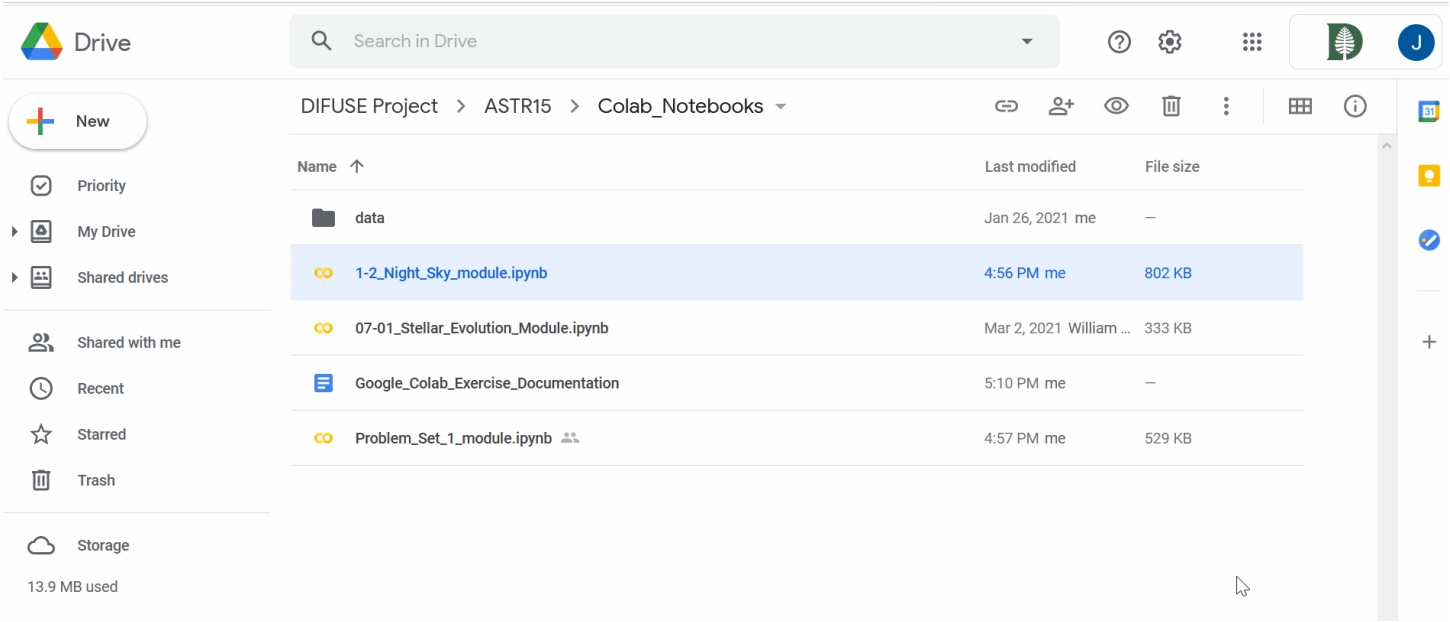
# Documentation for Google Colab Exercises

*Instructions for using Google Colab Notebook*

**Google Colab** is a tool which can be used to run Jupyter Notebooks or create stand-alone notebooks for writing, editing, and running code in Python. The Colab Notebooks are run on Google's servers, which means you don't have to install anything on your own computer and also gives you access to GPU support!

