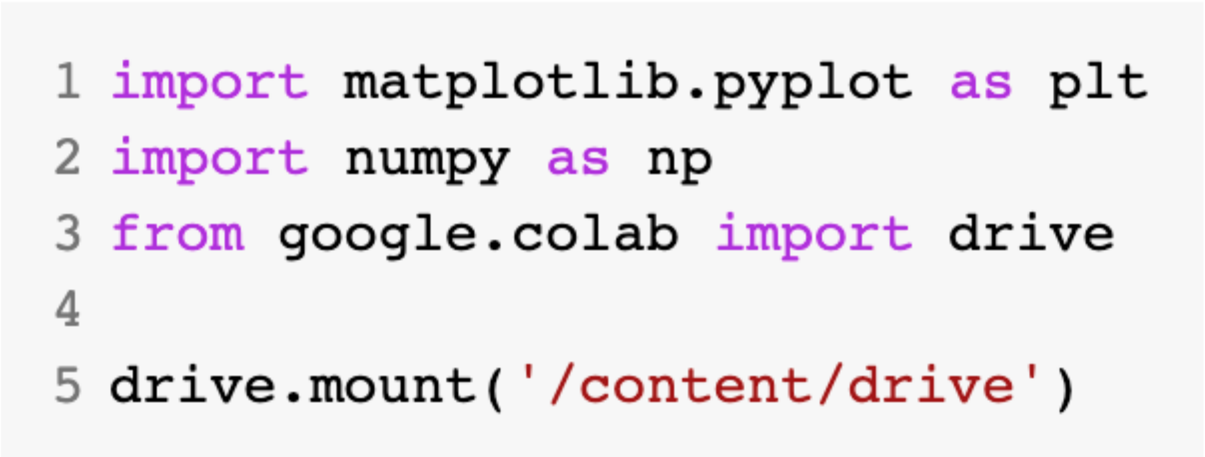


Scatter plot

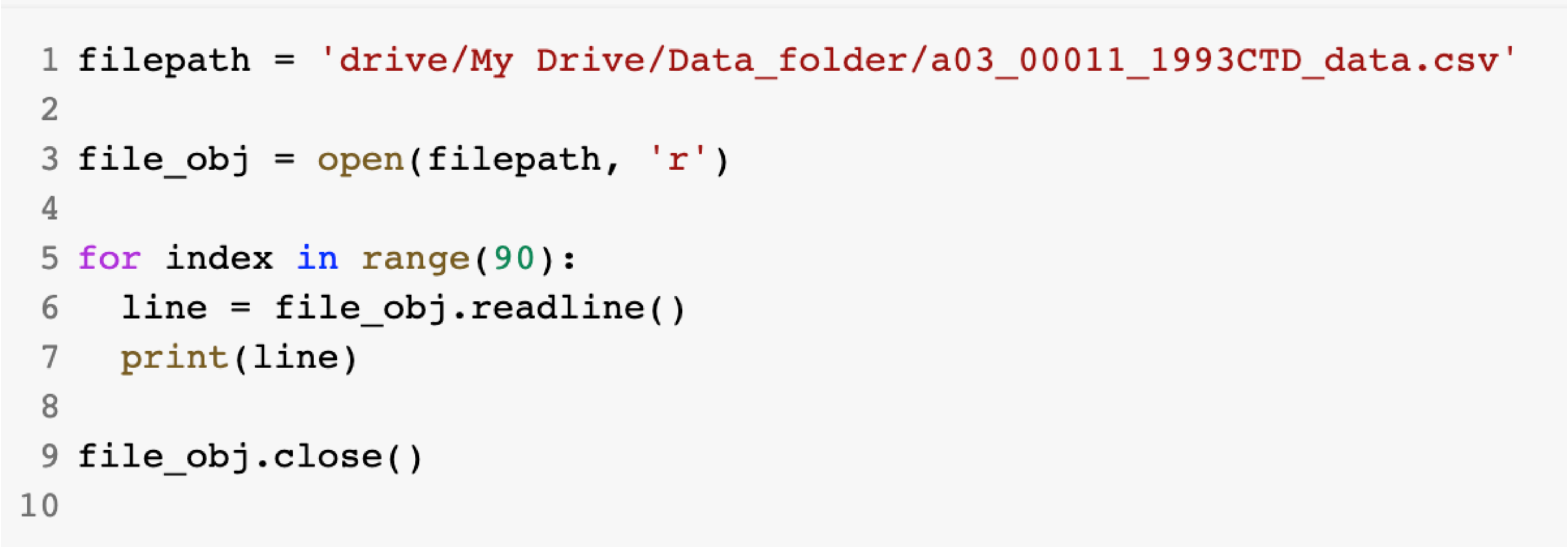
Example data: CTD data from 1993 WOCE



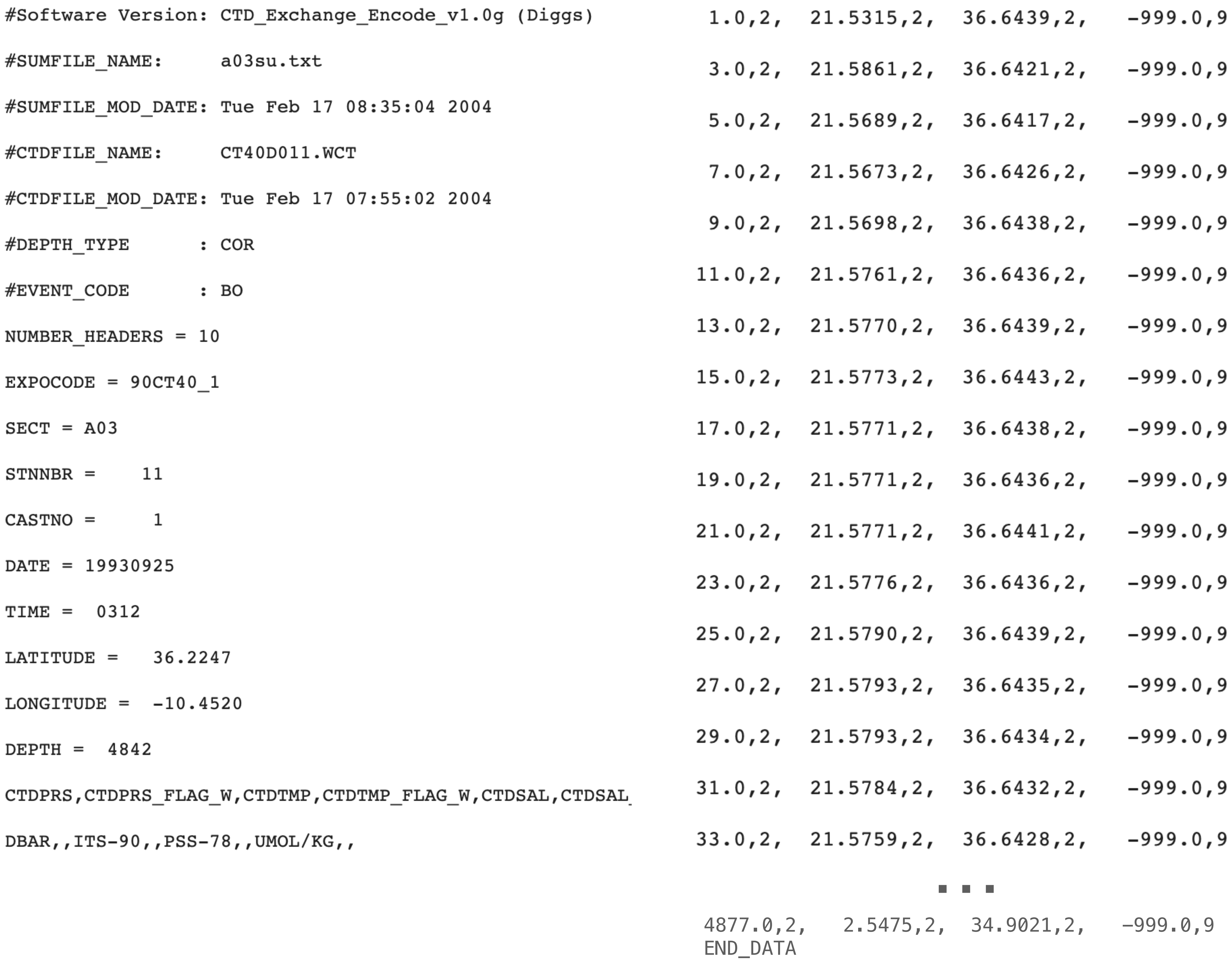
a03_00011_1993CTD_data.csv



```python
import matplotlib.pyplot as plt
import numpy as np
from google.colab import drive

drive.mount('/content/drive')
```



```python
filepath = 'drive/My Drive/Data_folder/a03_00011_1993CTD_data.csv'

file_obj = open(filepath, 'r')

for index in range(90):
  line = file_obj.readline()
  print(line)

file_obj.close()

