

Astro-Imaging Colab Instructions

Summary

This document provides a walkthrough to set up Google Colaboratory and complete the assignment. You may already know some of the information in here; skip over what isn't useful to you. Use the table of contents (displayed on the left sidebar & the following page) to quickly jump to a specific section.

Setup

- Install Google Colaboratory to your Google Drive.
- Make a copy of the Colab file.
- Upload your data file to Session Storage.
- Run all of the code blocks by going to “Runtime” and selecting “Run All.”
- Double-click text and code blocks to open them and edit them when required.

Procedure

- **Part 1:**
 - Edit the filename of the datafile you wish to import.
 - View the data table and make a prediction.
- **Part 2:**
 - view the first two plots to visually assess your data.
 - Comment on what the polar plot represents and add a figure caption.
 - Apply a linear fit to the position plots and discuss how well it fits.
 - Observe and describe the delta position plots.
- **Part 3:**
 - Read the anatomy of a sine curve section, and use this information to input guess fits for the RA and Dec of moons 1 and 2.

- Once you input parameter guesses, click “submit” (do NOT rerun this code block, doing so will erase your parameter inputs).
 - Repeat the process of guessing and plotting until you have good fits.
 - Run the next code block to refine the guesses with a fitting package.
 - Use code to calculate the mass of Uranus, comment on process and result.
- **Part 4:**
- Observe and comment on the Monte Carlo distance simulation.
 - Observe the mass simulation and the heatmap, test out different errors
 - Comment on what you see.

Table of Contents

[Summary](#)

[Table of Contents](#)

[Colab Basics:](#)

[What is Colab?](#)

[How do I set it up?](#)

[How do I upload a file?](#)

[How do I run the Code?](#)

[How do I edit code and text?](#)

[Assignment Instructions](#)

[Part 1: Setup](#)

[Part 2: Descriptive Plotting](#)

[Part 3: Data Analysis](#)

[Part 4: Estimating the Mass from Multiple Distances](#)

Colab Basics

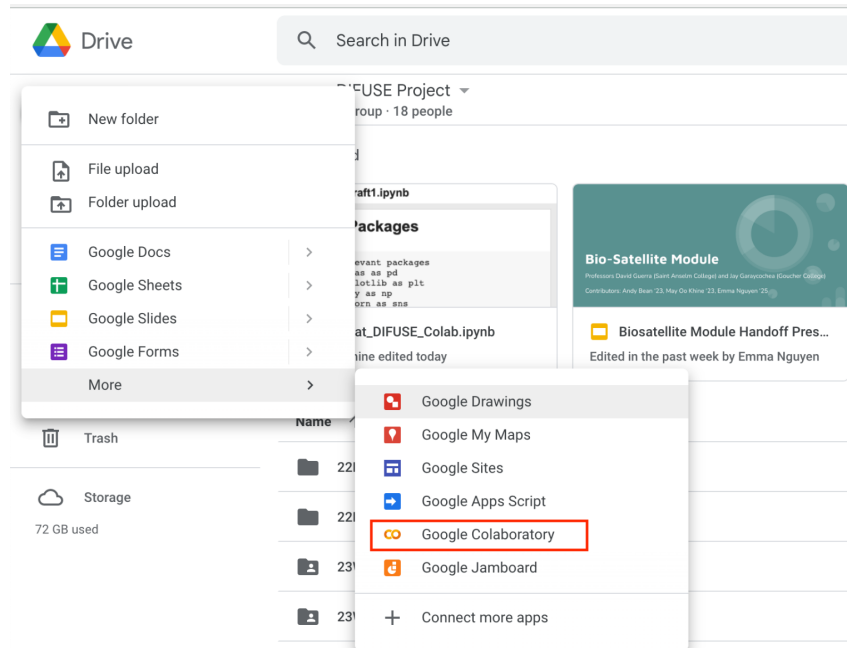
What is Colab?

Colab, or Google Colaboratory, is the “coding version” of Google Docs. Colaboratory files are a notebook type environment (meaning they contain both code blocks and text) that can be accessed by anyone with a google account. Colab can run code on a “hosted” runtime – this means you don’t need to download anything to run code! Google downloads everything that is needed to a space in the cloud.

Colab files run on the *Python* coding language (ver 3.9), and contain a number of additional packages for plotting, statistics, and many other useful tasks. More importantly, they allow us to create “interactive” plots and displays that don’t require the user to know how to code. As part of this lab, you’ll not only learn how to use a Colab file, but also how to write a little bit of Python code (if you don’t know already)!

