# Downloading and Exploring USGS River Data

*An Introduction to Geophysical Data Analysis in Python*

*Lessons Developed May 2023*

**Lesson Creator:** MacKenzie Jewell

**Total time:** 45 minutes

**Modality:** online

## Lesson Overview: Plot Taku River discharge in Python

This lesson serves as an introduction to geophysical data analysis in Python. In this lesson, learn how to access publicly available river gage data from the US Geological Survey and use Python to open, plot, and analyze the data. Perform simple timeseries and statistical analyses of river gage height from the Taku River near Juneau, Alaska.

Python is a programming language commonly used in data analysis in the geosciences. With Pythno, and other commonly used languages such as Matlab and R, you can write codes that allow you to read, modify, and analyze data using functions that can be executed programmatically in scripts. In this lesson, we will use an online browser-based Python program called JupyterLite. This program allows you to code in Python and view the results without installing any Python-supporting software on your computer.

## Learning Goals

1. Understand how to explore and download river data from the US Geologic Survey.
2. Understand how to open and plot text/CSV data in Python.

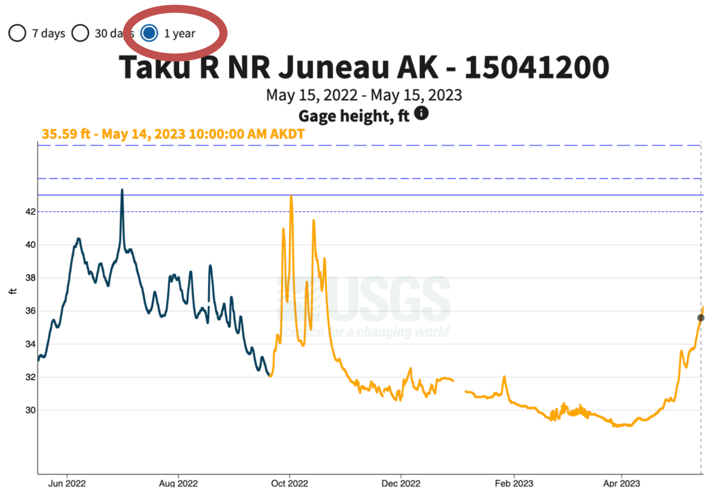
## Final Products

1. JupyterLite Python Notebook with code to load a CSV file, and plot(s) of data.

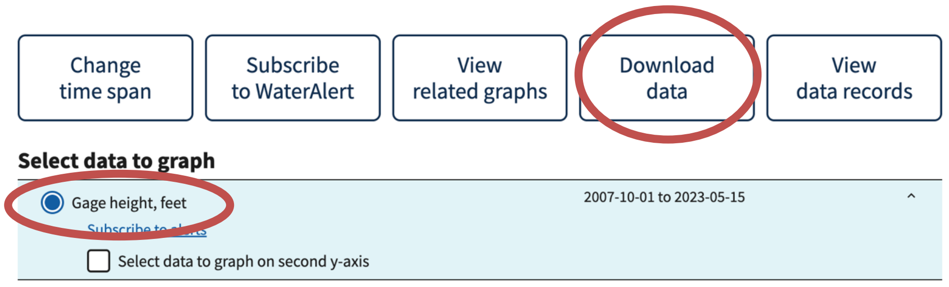
## Part 1: Download river data

1. Click the following link to navigate to the USGS a website where river data can be downloaded for the Taku River near Juneau, Alaska (site # 15041200) https://waterdata.usgs.gov/monitoring-location/15041200/#parameterCode=00065&period=P365D

2. At the top of the page, select **1 year** for the duration of the timeseries data.


7 days
30 days
[✓ 1 year]
Taku R NR Juneau AK - 15041200
May 15, 2022 - May 15, 2023
Gage height, ft
35.59 ft - May 14, 2023 10:00:00 AM AKDT
ft
Jun 2022
Aug 2022
Oct 2022
Dec 2022
Feb 2023
Apr 2023

3. Scroll down to the **Select data to graph** header. Select **Gage height, feet** if it is not already selected. Then, click the **Download data** button above.