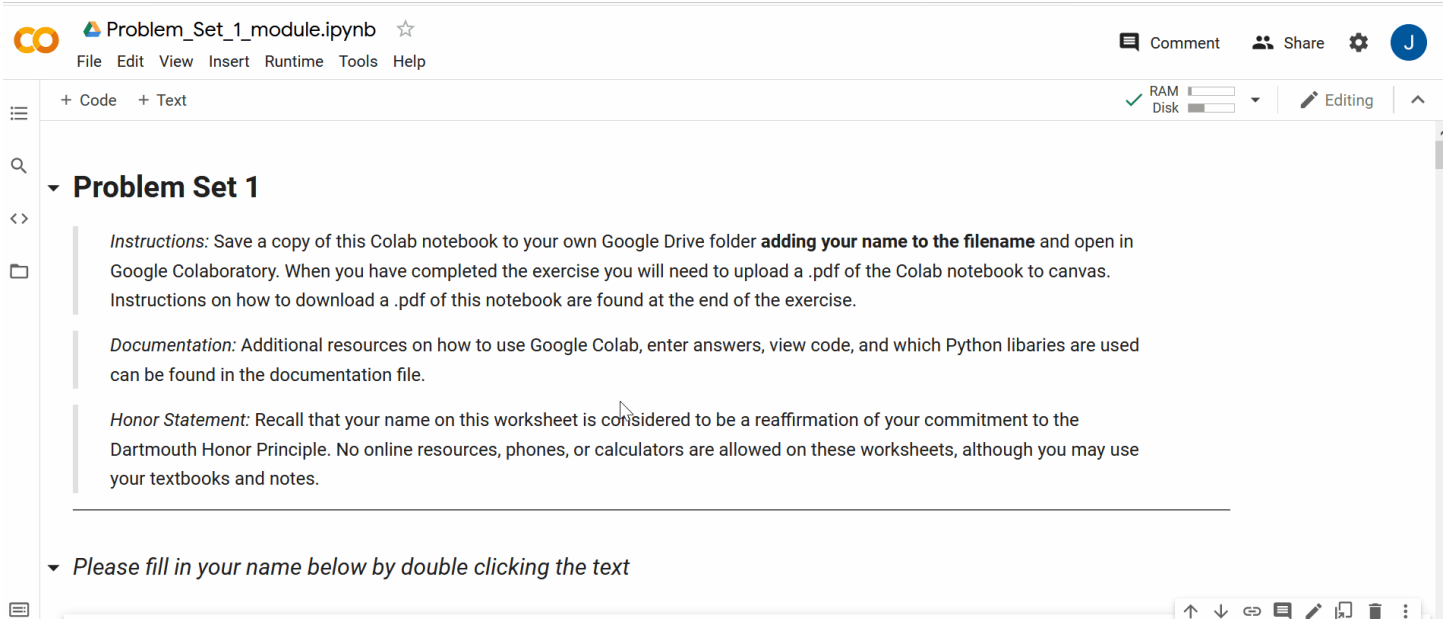
**Opening the Notebook in Google Colab**

- First, you must save a copy in your own Google Drive folder
- Then double click on the .ipynb file and select "Open with Google Colaboratory"