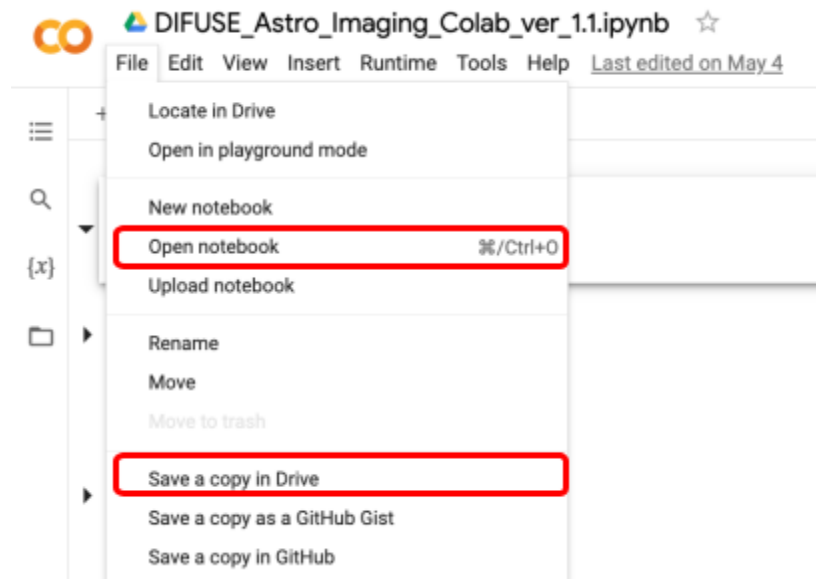
How do I set it up?

- 1) First, you will need to enable Colab on your google account. Go to Drive and click on “New” as if you were going to open a new document. Then go to “More” and finally “Connect More Apps.” Search for “Colaboratory” in the menu that appears and connect it to your google drive. Now, when you go to “New” and “More,” you should see an option for Google Colaboratory like in the image below:



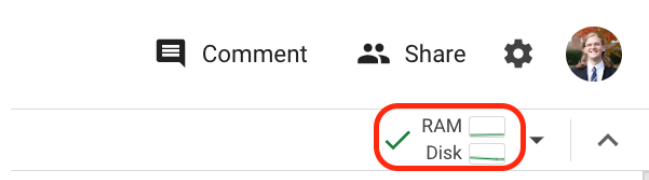
- 2) Open the link to the Colab file you were sent. Navigate to the “*File*” menu and select “*Save a Copy in Drive*” to create your own version for editing. You can also open up a blank file and go to “*File*” → “*Open Notebook*” and select the file you were sent, this will also open the Colab file you were sent into a new file.

Now, you’re ready to start Colabbing!

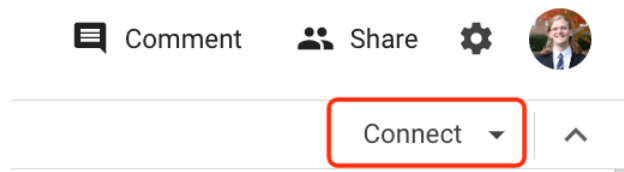


How do I upload a file?

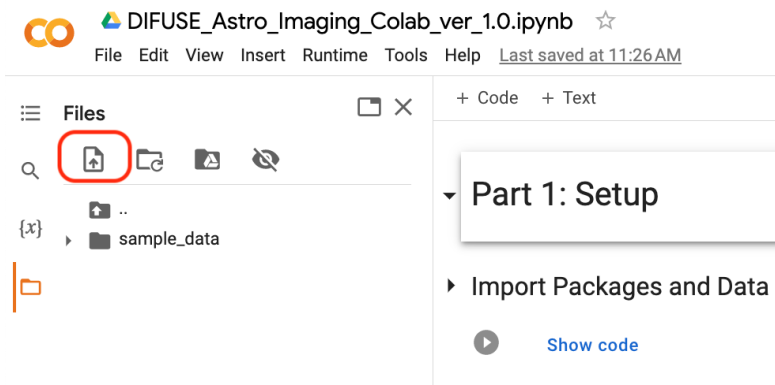
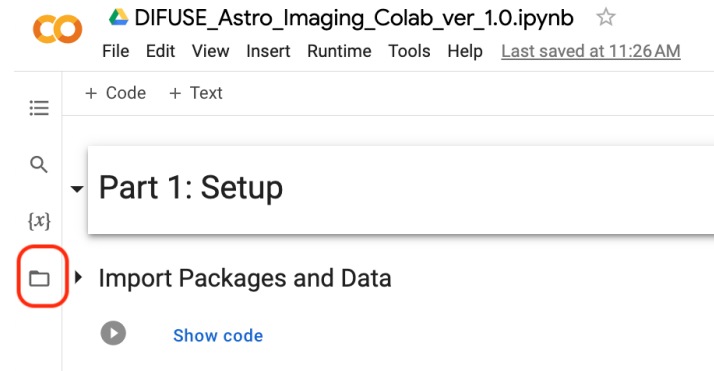
- 1) Many code files require a data file to pull data from. In some cases, the code is written so that it obtains the data itself through the internet. In this case, you don't need to do anything!
- 2) If you do need to upload a file yourself, you will upload a file to the *session storage*. This storage is temporary, so anytime you close the Colab or the runtime is disconnected, you need to upload the file again.
 - a) First, check that you are connected to a runtime. If you see a green checkmark in the top right corner of the Colab file with the words RAM and Disk, then you are connected and you can proceed to uploading.