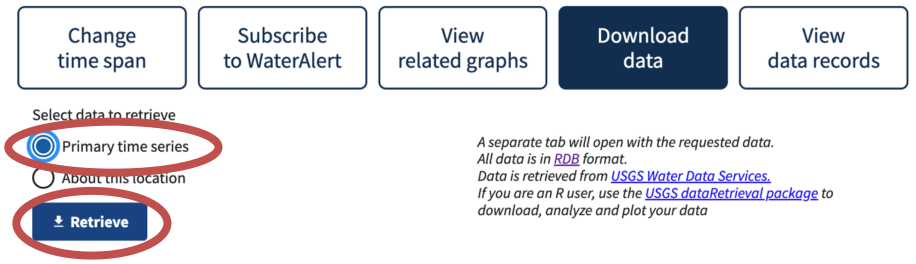
Select data to graph

[✓ Gage height, feet] 2007-10-01 to 2023-05-15

Subscribe to alerts

Select data to graph on second y-axis

4. Select **Primary time series** then click the **Retrieve** button.



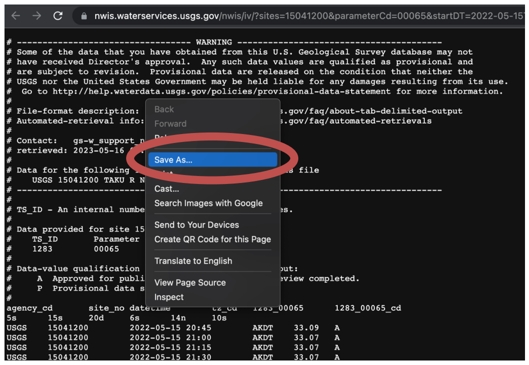
Select data to retrieve

[✓ Primary time series]

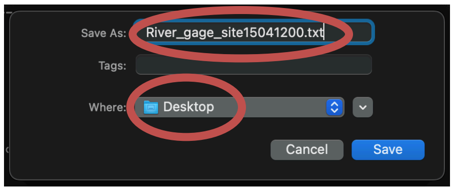
About this location

*A separate tab will open with the requested data.*
*All data is in RDB format.*
*Data is retrieved from USGS Water Data Services.*
*If you are an R user, use the USGS dataRetrieval package to download, analyze and plot your data*

5. You will be directed to a new webpage containing the data. **Right click** a blank space on the page, then select **Save As…** to save the data to your computer as a text file. Rename the text file using a clear and specific name. It's a good idea to avoid spaces, for example using a name like **River_gage_site15041200.txt**. Choose a download folder where you can easily locate the data again.



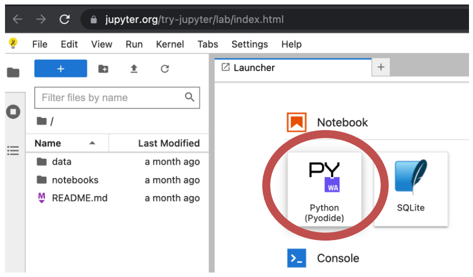
```
nwis.waterservices.usgs.gov/nwis/iv/?sites=15041200&parameterCd=00065&startDT=2022-05-15
# -------------------------------- WARNING ----------------------------------------
# Some of the data that you have obtained from this U.S. Geological Survey database may not
# have received Director's approval.  Any such data values are qualified as provisional and
# are subject to revision.  Provisional data are released on the condition that neither the
# USGS nor the United States Government may be held liable for any damages resulting from its use.
#  Go to http://help.waterdata.usgs.gov/policies/provisional-data-statement for more information.
#
# File-format description:                          s.gov/faq/about-tab-delimited-output
# Automated-retrieval info:                         s.gov/faq/automated-retrievals
#
# Contact:   gs-w_support_n
# retrieved: 2023-05-16
#
# Data for the following                             s file
#    USGS 15041200 TAKU R N
# -------------------------                          ----------------------------------
#
# TS_ID - An internal numbe                          es.
#
# Data provided for site 15
#    TS_ID       Parameter
#    1283        00065
#
# Data-value qualification                           put:
#     A  Approved for publi                          eview completed.
#     P  Provisional data s
#
agency_cd       site_no datetime        tz_cd   1283_00065      1283_00065_cd
5s      15s     20d     6s      14n     10s
USGS    15041200        2022-05-15 20:45        AKDT    33.09   A
USGS    15041200        2022-05-15 21:00        AKDT    33.07   A
USGS    15041200        2022-05-15 21:15        AKDT    33.07   A
USGS    15041200        2022-05-15 21:30        AKDT    33.07   A
```