```

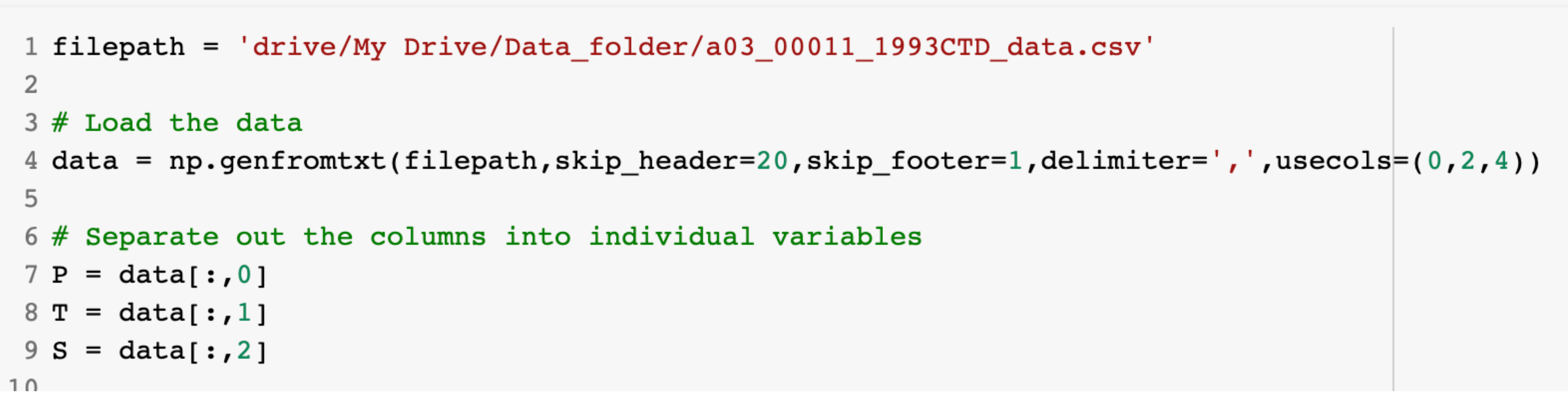


```
#Software Version: CTD_Exchange_Encode_v1.0g (Diggs)
#SUMFILE_NAME:     a03su.txt
#SUMFILE_MOD_DATE: Tue Feb 17 08:35:04 2004
#CTDFILE_NAME:     CT40D011.WCT
#CTDFILE_MOD_DATE: Tue Feb 17 07:55:02 2004
#DEPTH_TYPE      : COR
#EVENT_CODE      : BO
NUMBER_HEADERS = 10
EXPOCODE = 90CT40_1
SECT = A03
STNNBR =     11
CASTNO =      1
DATE = 19930925
TIME =  0312
LATITUDE =   36.2247
LONGITUDE =  -10.4520
DEPTH =   4842
CTDPRS,CTDPRS_FLAG_W,CTDTMP,CTDTMP_FLAG_W,CTDSAL,CTDSAL
DBAR,,ITS-90,,PSS-78,,UMOL/KG,,
```

```
  1.0,2,  21.5315,2,  36.6439,2,   -999.0,9
  3.0,2,  21.5861,2,  36.6421,2,   -999.0,9
  5.0,2,  21.5689,2,  36.6417,2,   -999.0,9
  7.0,2,  21.5673,2,  36.6426,2,   -999.0,9
  9.0,2,  21.5698,2,  36.6438,2,   -999.0,9
 11.0,2,  21.5761,2,  36.6436,2,   -999.0,9
 13.0,2,  21.5770,2,  36.6439,2,   -999.0,9
 15.0,2,  21.5773,2,  36.6443,2,   -999.0,9
 17.0,2,  21.5771,2,  36.6438,2,   -999.0,9
 19.0,2,  21.5771,2,  36.6436,2,   -999.0,9
 21.0,2,  21.5771,2,  36.6441,2,   -999.0,9
 23.0,2,  21.5776,2,  36.6436,2,   -999.0,9
 25.0,2,  21.5790,2,  36.6439,2,   -999.0,9
 27.0,2,  21.5793,2,  36.6435,2,   -999.0,9
 29.0,2,  21.5793,2,  36.6434,2,   -999.0,9
 31.0,2,  21.5784,2,  36.6432,2,   -999.0,9
 33.0,2,  21.5759,2,  36.6428,2,   -999.0,9
                    ...
4877.0,2,   2.5475,2,  34.9021,2,   -999.0,9
END_DATA
```