- b) If you do not see this but instead see the word “*Connect*,” you are not yet connected to a runtime. Simply click the word “*Connect*” and wait a moment as Google creates a connection for you.



- d) Now you are ready to upload a file! Navigate to the left side of the page and click on the “folder” icon to open the file directory. Then, click the button to “upload to session storage” and select the file to upload from your computer. **Make sure that your file is the right type and has the correct formatting (column names and what's inside them)!**



- e) You may need to change the filename in the code so that the code knows what file to read in. See “how do I edit code and text” below for more instructions. You should open the first code block and change the file name to match yours. A long line of “+++”s will indicate where to edit. Simply change the filename inside the quotes to match yours:

```
# ++++++
# EDIT FILENAME

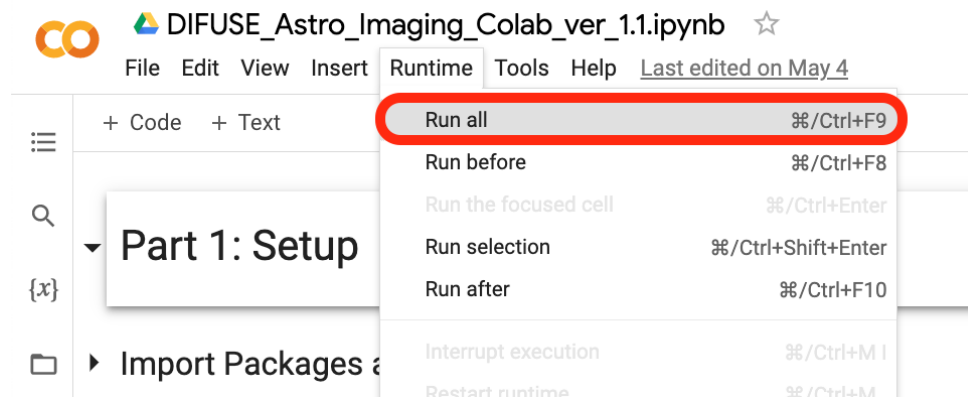
# change the filename inside the quotes to match the name of the file you uploaded
# you don't need to change anything else, and make sure to leave the quotes there!

df = pd.read_csv('Test_VG_Uranus_Data.csv')

# ++++++
```

How do I run the Code?

- 1) Colab is a notebook-type environment that has both code blocks and text blocks (also known as markdown blocks). You need to “run” the code blocks to view their output. The easiest way to get everything working is to go to the “Runtime” menu and then select “Run all.” This will run all of the code blocks in order. This may take up to a minute depending on the complexity of the code.



- 2) In most cases, you won't need to run the code blocks again, but if you edit any of the code (see “how do I edit code” below) you will need to run them again. Rather than running all of the code blocks in the file again, you can run a block individually by pressing the “play” button on the top left of the block.

► View Data Table



[Show code](#)

index	N	Decimal Day (UT)	Moon1 (RA)	Moon1 (Dec)	Moon2 (RA)
0	1	14.903	34.258	13.232	34.25
1	2	14.944	34.258	13.232	34.249
2	3	14.985	34.258	13.232	34.25

- 3) Note: If you close the Colab file, or leave it idle for a long time, Google will disconnect the runtime. This deletes the code's memory, but won't erase any changes you made. When you are ready to use the file again, begin by uploading your data and selecting “Run All” again.
- 4) Errors may occur when running your code. You will be alerted by a red exclamation point at the bottom of the screen, and you might see some output like the image below. The error is most likely in your data table or file name, so go to the Part 1 instructions for troubleshooting help on this.

View Data Table



Show code

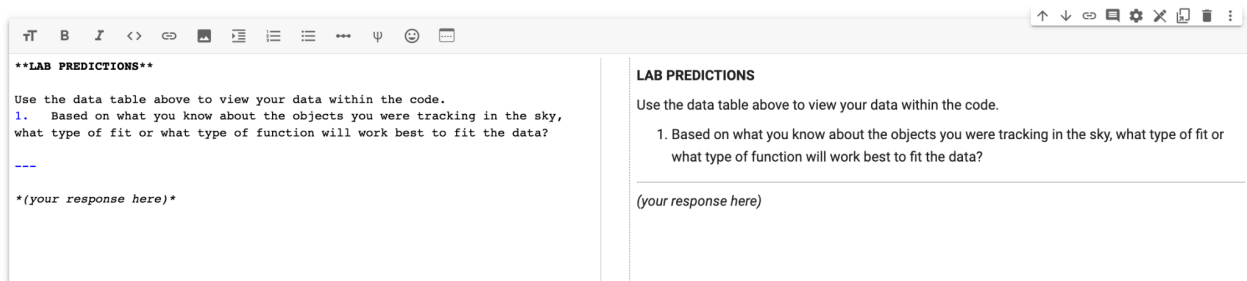


```
-----  
TypeError                                Traceback (most recent call last)  
  /usr/local/lib/python3.10/dist-packages/pandas/core/ops/array_ops.py in _na_arithmetic_op(left, right, op, is_cmp)  
    164     try:  
--> 165         result = func(left, right)  
    166     except TypeError:
```

How do I edit code and text?