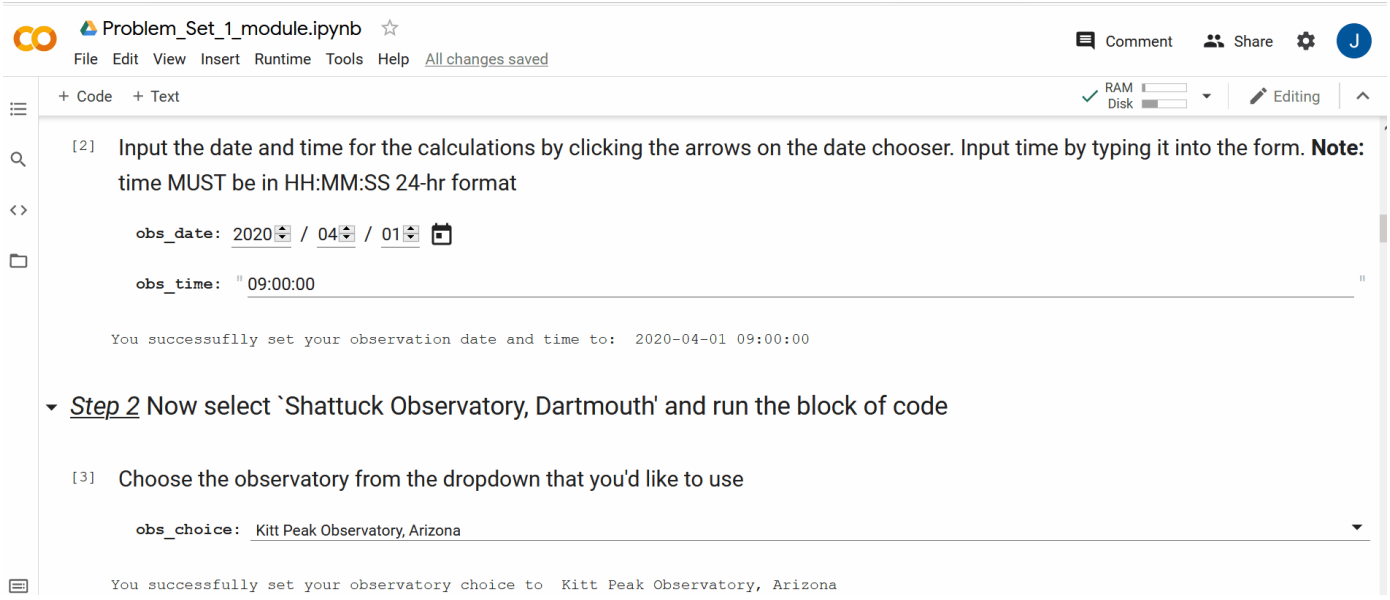
**Blocks consist of *text* or *code*:**

- If you want to edit a text block (e.g., for answering a question), simply double-click the block or click “SHOW CODE.”
- Similarly, if you want to edit a code block or see hidden code, double-click the code block. You can also hide/show the code by right clicking on the block, selecting “Forms,” and then clicking “show code” or “hide code.”