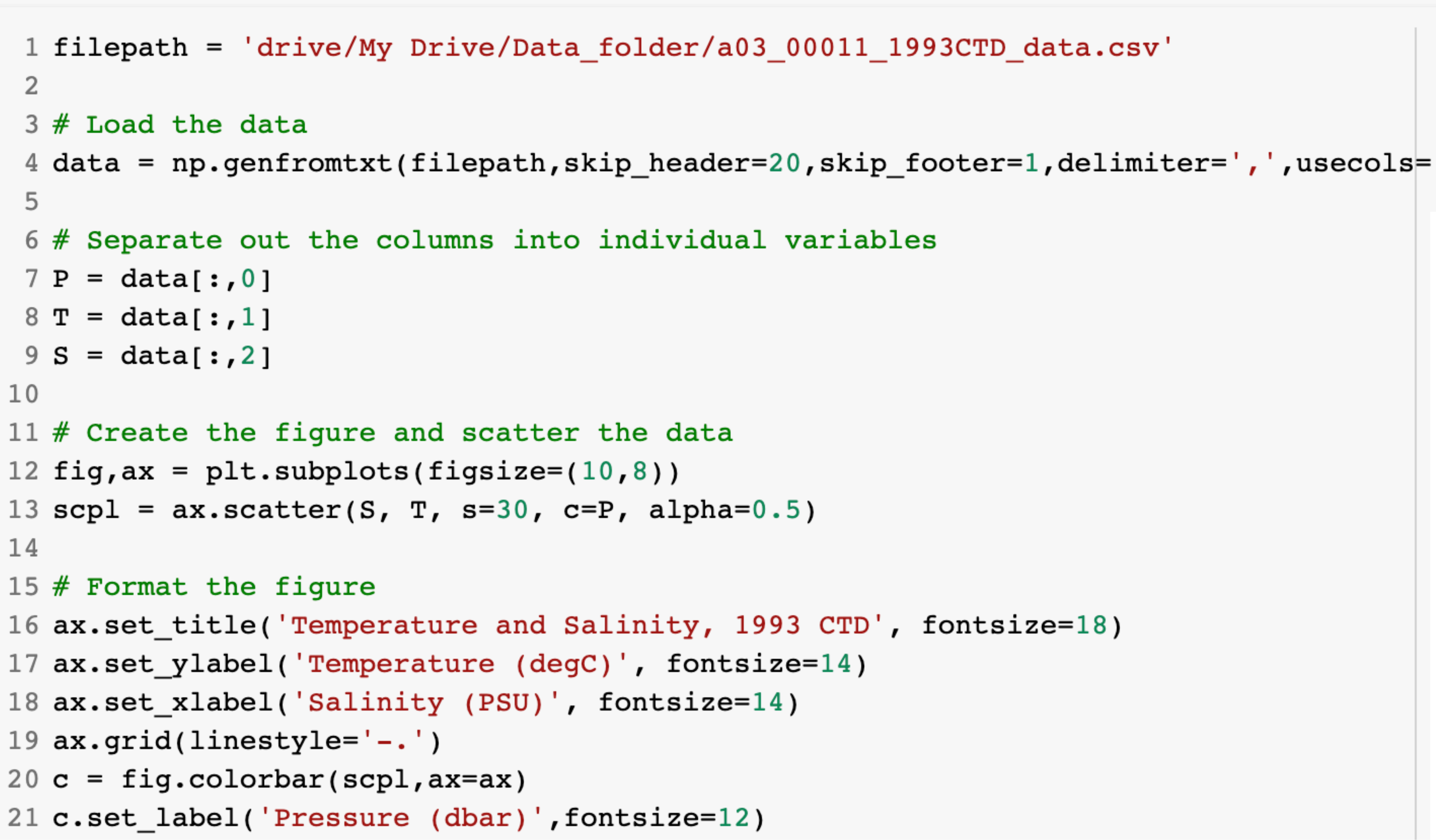
Scatter plot

Loading data:



```
filepath = 'drive/My Drive/Data_folder/a03_00011_1993CTD_data.csv'

# Load the data
data = np.genfromtxt(filepath,skip_header=20,skip_footer=1,delimiter=',',usecols=(0,2,4))

# Separate out the columns into individual variables
P = data[:,0]
T = data[:,1]
S = data[:,2]
```

Plotting:



```python
filepath = 'drive/My Drive/Data_folder/a03_00011_1993CTD_data.csv'

# Load the data
data = np.genfromtxt(filepath,skip_header=20,skip_footer=1,delimiter=',',usecols=(0,2,4))

# Separate out the columns into individual variables
P = data[:,0]
T = data[:,1]
S = data[:,2]

# Create the figure and scatter the data
fig,ax = plt.subplots(figsize=(10,8))
scpl = ax.scatter(S, T, s=30, c=P, alpha=0.5)

# Format the figure
ax.set_title('Temperature and Salinity, 1993 CTD', fontsize=18)
ax.set_ylabel('Temperature (degC)', fontsize=14)
ax.set_xlabel('Salinity (PSU)', fontsize=14)
ax.grid(linestyle='-.')
c = fig.colorbar(scpl,ax=ax)
c.set_label('Pressure (dbar)',fontsize=12)
```