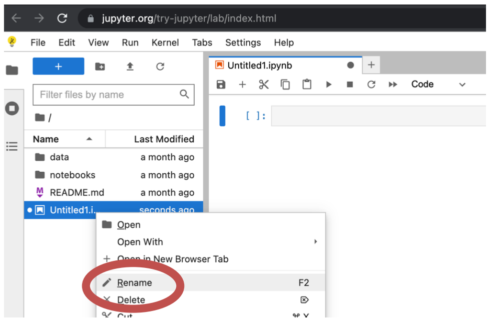
Save As: River_gage_site15041200.txt
Tags:
Where: Desktop

## Part 2: Plot river data in JupyterLite

6. Open a new JupyterLite tab: https://jupyter.org/try-jupyter/lab/index.html. You will arrive at a Launcher screen where you can open a new Jupyter Notebook, which is an interactive script for writing code in Python and plotting the results. Under Notebook at the top of the screen, double click **PY Python (Pyodide)** to launch a new notebook.



7. A notebook called **Untitled1.ipynb** should open. This is your Jupyter Notebook. The left side of the screen shows local folders and files on the JupyterLite webpage. You should see your notebook name there. **Right click** the file name to **rename** it. It's good to avoid spaces in the name, for example: **USGS_river_analysis.ipynb**.



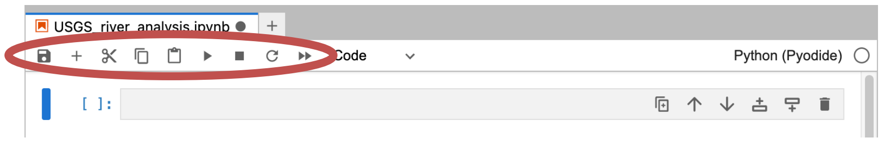
8. Your code is broken up into "Cells" (displayed as gray lines). You can run each cell individually. This allows you to work progressively and view results of each chunk of code. Click on a gray code cell to type in it. Hover over the symbols above the gray code lines to learn what they do.

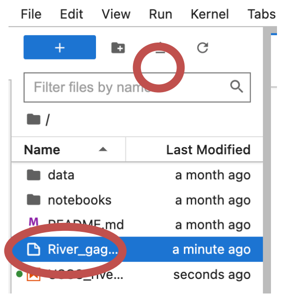
Which symbol will **insert a cell below**?

Which will **Run the selected cells and advance**?

Which will **Interrupt the kernel**? (stop executing the code while running if needed)

Which will **Restart the kernel**? (stop code execution, and refresh working memory)

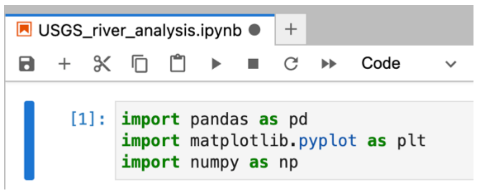




9. Now we'll upload our data to JupyterLite. **Drag and drop** your river data text file into the main folder containing your notebook (on the left side of the screen) or use the **Upload Files** icon (upward pointing arrow on the top left of the screen) to upload your river data text file. Check that you see the file name on the left side of the screen.