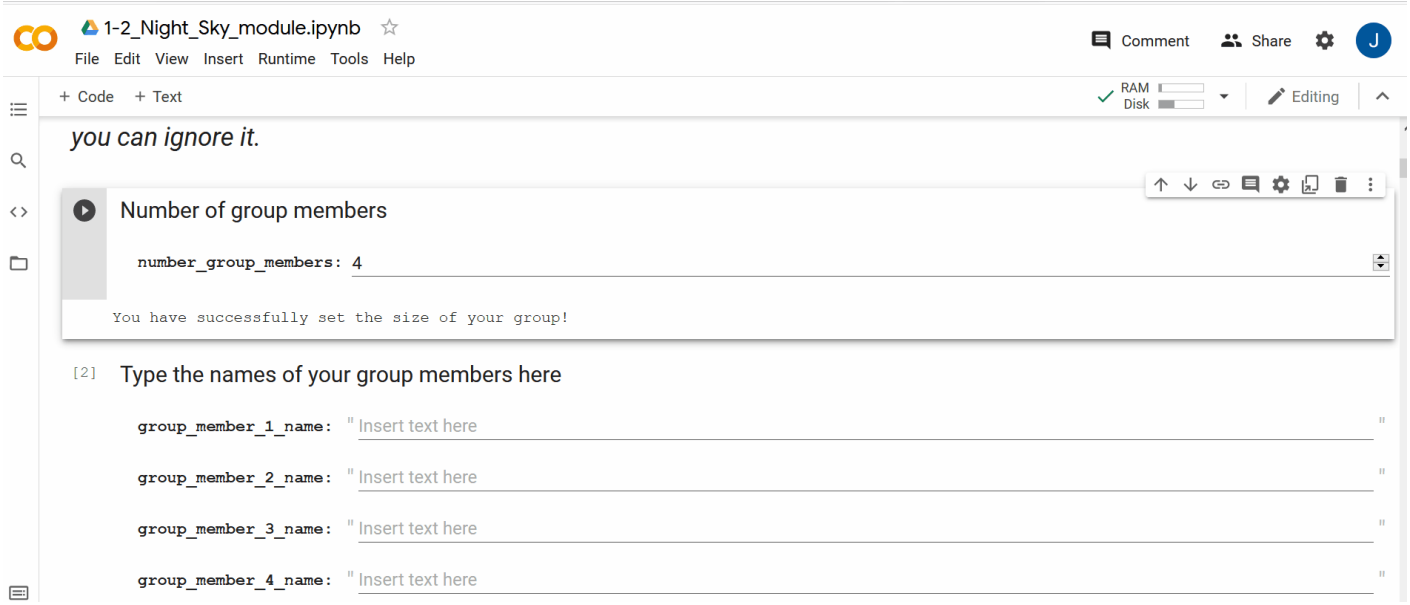

**Running blocks of code:**

- Blocks of code have a play button on the left hand side - to run the code, just press the play button, or you can select the block and press Shift+Enter.
- If you would like to run all blocks of code for the entire Notebook, you can select the Runtime dropdown menu at the top of the screen and “Run all” or “Restart and Run all”



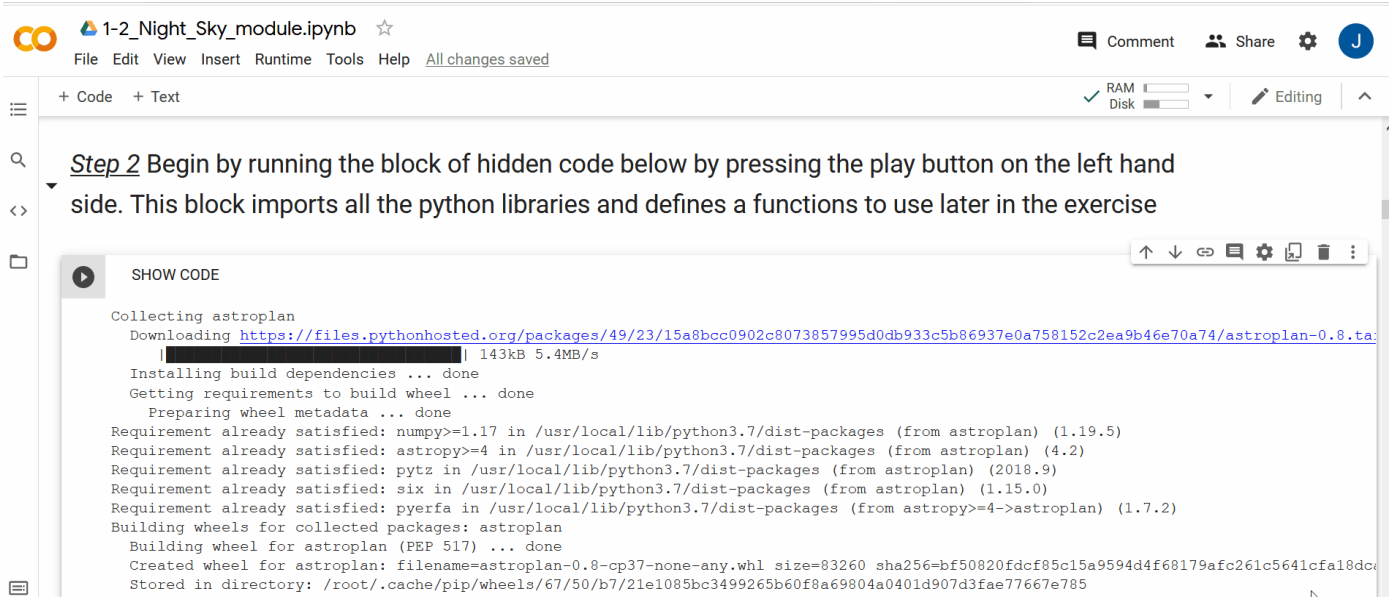
**Inputting information in Forms:**

- Some of the blocks make use of Forms, which consist of an interactive interface that is meant for the user to input text, numbers/dates, or select from a list of options
- For *text input,* click on on the empty line and type in the necessary information
- For *number or date inputs,* arrows allow you to change the number/date or you can manually type in the necessary information
- For *dropdown menus*, simply click and select the option you'd like to choose
- **NOTE:** once you have input the information into the forms, ***<u>you must run the block of code</u>*** for the information you input to be stored as variables and for the functions to be performed