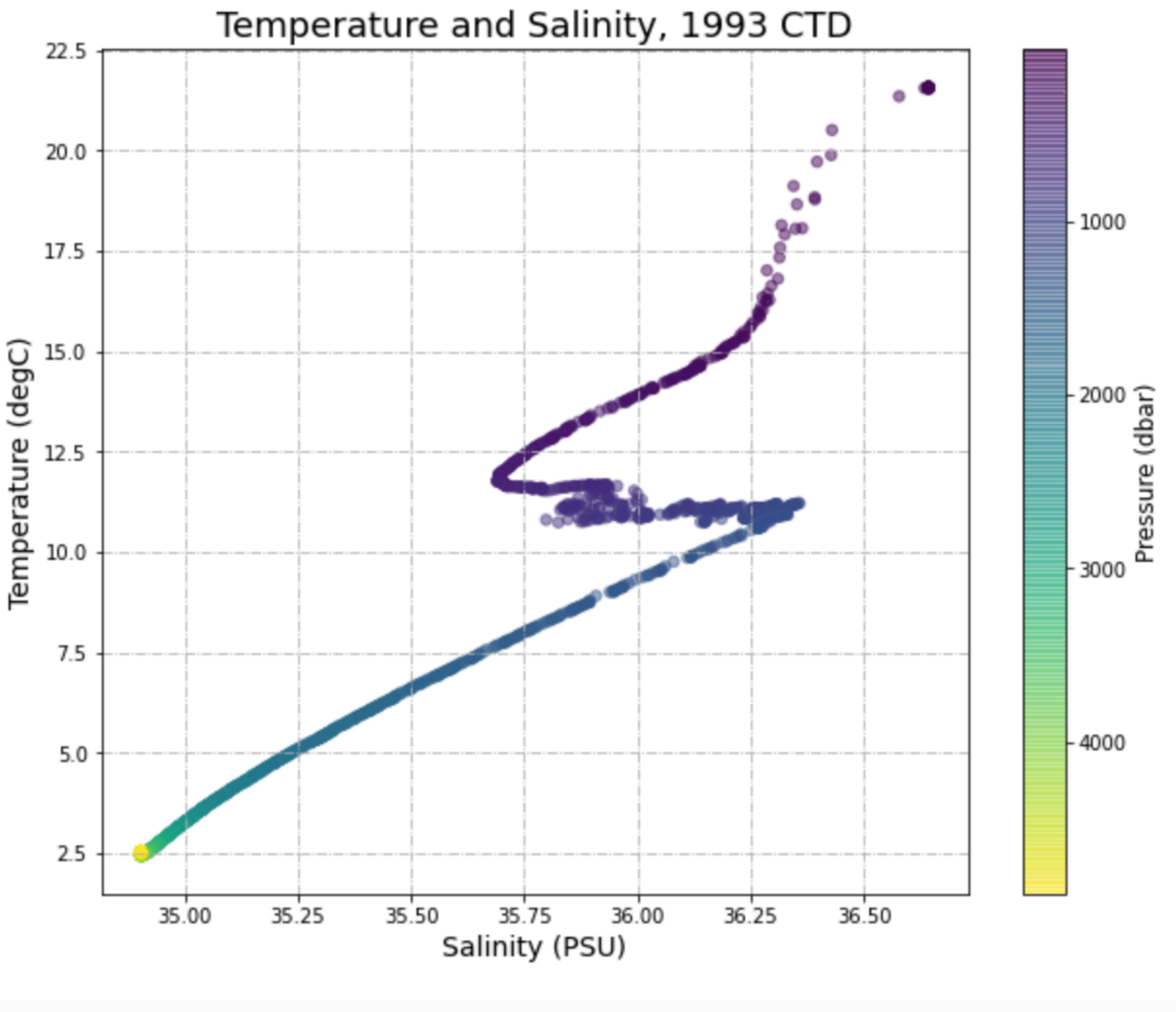


Figure requirements for this course

1) Title