10. We'll start by importing some common python libraries into the notebook that we can use to open and plot data. Type the code lines below into the gray code cell. Execute the code by pressing **shift + enter** on your keyboard or pressing the **Run** button (single right arrow above the cell). An asterisk [*] will appear within the brackets left of the code while it is running. You will know the code cell ran successfully if a number [1] appears in the empty bracket left of the cell.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```



```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

11. Next, assign the name of your text file to the variable **filename** in a new code cell below the first. Run the cell (**shift + enter**), then check that it ran (a [number] appears left of the cell). Make sure to use quotation marks around the name as in the example below and match the name to your file.

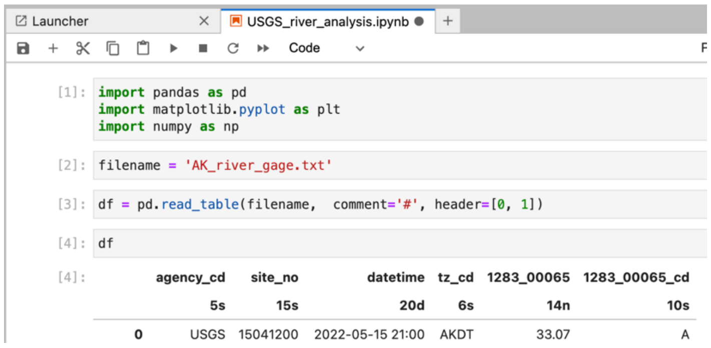
```
filename = 'River_gage_site15041200.txt'
```

**A tip to type this faster*: Once you've typed `filename = 'R` (or whatever is the first letter of your filename) try pressing **tab** on your keyboard. Python may autocomplete the name of the file for you! Just make sure to add the quotation mark at the end.

12. In a new cell, use the code below to read the data into a data frame called `df`. This uses the `read_table` function from *pandas* to read the data in a table format. The extra commands following `filename` tell *pandas* to ignore the comments at the top of the USGS text file (indicated by `#`) and assign the first two text rows as table headers. **Run** the cell.

```
df = pd.read_table(filename,  comment='#', header=[0, 1])
```

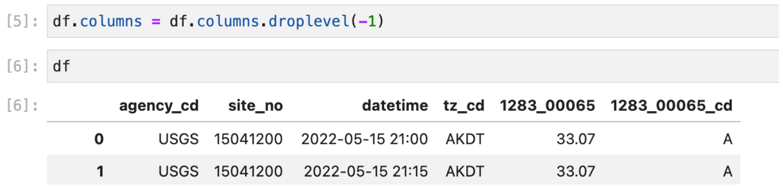
13. After executing the cell above, type `df` into a new cell below and **run** the cell to display the contents of `df`. Your code should look something like below. What are the columns and rows of `df`?



```
[1]: import pandas as pd
     import matplotlib.pyplot as plt
     import numpy as np

[2]: filename = 'AK_river_gage.txt'

[3]: df = pd.read_table(filename,  comment='#', header=[0, 1])

[4]: df

[4]:    agency_cd   site_no           datetime  tz_cd  1283_00065  1283_00065_cd
               5s       15s                20d     6s         14n            10s
     0       USGS  15041200   2022-05-15 21:00   AKDT       33.07              A
```

14. The bold headers at the top of `df` contain two lines, though we only need the information from the top one. **Run** the code below in a new cell to remove the second header. Then type `df` again into a cell below and **run** it to see whether it worked.

    `df.columns = df.columns.droplevel(-1)`

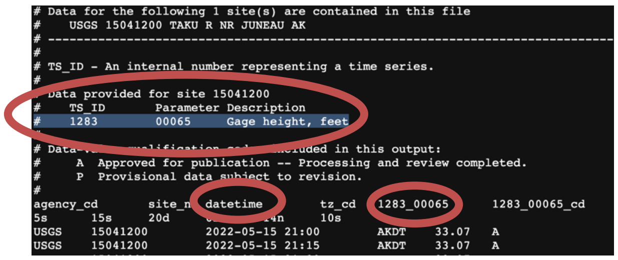