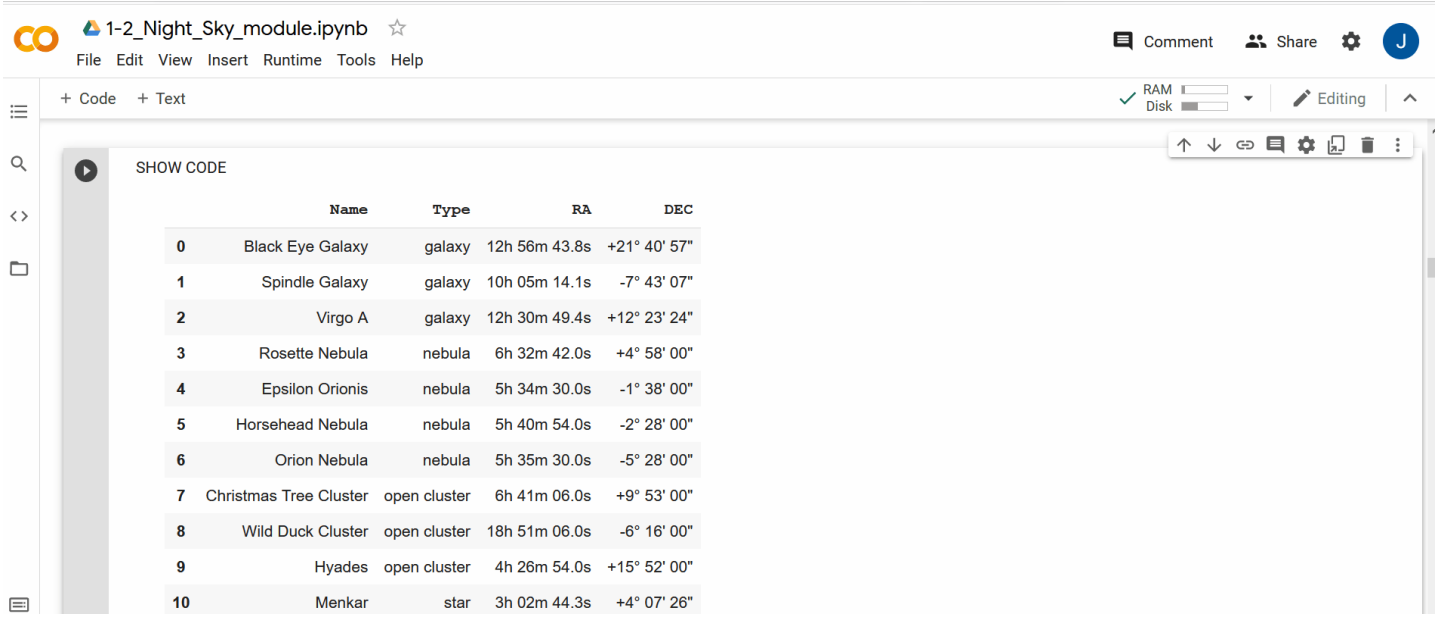
**Examining hidden code:**

- All of the code is hidden in this Colab Notebook to make the document easier to read and look through while completing the exercises
- If you'd like to examine the code, click the "SHOW CODE" button
    - *Be careful not to accidentally modify the code or you might end up with an unexpected error or "bug"*
- To hide the code again, right click the block, go to Forms, and select "Hide code"