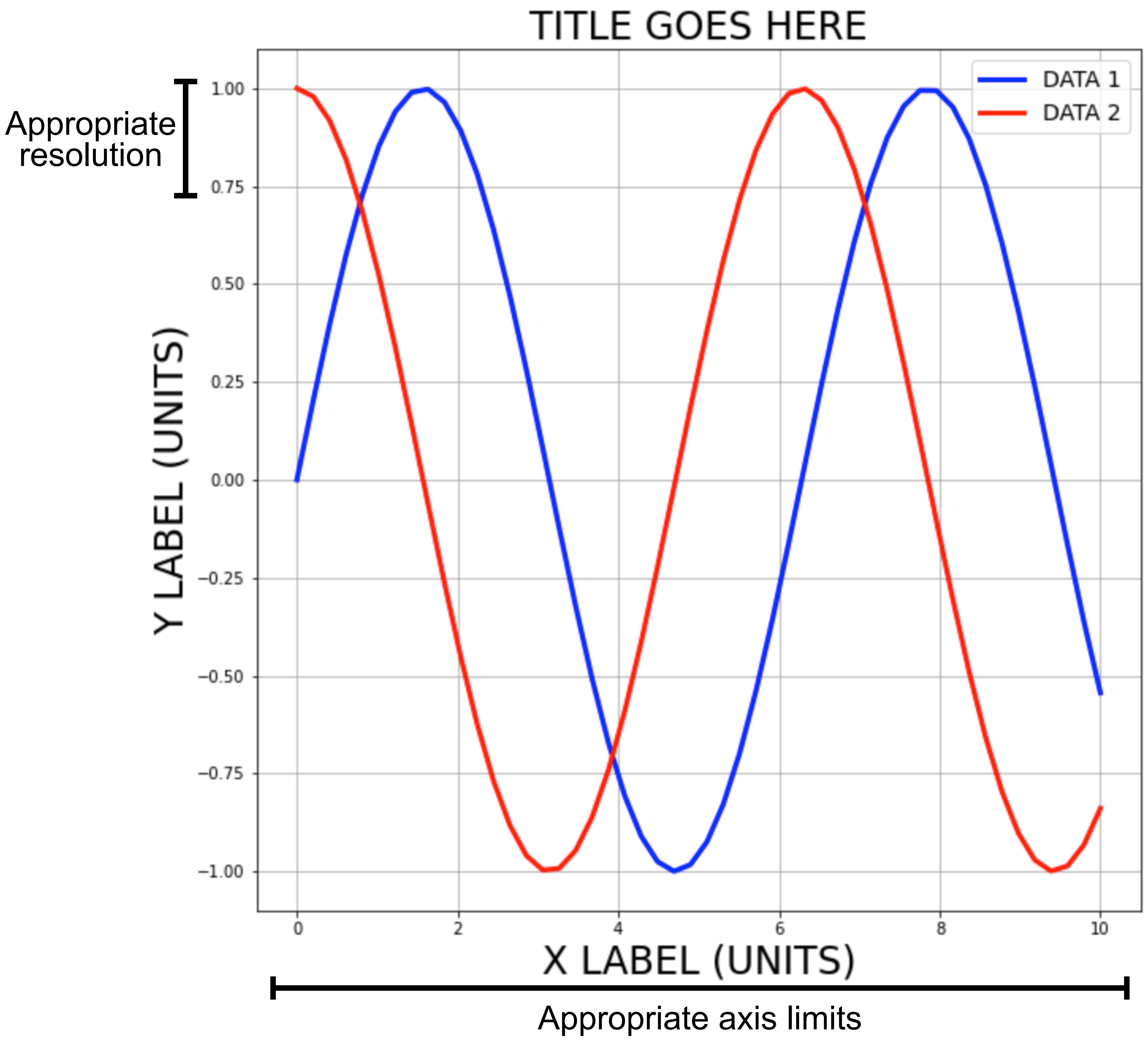
2) Axis labels (with units, when possible)

3) Appropriate axis limits (e.g. max/min)