```
[5]: df.columns = df.columns.droplevel(-1)

[6]: df

[6]:    agency_cd   site_no           datetime  tz_cd  1283_00065  1283_00065_cd
     0       USGS  15041200   2022-05-15 21:00   AKDT       33.07              A
     1       USGS  15041200   2022-05-15 21:15   AKDT       33.07              A
```

15. One last data modification before we get to plotting the data! Run the code below in a new cell to put the date information (in the `'datetime'` column) in a nicer format for plotting.

    `df['datetime'] = pd.to_datetime(df['datetime'])`

16. Locate the names of headers that contain the **dates** and **gage height**. Refer to the contents of the text file you're analyzing if needed.

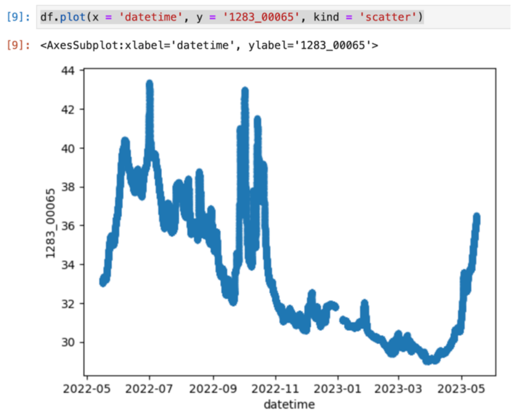


```
# Data for the following 1 site(s) are contained in this file
#    USGS 15041200 TAKU R NR JUNEAU AK
# -----------------------------------------------------------------------------------
#
# TS_ID - An internal number representing a time series.
#
# Data provided for site 15041200
#    TS_ID       Parameter Description
#    1283        00065     Gage height, feet
#
# Data-value qualification codes included in this output:
#     A  Approved for publication -- Processing and review completed.
#     P  Provisional data subject to revision.
#
agency_cd       site_n  datetime        tz_cd   1283_00065      1283_00065_cd
5s      15s     20d     14n     10s
USGS    15041200        2022-05-15 21:00        AKDT    33.07   A
USGS    15041200        2022-05-15 21:15        AKDT    33.07   A
```

*Here is a link to the text file for site 15041200 used in this example:*

https://nwis.waterservices.usgs.gov/nwis/iv/?sites=15041200&parameterCd=00065&startDT=2022-05-15T20:45:43.881-08:00&endDT=2023-05-15T20:45:43.881-08:00&siteStatus=all&format=rdb

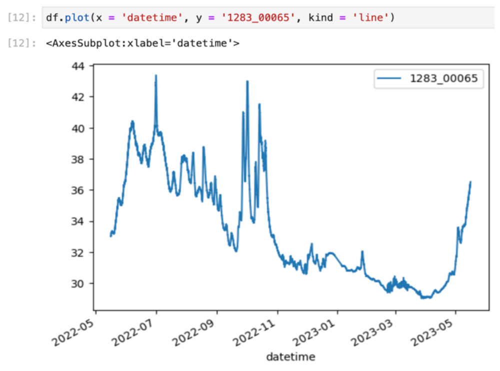
17. In a new code cell, make a timeseries of the gage height data using the following code. This code makes a scatter plot of the data in `df`, using the `'datetime'` column (date information) as the x-axis and the `'1283_00065'` column (gage height information) as the y-axis. `kind='scatter'` makes a scatter plot, which plots individual data points.