**Dealing with Errors:**

- If you enter data or information into forms which are in an unacceptable format or are out of the range of acceptable values you will see an error
- Do not fear! Look carefully at the information you have entered and look at the error message for clues about what may have triggered it
- If you believe the error might be because you have accidentally altered the code, try downloading a new version of the exercise and start over




| | Name | Type | RA | DEC |
|---|---|---|---|---|
| 0 | Black Eye Galaxy | galaxy | 12h 56m 43.8s | +21° 40' 57" |
| 1 | Spindle Galaxy | galaxy | 10h 05m 14.1s | -7° 43' 07" |
| 2 | Virgo A | galaxy | 12h 30m 49.4s | +12° 23' 24" |
| 3 | Rosette Nebula | nebula | 6h 32m 42.0s | +4° 58' 00" |
| 4 | Epsilon Orionis | nebula | 5h 34m 30.0s | -1° 38' 00" |
| 5 | Horsehead Nebula | nebula | 5h 40m 54.0s | -2° 28' 00" |
| 6 | Orion Nebula | nebula | 5h 35m 30.0s | -5° 28' 00" |
| 7 | Christmas Tree Cluster | open cluster | 6h 41m 06.0s | +9° 53' 00" |
| 8 | Wild Duck Cluster | open cluster | 18h 51m 06.0s | -6° 16' 00" |
| 9 | Hyades | open cluster | 4h 26m 54.0s | +15° 52' 00" |
| 10 | Menkar | star | 3h 02m 44.3s | +4° 07' 26" |

## *Python Libraries Used (linked to webpages for additional documentation)*

**Pandas**: easy to use data organization, analysis, and manipulation tool. Primarily used for creating, viewing, and accessing *dataframes*, which is just a Pandas version of a table.

**Numpy:** one of the fundamental packages for scientific computing with Python. Primarily used in the creation and manipulation of *arrays*, which are just matrices.

**Matplotlib:** a fundamental package for plotting and visualization of data in Python. All the plots are constructed using matplotlib.

**Datetime:** a Python module for manipulating dates and times in Python. Used for performing operations on dates and times (makes time and date math a lot easier!).

**Astropy:** the primary Python library for everything Astronomy! We use Astropy for:

1. Retrieving the RA and Dec of celestial objects
2. Calculating the Altitude and Azimuth of celestial objects at different times based on location

3. Creating and manipulating *SkyCoord* objects: high-level objects that provide a flexible interface for celestial coordinate representation, manipulation, and transformation between systems
4. Creating and manipulating *EarthLocation* and *Observer* objects: used for storing a location on the Earth and for storing information about an observer's location and environment, respectively
5. Creating and manipulating *FixedTarget* objects: contain coordinates and metadata for a celestial object that is "fixed" with respect to the celestial sphere
6. Creating and manipulating *Time* objects: used to represent and manipulate times and dates for astronomy, similar to the Datetime module but specific to the ways time is represented in Astronomy.
7. Creating and manipulating *Time* objects: used to represent and manipulate times and dates for astronomy, similar to the Datetime module but specific to the ways time is represented in Astronomy
8. The *Lomb-Scargle Model*: used to fit a sinusoidal model to data at each frequency, with a larger power reflecting a better fit. With this in mind, it is often helpful to plot the best-fit sinusoid over the phased data.

**Astroplan:** a library built on top of Astropy that is specifically for planning the observation of celestial objects. Astroplan is primarily used for creating the "SkyPlots" which are a 2-D representation of the night sky for a given location.

1. Creating the "SkyPlots" which are a 2-D representation of the night sky for a given location
2. Creating the Airmass plots, which are an estimation of the relative distance of atmosphere light from a celestial object has to pass through to reach the surface of the Earth

**Astroquery:** a library built on top of Astropy for querying astronomical web forms and databases. We use Astroquery for retrieving images of your selected objects from an online database.

**Sympy**: a library for symbolic mathematics in Python. We use Sympy to formulate and solve a fairly simple algebraic equation but can be used in many more complex ways for solving a wide range of mathematical problems.