Text

- 1) To edit text, double click on a text block to open it in an editor view. This view displays the [editable text on the left](#), and displays its output on the right. Type normally into the left hand side. You can use the formatting options to add bold, italics, etc.



- 2) When you are finished, click outside the text box or press the escape key. The editor view will disappear and only the output will be displayed.

Code

- 1) To edit code, you first need to view it. Most code blocks have their code “hidden” to avoid clutter. Click on “show code” to reveal the hidden code. To hide the code again, click on the triangle to the left of the code block header (this also collapses the bigger section headers if you really want to reduce clutter!)

▸ View Data Table



Show code



index	N	Decimal Day (UT)	Moon1 (RA)	Moon1 (Dec)	Moon2 (RA)
0	1	14.903	34.258	13.232	34.25
1	2	14.944	34.258	13.232	34.249
2	3	14.985	34.258	13.232	34.25
3	4	15.046	34.258	13.232	34.25

▾ View Data Table



```
##title View Data Table

# =====
# create a table view as well
# =====

dfview = df[['N', 'Decimal Day (UT)', 'Moon1 (RA)', 'Moon1 (Dec)',
             'Moon2 (RA)', 'Moon2 (Dec)', 'Moon1 (dRA)', 'Moon1 (dDec)',
             'Moon2 (dRA)', 'Moon2 (dDec)', 'Moon1 (r)', 'Moon1 (theta)',
             'Moon2 (r)', 'Moon2 (theta)']].round(3)
```

- 2) Now you will view the underlying code with the output displayed at the end of the code block. You can type into this space and then run the individual block by selecting the “play button” at the top of the code block. Places in the code you will need to edit are denoted by a long line of “+++”s. You don’t need to edit any code outside of the +++’s, but you can read the rest if you’re curious!

```
# ++++++
# EDIT FILENAME

# change the filename inside the quotes to match the name of the file you uploaded
# you don't need to change anything else, and make sure to leave the quotes there!

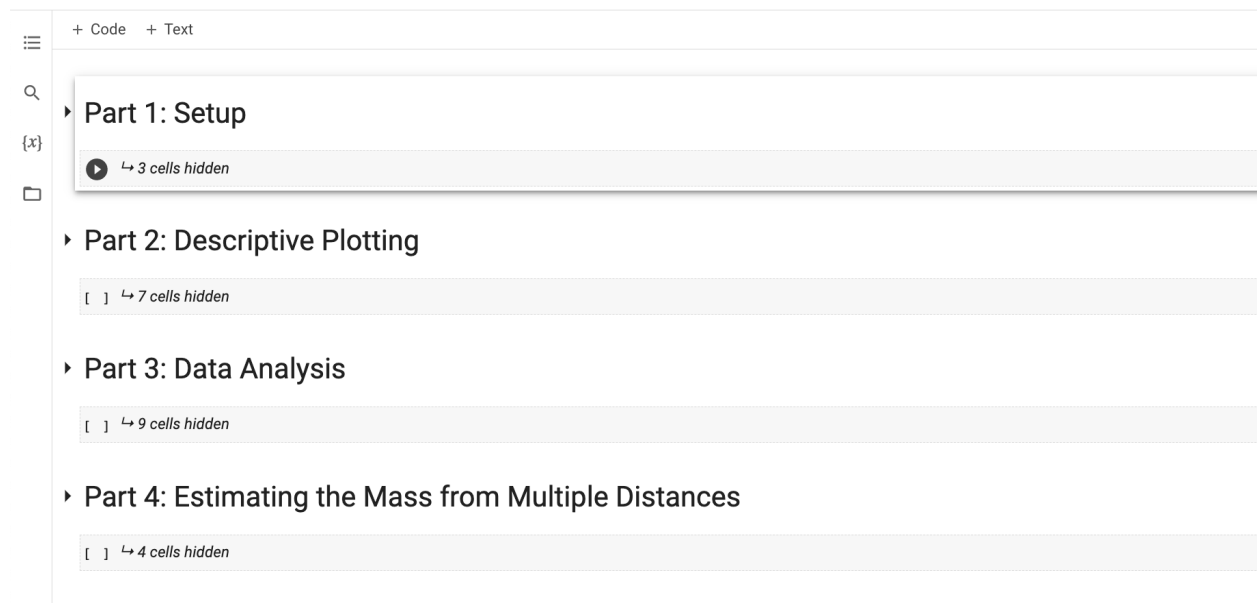
df = pd.read_csv('Test_VG_Uranus_Data.csv')

# ++++++
```

Assignment Instructions

The assignment colab outline is shown below.