```
[9]: df.plot(x = 'datetime', y = '1283_00065', kind = 'scatter')
[9]: <AxesSubplot:xlabel='datetime', ylabel='1283_00065'>
```

```
df.plot(x = 'datetime', y = '1283_00065', kind='scatter')
```

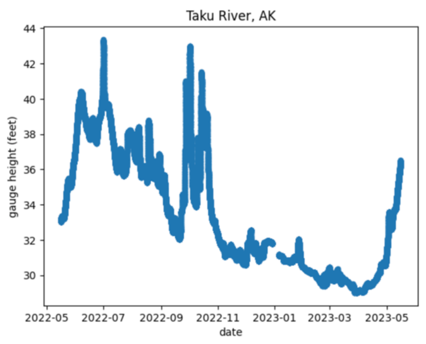
18. Copy and paste the code to a new code cell, but this time replace `kind='scatter'` with `kind='line'`, which plots a continuous line instead of individual data points. Compare the two plots. What is the difference between the two plots in January 2023? What are the strengths and limitations of these two different types of plots in displaying your data?



```
[12]: df.plot(x = 'datetime', y = '1283_00065', kind = 'line')
[12]: <AxesSubplot:xlabel='datetime'>
```

When you are analyzing data, it is important to look at the data in multiple different ways. Some methods of displaying data appear to provide information that is actually erroneous.

19. Let's save this timeseries plot to our computer. But first, let's add better labels so we know what we are looking at later. Go back to the code cell containing your scatter plot. Add the 3 new lines below to the existing code line to add a title and labels to the x and y axes. Be sure to enter all the lines in the same cell so they are added to the same plot. **Run** the cell again.



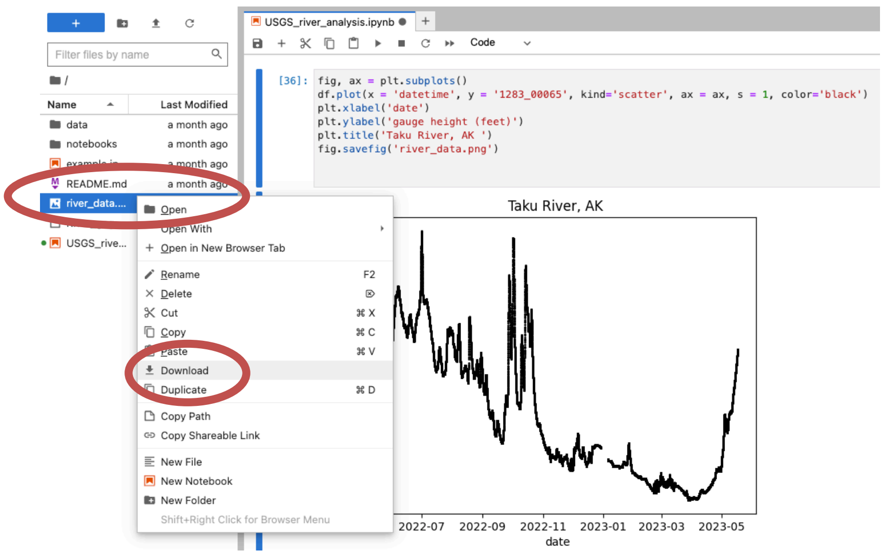
```
df.plot(x = 'datetime', y = '1283_00065', kind='scatter')
plt.xlabel('date')
plt.ylabel('gauge height (feet)')
plt.title('Taku River, AK')
```

20. Now that our plot is clearer, we can add to the current code in the cell to save the plot as a **png** file.

```
fig, ax = plt.subplots()
df.plot(x = 'datetime', y = '1283_00065', kind='scatter', ax = ax)
plt.xlabel('date')
plt.ylabel('gauge height (feet)')
plt.title('Taku River, AK ')
fig.savefig('river_data.png')
```

- Add the top line above the current code to create a figure which holds the plot information.
- Add `, ax = ax` at the end of the plot command (but before the last parenthesis) to add plot data to the figure axis.
- Add the bottom line below the code to save the figure to a file called `river_data.png`.

Once you've added these lines, **run** the code cell again. You should see the name of the image you created show up on the left side of the screen by your data and notebook names. Right click the file name to **Download** it to your computer locally.