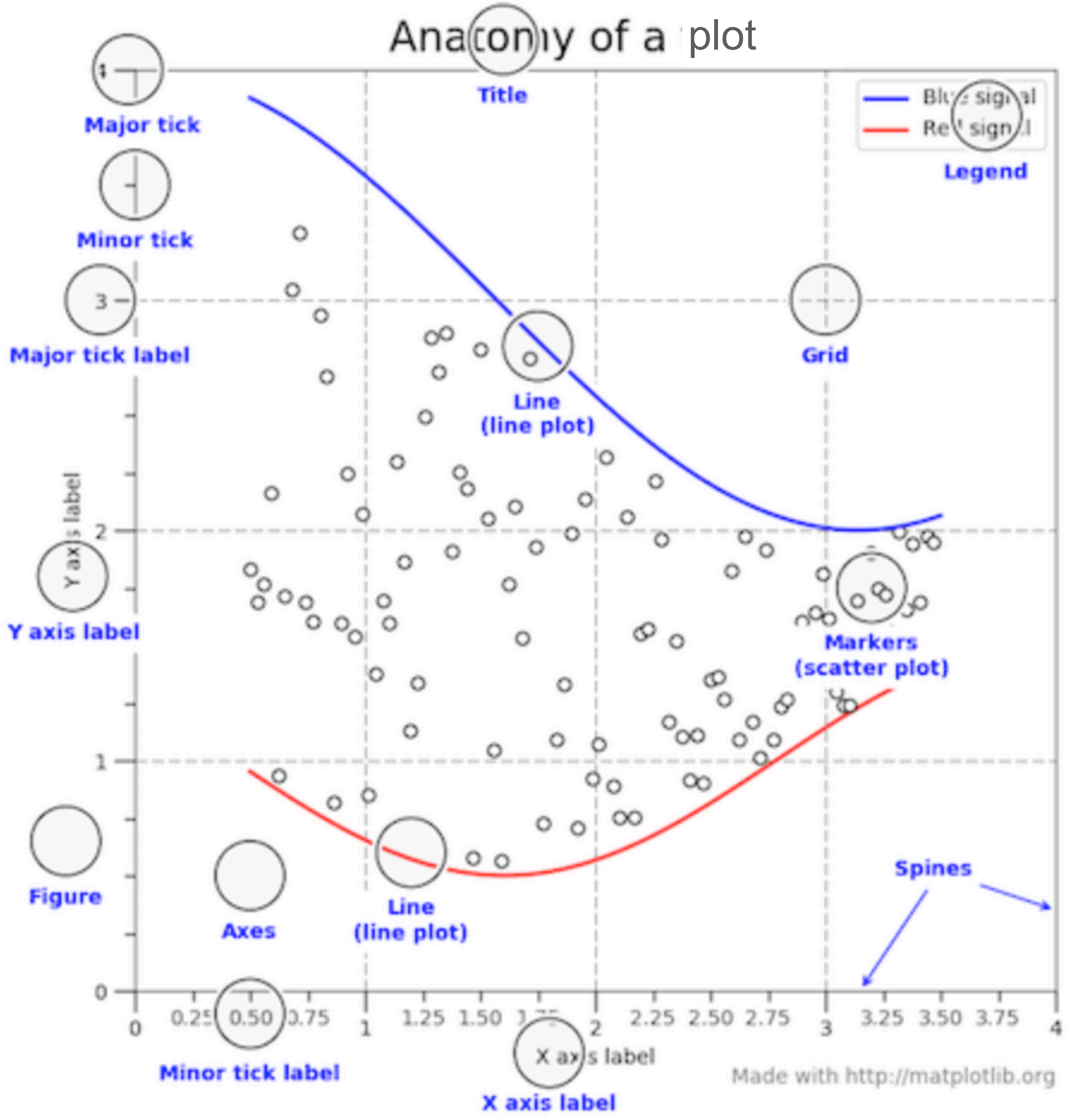
4) Appropriate tick resolution

5) Legend for different datasets, when applicable

6) Large enough fontsizes



Everything is customizable when plotting



https://realpython.com/python-matplotlib-guide/

You can change anything in a plot if you know how.

You can usually find how to do something by searching the documentation or searching the internet.

Official matplotlib documentation:

https://matplotlib.org/3.3.2/index.html

Resources

Loading data in Google Colab:

https://colab.research.google.com/notebooks/io.ipynb

Official numpy documentation:

https://numpy.org/doc/stable/reference/generated/numpy.genfromtxt.html

Official matplotlib documentation:

https://matplotlib.org/3.3.2/index.html

Tidal data:

https://tidesandcurrents.noaa.gov/noaatidepredictions.html?id=9447130&units=metric&bdate=20201001&edate=20201024&timezone=LST/LDT&clock=24hour&datum=MTL&interval=6&action=data