You can collapse the code down to look like this, and then open just one section of the code at a time



Part 1: Setup

After you upload your data and “run all”, you should be looking at a Colab file with all the code recently run and ready for you to play with.

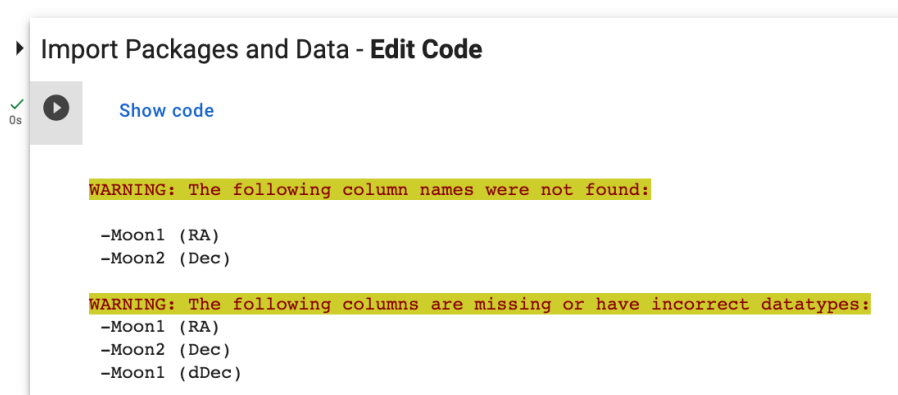
In the colab, open “Part 1: Setup”, and view the 3 blocks within it:

- Import Packages and Data - **Edit Code** (*Code Block*)
- View Data Table (*Code Block*)
- Lab Predictions (*Text Block*)

Import Packages and Data

At this point, you should have already edited the code to reflect the file name of your datafile. If you haven't done that, go back to [How do I upload a file?](#) and do that step now. If you get an error with a red exclamation mark in this section, check to make sure the file name in the code matches the file you want to upload.

The code also has some more helpful error messages manually coded in. If this block of the Colab runs, but you see output like the image below, you need to check that your data file has the right column names and doesn't have any stray characters (letters or symbols) in the data, just numbers:



```
Import Packages and Data - Edit Code

0s Show code

WARNING: The following column names were not found:

-Moon1 (RA)
-Moon2 (Dec)

WARNING: The following columns are missing or have incorrect datatypes:

-Moon1 (RA)
-Moon2 (Dec)
-Moon1 (dDec)
```

View Data Table

If you don't have any of the above error messages, you should be able to view a data table of the file you just uploaded! Check that everything looks as expected. You can use the "Filter" button in the top right corner to filter down the data, the page buttons in the bottom right corner to change pages, and the dropdown in the bottom left corner to choose how many entries to display per page. Note that the columns with (r) and (theta) were created by the code from the RA and dec observations, so you don't need to have them in your data file.