```
[36]: fig, ax = plt.subplots()
      df.plot(x = 'datetime', y = '1283_00065', kind='scatter', ax = ax, s = 1, color='black')
      plt.xlabel('date')
      plt.ylabel('gauge height (feet)')
      plt.title('Taku River, AK ')
      fig.savefig('river_data.png')
```

**** to change the plot point color and size, add commands like `color='black'` and `s=1` as below:*

```
df.plot(x='datetime', y='1283_00065', kind='scatter', ax=ax, s=1, color='black')
```

21. Once you are finished, navigate to **File** in the top ribbon of the webpage. From there, **save** the notebook. Then, you can **Download** the notebook to save it to your computer.

## Part 3: Data Analysis Tasks

### Task 1

What are the maximum and minimum gage heights reached over the last year? And when did these occur? Do you have a hypothesis about what controls the timing of these events?

**Run** the following code in a new cell to find the index in the df dataframe corresponding to the maximum gage height value. What is the index?

```
index = df['1283_00065'].idxmax()
index
```

**Run** the following code in a new cell to print the data point associated with this index. What is the maximum gage height? And when did this occur?

```
df.loc[index]
```

**Run** the following code in a new cell to find and print the data point associated with the minimum river gage height.

```
index = df['1283_00065'].idxmin()
df.loc[index]
```

Navigate back to the USGS site where you downloaded the data:
https://waterdata.usgs.gov/monitoring-location/15041200/#parameterCode=00065&period=P365D

Scroll down to the bottom of the page to look at the Map of the Taku River and the **Monitoring camera**. What do you notice about the location of the river and how it appears over time? Can you use this information to infer why the timeseries data looks like it does? And why the minimum and maximum river gage heights occurred when they did?

### Task 2

Can you tell from your scatter plot what the most common range of river gage heights is over the past year? We can make a histogram of the gage height variable using the code below to answer this question more precisely. What is the most frequent river gage height?

```
df.plot(y = '1283_00065', kind = 'hist')
plt.xlabel('gauge height (feet)')
```

## Task 3

What is the average gage height over this year? How often is the gage height greater than average? How often is the gage height greater than 40 feet?

To answer these questions, assign the gage height data to a variable called `gage_height` by accessing the gage column (`'1283_00065'`) of `df`. **Run** the code below in a new cell.

```
gage_height = df['1283_00065']
gage_height
```

**Run** the code below in a new cell to calculate the mean gage height over the time series.

```
np.mean(gage_height)
```

**Run** the code below in a new cell to calculate the total length of the time series.

```
total_length = len(gage_height)
total_length
```

**Run** the code below in a new cell to calculate the amount of time in the series when the gage height is above average. In your code, replace `NUM` with the mean gage height you calculated previously.

```
length_above = np.sum(gage_height > NUM)
length_above
```

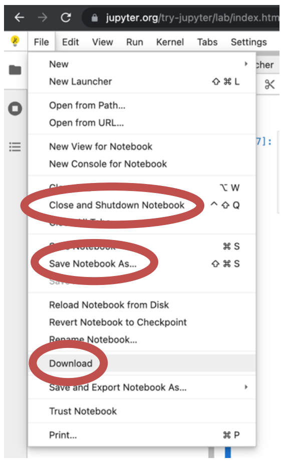
**Run** the code below in a new cell to find the fraction of time over the series when the gage height is above average.

```
length_above/total_length
```

Go back to the cell where you calculated `length_above`. Replace `NUM` and **rerun** the cell to find the length of time above 40 feet. **Rerun** the `length_above/total_length` cell to find the fraction of time the gage height is above 40 feet.

## Saving and Downloading the notebook

Once you are finished, navigate to **File** in the top ribbon of the webpage. From there, **save** the notebook. Then, you can **Download** the notebook to save it to your computer. Once you are all finished, select **Close and Shutdown Notebook** to terminate the code.



If you'd like to run the code again on JupyterLite at a later date, keep track of the notebook you created today and the link from where you downloaded the data. You can upload both to JupyterLite and run the code again, or modify it to explore new data.