

Problem Set 1

Due Friday, September 7 at 10am via Moodle

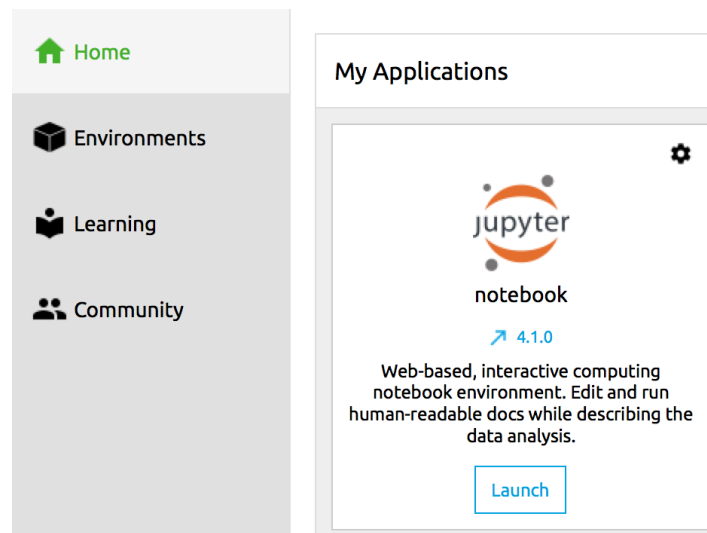
Setup - Installing the Anaconda Python Distribution

Anaconda is the world's most widely-used data science platform for the Python programming language. It is a "package manager" for Python, i.e., it has (nearly) all of the software packages we will need in ASTR 228 built into it. It also makes updating existing programs and installing new ones much simpler.

The download and installation is fairly straightforward. The package is free and available for Windows, Mac, and Linux computers:

<https://www.anaconda.com/download/>

Once Anaconda is installed, it will create a desktop icon named "Navigator". Open this program (it may take a few minutes to run the first time). You are now ready to use the Jupyter notebook, by clicking on the "Launch" button. The Jupyter notebook will then open as a new tab or window in your computer's default web browser.



Exercise #1: First Steps in Python

To work through Exercise #1, first download the file "Exercise1.zip" from the course website at www.katefollette.com/teaching/courses/ast228f18/homework

Once you've downloaded and unzipped this file on your computer, you can navigate to it through the Jupyter notebook interface within your browser. Once there, click on "Exercise1.ipynb" and follow the tutorial. Within your browser, it should look like this:

portions of this notebook were adapted from [Mark Krumholz's public course notes](#)

Lab 1 - First Steps in Python

Lab 1 Contents

- Jupyter
 - Jupyter Intro
 - Jupyter cautions
- Using Jupyter as a calculator
- Variables
- Arrays
 - Defining arrays
 - Array manipulation
 - Multidimensional arrays
 - Array attributes

Each section 1-4 of this notebook contains an exercise that you must complete.

1. Jupyter

1.1 Jupyter Intro

Jupyter notebooks have two kinds of cells. Markdown cells like this one have no labeling to the left of the cell, and, when executed, appear as ordinary text. Code cells like the one below have In []: printed to the side of them and, when executed, a number appears inside of the brackets. To execute a cell of either type in Jupyter, hit shift + enter inside of that cell. Try it by double clicking on this cell and hitting shift + enter, and then do the same with the code cell below. Note that a markdown cell becomes pure text when you execute it, while a code cell spits out some output labelled Out []:

```
In [ ]: 1+2
```

The guts of Jupyter is the coding language Python, and any and all Python syntax will work inside of code cells. However, Jupyter code cells also have some extra capabilities, which we will talk about as needed in the future.

You can now walk through the tutorial and run code within the notebook.

Exercise #2: Basic Plotting in Python

As before, download the zip file from the course website and save it to a directory on your computer, then navigate to that directory within a Jupyter notebook.

Once the file is unzipped, you should see the "Exercise2.ipynb" notebook and be able to open it within your browser:

2. Basic Python Plotting

To make graphs and figures in Python, we will use a plotting library called *matplotlib*. Python (unlike many languages) generally does a lovely job with coloring, line thickness etc. with simple plot commands. It does not, however, add titles, axis labels, legends, etc. and these are **very** important things to include. From now on, any plots that you make in Labs or homeworks should always, at a minimum, include: axis labels (including units), a plot title and a legend in any case where there's more than one line on the same plot.

There are many useful optional inputs to the plot command that allow you to tweak the appearance of the plot, including: linestyle, color, placement of the legend, etc.

So let's learn the basics by plotting some things.

We are going to do this more properly by importing the matplotlib library's plotting module `pyplot` and then telling the notebook that you still want it to display any plots inline (inside the notebook) with the magic function `%matplotlib inline` with the following lines

```
In [ ]: import matplotlib.pyplot as plt
        %matplotlib inline
```

This gives you access to all of pyplot's functions in the usual way of calling modules (`plt.functionname`). For example:

```
In [ ]: x=arange(-10,10,0.01)
        y=x**2
        plt.plot(x, y)
```

Here are some especially useful pyplot functions, called with `plt.functionname(input(s))`:

Once completed, zip Exercise1.ipynb and Exercise2.ipynb together into one file and submit them both to the course Moodle site by the start of class on Friday, September 7.