View Data Table

Show code

1 to 25 of 65 entries **Filter**

index	N	Decimal Day (UT)	Moon1 (RA)	Moon1 (dRA)	Moon1 (r)	Moon1 (Dec)	Moon1 (dDec)	Moon1 (theta)	Moon2 (RA)	Moon2 (dRA)	Moon2 (r)	Moon2 (Dec)	Moon2 (dDec)	Moon2 (theta)
0	1	14.903	34.258	0.002	0.012	13.232	-0.012	-1.441	34.25	-0.007	0.007	13.245	0.002	2.847
1	2	14.944	34.258	0.002	0.011	13.232	-0.011	-1.404	34.249	-0.007	0.007	13.245	0.002	2.889
2	3	14.985	34.258	0.002	0.011	13.232	-0.011	-1.409	34.25	-0.007	0.007	13.245	0.002	2.918
3	4	15.046	34.258	0.002	0.012	13.232	-0.012	-1.389	34.25	-0.007	0.007	13.244	0.001	3.006
4	5	15.854	34.262	0.005	0.011	13.234	-0.009	-1.082	34.251	-0.006	0.007	13.24	-0.004	-2.554
5	6	15.916	34.262	0.005	0.011	13.234	-0.009	-1.057	34.251	-0.006	0.007	13.24	-0.004	-2.521
6	7	15.979	34.262	0.005	0.011	13.234	-0.01	-1.062	34.252	-0.006	0.007	13.239	-0.004	-2.465
7	8	16.042	34.262	0.005	0.011	13.235	-0.009	-1.045	34.252	-0.005	0.007	13.239	-0.005	-2.386
8	9	16.125	34.263	0.006	0.011	13.235	-0.009	-0.98	34.252	-0.005	0.007	13.239	-0.005	-2.346
9	10	16.868	34.266	0.008	0.01	13.239	-0.006	-0.662	34.256	-0.002	0.008	13.237	-0.008	-1.822
10	11	16.912	34.266	0.008	0.01	13.239	-0.006	-0.637	34.257	-0.002	0.008	13.237	-0.008	-1.783
11	12	16.958	34.266	0.008	0.01	13.239	-0.006	-0.617	34.257	-0.002	0.008	13.237	-0.008	-1.775
12	13	17.042	34.267	0.008	0.01	13.24	-0.005	-0.585	34.258	-0.001	0.008	13.237	-0.008	-1.68
13	14	17.118	34.267	0.008	0.009	13.24	-0.005	-0.506	34.258	-0.001	0.008	13.237	-0.008	-1.657
14	15	17.853	34.27	0.009	0.009	13.245	-0.001	-0.148	34.264	0.003	0.008	13.238	-0.008	-1.245
15	16	17.916	34.27	0.009	0.009	13.246	-0.001	-0.079	34.264	0.003	0.008	13.238	-0.008	-1.226
16	17	17.979	34.27	0.009	0.009	13.246	-0.0	-0.046	34.264	0.003	0.008	13.239	-0.008	-1.193
17	18	18.083	34.27	0.009	0.009	13.247	0.0	0.011	34.265	0.004	0.008	13.239	-0.008	-1.123
18	19	18.121	34.27	0.009	0.009	13.247	0.0	0.034	34.265	0.004	0.008	13.239	-0.007	-1.113
19	20	18.854	34.272	0.008	0.009	13.252	0.004	0.489	34.27	0.006	0.007	13.244	-0.004	-0.588
20	21	18.916	34.272	0.008	0.009	13.252	0.004	0.519	34.27	0.006	0.007	13.244	-0.004	-0.565
21	22	18.978	34.272	0.007	0.009	13.253	0.005	0.556	34.271	0.006	0.007	13.245	-0.004	-0.548
22	23	19.042	34.272	0.007	0.009	13.253	0.005	0.585	34.271	0.006	0.007	13.245	-0.003	-0.457
23	24	19.833	34.274	0.005	0.01	13.258	0.008	1.005	34.274	0.006	0.007	13.252	0.002	0.264
24	25	19.906	34.274	0.005	0.01	13.258	0.008	1.025	34.275	0.006	0.007	13.252	0.002	0.312

Show 25 per page

1 2 3

Lab predictions:

Read this text block and answer the numbered questions by editing the text block to write your answer below the horizontal line.

Part 2: Descriptive Plotting

Now we're ready to start plotting your observations! In the descriptive plotting section, we will examine the moon position observations in several ways. This section is valuable to understand the moon movement and what each variable represents.

Open up "Part 2: Descriptive Plotting" and look at the 7 blocks within it:

- Plot moon observations in sky coordinates (*Code Block*)
- Plot moon observations in a polar projection - **Edit Code** (*Code Block*)
- Response 1 (*Text Block*)
- Examine Moon position over time (*Code Block*)

- Response 2 (*Text Block*)
- Examine Relative (delta) moon positions over time (*Code Block*)
- Response 3 (*Text Block*)

Plot moon observations in sky coordinates and polar projection

View both of these plots that the code produced for you. Use the questions in response block 1 to help you understand the second plot. Once you do, [open up the code and type an informative figure label](#).

```
# ++++++
# YOUR CODE HERE:
# Write an informative figure caption that will display beneath the plot
# Add "\n" to start a new line of text
plt.text(0.00, -0.019,
         "Figure 1. [Type a figure caption here]",
         transform=plt.gcf().transFigure)

# ++++++
```

Response 1

Read this text block and answer the numbered questions by editing the text block to write your answer below the horizontal line.

Examine Moon Position Over Time

View the plot and use the dropdown to apply a fit to the data. An example dropdown is highlighted in red in the image below.

► Examine moon position over time

✓ [5] [Show code](#)

Model: No Model

Uranus Moons



Response 2

Read this text block and answer the numbered questions by editing the text block to write your answer below the horizontal line.

Examine Relative Moon Position Over Time

View the plot and use the dropdown to change between viewing RA or Dec for each moon.

Response 3

Read this text block and answer the numbered questions by editing the text block to write your answer below the horizontal line.

Part 3: Data Analysis

Now that we understand what's going on with the data, we want to analyze it to fit a sine wave to the motion of the moons. From this fit, we can extract an amplitude and frequency, and use that to calculate the mass of Uranus! Open up “Part 3: Data Analysis” if you haven't done so already and look at the 9 blocks within it:

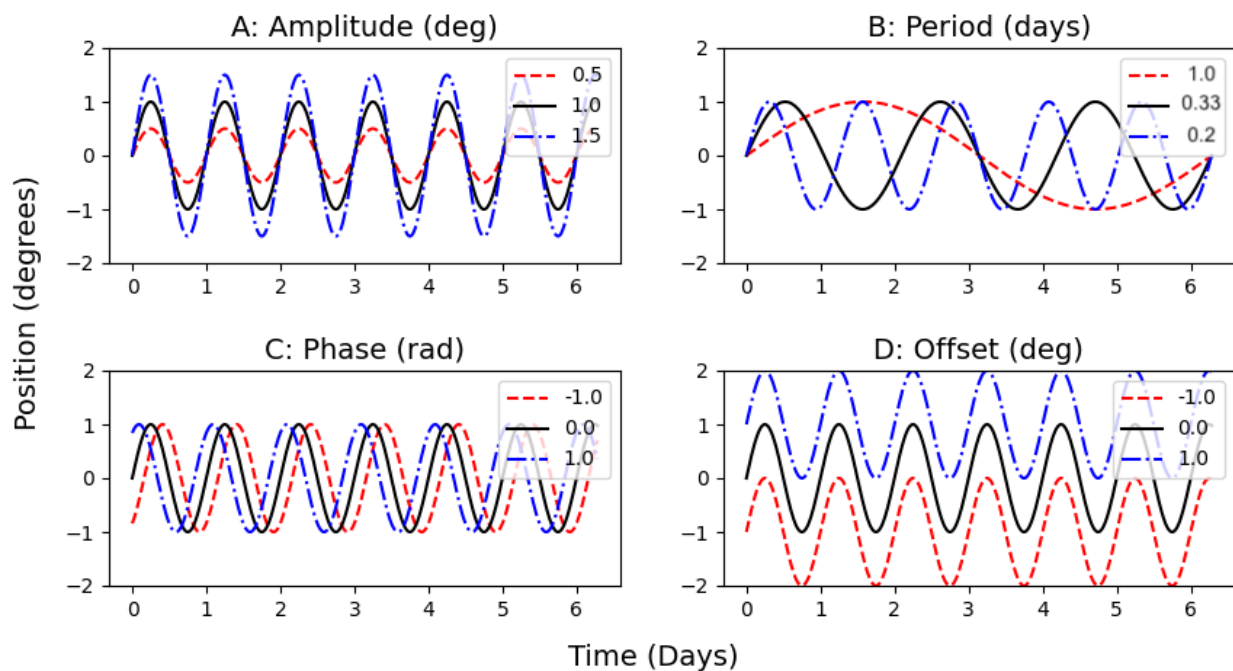
- Anatomy of a Sine Curve (*Text Block*)
- “Fit a Sine Curve...” (*Text Block*)
- Input Initial Parameter Values Here (*Code Block*)

- Store your Parameters Here (*Text Block*)
- “When you are satisfied...” (*Text Block*)
- Fit a Sine curve using the Scipy package (*Code Block*)
- Response 4 (*Text Block*)
- Calculate the Mass of Uranus - **Edit Code** (*Code Block*)
- Response 5 (*Text Block*)

Anatomy of a Sine Curve and “Fit a Sine Curve...”

Read this text block to learn more about the parameters of sine curves, in a moment, we’ll apply this information to make our own sine curves.

Anatomy of a sine curve



Input Initial Parameters and Plot sine functions

Use the text boxes in the input initial parameters section to make a guess at a sine curve that will fit the data, then [press submit to update the values within the code](#). This will plot your guess on top of the data and show you the residuals of your fit. The first

submission will make plots appear, following submissions will replace the previous plot with an updated one. The goal is to minimize the residuals!

Input initial parameter values here

[7] [Show code](#)

Input initial parameter values here

Moon1_dRA

A (°)

B (days)

C (rad)

D (°)

Submit/ Resubmit

1. Enter guesses in the boxes

Moon1_dDec

A (°)

B (days)

C (rad)

D (°)

Moon2_dRA

A (°)

B (days)

C (rad)

D (°)

Moon2_dDec

A (°)

B (days)

C (rad)

D (°)

2. press submit to make the plots appear the first time.
After that, resubmitting will replace the old plots with new ones

Store your parameters here

Type your parameter estimates here when you are satisfied with them. This way they won't be deleted if you come back to the file and press "Run All".

double click me to edit this text and store your numbers!

Moon1_ΔRA | A = __, B = __, C = __, D = __

Moon1_ΔDec | A = __, B = __, C = __, D = __

Moon2_ΔRA | A = __, B = __, C = __, D = __

Moon2_ΔDec | A = __, B = __, C = __, D = __

3. Repeat 1 and 2 until you think the fit is close, then store your values here

Repeat this process of submitting values and plotting the curve until you are satisfied with your fits. Then try running the next code block to optimize your fit (see below). You can try to fit all at once, or go one at a time and leave the others alone

Store your Parameters and “When you are satisfied...” and Fit a sine curve using the Scipy package

The text block beginning with “Store your parameters” can be used to store your A, B, C, and D values for each curve. If you “Run all” again, your entered values will be deleted, so typing them in a markdown block is a good way to save them.

The text block beginning with “when you are satisfied...” has instructions for this next step: Once you have decent guesses from the previous code block, re-run the “Scipy” code block to optimize your fit. If one of your guesses did not converge to the data, you will get an error message like the ones shown below:

When you are satisfied with how well your sine curves match the data, run the next code block to use the scipy package to fit a sine curve to each dataset. The function to generate the best-fit parameters will use your initial parameters as a starting point. If one of your initial models is not a good fit, the function might not converge parameter solutions and you will see an error message.

► Fit a sin curve using the scipy package



Re-run this block when you have made good manual fits

[Show code](#)

```

/usr/local/lib/python3.10/dist-packages/scipy/optimize/_minpack_py.py:906: OptimizeWarning: Covariance of the parameters could not be estimated
warnings.warn('Covariance of the parameters could not be estimated',
               (these are error messages from failed fits)
/usr/local/lib/python3.10/dist-packages/scipy/optimize/_minpack_py.py:906: OptimizeWarning: Covariance of the parameters could not be estimated
warnings.warn('Covariance of the parameters could not be estimated',
/usr/local/lib/python3.10/dist-packages/scipy/optimize/_minpack_py.py:906: OptimizeWarning: Covariance of the parameters could not be estimated
warnings.warn('Covariance of the parameters could not be estimated',
/usr/local/lib/python3.10/dist-packages/scipy/optimize/_minpack_py.py:906: OptimizeWarning: Covariance of the parameters could not be estimated
warnings.warn('Covariance of the parameters could not be estimated',

```

Response 4

Read this text block and answer the numbered questions by editing the text block to write your answer below the horizontal line.

Calculate the Mass of Uranus

This section might look a little strange at first, you'll want to click "Show code" to open it up. [Read through the "README" section](#) that explains how to use variables, functions, and comments in Python. The text you see with "myvariable" is from these examples. Then, [use python like a calculator to estimate the mass of Uranus from your fit parameters](#) given that the average distance between the Earth and Uranus is 19.2 AU.

► Calculate the mass of Uranus - **Edit Code**



[Show code](#)

```

myvariable = 2.0
myvariable = 34.0
myvariable*2 = 68.0
myvariable_2/myvariable = 2.0
The square root of myvariable + 2 is 6.0

The mass of Uranus is estimated to be 0.00e+00 kg

```

```

# Now it's your turn!

# =====
# USING YOUR PARAMETERS FROM THE FITS ABOVE,
# FIND THE MASS OF URANUS
# =====

# other helpful functions or commands:
# - np.pi is the value of pi, this works just like a number (e.g. np.pi*2 = 6.28)
# - np.cos(x) takes the cos of x (where x is in radians)
# - np.deg2rad(x) converts x from degrees to radians
# - x**2 takes the square of x
# - x**3 takes the cube of x
# - x**n takes x to the nth power (replace n with a number or a variable)
# - np.sqrt(x) takes the square root of x

# you may also want to take the mean of two estimates, one from moon 1 and one from moon 2
# - np.mean([x1,x2]) takes the mean of all values in the square brackets, separated by a comma

# ++++++
# YOUR CODE HERE

m_uranus = 0 #replace this with your final answer to print it out!

# ++++++

print("\nThe mass of Uranus is estimated to be %.2e kg" % m_uranus)

```

Response 5

Read this text block and answer the numbered questions by editing the text block to write your answer below the horizontal line.

Part 4: Estimating the Mass from Multiple Distances

In part 3, we calculated the mass from our fit parameters and just one distance. However, computers allow us to make many calculations at once, so what if we put in a whole range of distances? The code will walk you through this process in this section. Open up “Part 4: Estimating the Mass from Multiple Distances”, and view the 4 blocks within it:

- Simulate the Distance from the Earth to Uranus (*Code Block*)
- Response 6 (*Text Block*)
- Heatmap of Possible Masses vs Possible Distances (*Code Block*)
- A Different Distribution (*Text Block*)
- Heatmap of Masses vs Distances for a Normal Distribution (*Code Block*)

- Response 7 (*Text Block*)

Simulate the Distance from the Earth to Uranus

View this histogram output from this block and use it to answer response block 6.

Response 6

Read this text block and answer the numbered questions by editing the text block to write your answer below the horizontal line.

Heatmap of Possible Masses vs Possible Distances

The top three plots are simplified views to break down what we did: first the histogram from the previous code block, then the histogram of the resultant masses, then a scatter plot matching the distances to the masses. Then, a larger heatmap which shows the density of points in a given square of the grid by color. [Use the dropdown to add error](#) to the data, and use what you see to answer response block 7.

A Different Distribution and Heatmap for a Normal Distribution

For comparison, the process of the monte carlo simulation is repeated again for a different distance distribution. The heatmap and three plots are the same as for the previous distribution. [Use the dropdown to add error](#) to the data, and use what you see to answer response block 7.

Response 7

Read this text block and answer the numbered questions by editing the text block to write your answer below the horizontal line.