

*MTR 3420??*

*Python Lab*

*Dr. Schuenemann*

Welcome to the Python Atmospheric Dynamics Lab!

This lab will run co-currently with your homework and is designed for you to put the equations that you’ve learned in class to work as we calculate and plot variables derived from model reanalysis data in Python!

What the goals are for this lab:

* Learn some basics of Python and intro to specific advanced topics in Python
* Read in archived Geopotential Heights data from GFS model reanalysis netCDF file and write a new netCDF file
* Make functions to calculate **Pressure Gradient Force**, **Coriolis Parameter**, **Geostrophic Winds**, and **Absolute Vorticity** at 500mb
* Plot **Geopotential** **Heights**, **Geostrophic Winds,** and **Absolute Vorticity** at 500mb

This lab will consist of \_\_% of your total grade, but if you fall behind, please ask for help; it is not meant to be unattainable!

You will be graded on your attendance and completion of the lab, but most importantly, **your comments in your code**. A lot of the code you have will not be your own and will have comments in it already. Feel free to add more of your own if it helps solidify any questions you have; that is optional. However, any code you make (functions, arrays, etc) you must write it as if you were stepping away from it for an extended amount of time and be able to come back and run it from your comments! This also helps us decipher your thoughts and make sure the topics are being understood.

Don’t do the bare minimum here; I want you thinking outside of your traditional mindset and I want you to start thinking in terms of projects. Programming may be in your future, so let’s develop these good habits now.

Intro Notebook

* Math – NumPy
* Jupyter Notebooks
  + Cells
  + Comments
* Data types
  + Integer
  + String
  + Float
  + Array
* Plotting – CartoPy/Matplotlib
* Loops
  + For
  + If
  + While
* Arrays
  + Idea of
  + Slices

Plotting Notebook

* Geopotential Heights
* Geostrophic Winds
  + 3 different times
    - 1 time plot u and v components
* Absolute Vorticity
  + 3 different times

Data Notebook

* PGF at Denver
* Coriolis Parameter
* Geostrophic Winds
* Absolute Vorticity

Python is designed to be a scripting language that you customize with different *packages*. Each package contains specific functions that you can call in your script instead of writing them in your script. This becomes very advantageous in both space and debugging, not to mention they aren’t a bunch of libraries you don’t need! All the libraries you will need have been installed on the lab computers, **but not your laptop**.

We will be working with an interactive Python environment called Jupyter, which is a massive blend of several coding languages that all come together to offer something very unique; *Jupyter notebooks*. A Jupyter notebook allows the user to run interactively, and in individual **cells**. Each cell can be run on its own, separately from all the other code in the notebook. This is a wonderful diagnosis tool and testing bed for code. In our case, this is very handy as we will be making functions, setting arrays (matrices) with data, and plotting, all of which will inevitably break while we learn.

You will be given 3 notebooks to work in:

* Intro to Python notebook
* Data and Calculation notebook
* Plotting notebook

Each notebook will have certain things already set up in them and space to work on things like data arrays and functions.

The intro notebook will consist of getting to know the Python *syntax* – how the code is set to run – and playing with simple things such as different data types, math functions, and *loops*. This notebook will be done on your own before the first lab class so we can try and hit the ground running.

The data and calculations notebook will be used to draw in the data from a netCDF file, manipulate it, and create functions to calculate our PGF, Coriolis Parameter, Geo Winds, and Vorticity and store these values in a new netCDF file.

The plotting notebook will be used to draw in your new netCDF file data (so you don’t have to keep calculating your variables every time you open your notebook) and plot the data!

# Intro to Python Notebook

This notebook will not be graded, but your grade may depend on it in the long run since it will help introduce relevant topics.

There are comments in the code so try your best to follow those and if you still have questions make sure to address them **before** the first lab class.

We will also make use of limited and simple command line commands and basic computer navigation.

\*\* Have the students download the *.nc* file and have code for them to open it to ensure they have they file and can open it before first lab class \*\*

We are using Linux machines in the lab, as opposed to Mac which uses OSX and PC’s that use Windows. As with any operating system, we can find/save files based on their *path* on the computer in Python.

The main level on your computer is the **home** folder or directory. Within that other folders/directories exist. One is your user account, i.e. **jrichlin**. So, the path to that folder and its contents is “/home/jrichlin/”. In our user account we have other folders like **Desktop** or **Downloads**. Those paths would look like /home/jrichlin/Desktop/ or /home/jrichlin/Downloads/ respectively.

When you download the *.nc* file, the path on your computer where is located matters because we need to specify that in our code when we try to read it. Thus, we would have Python look to the *path* /home/jrichlin/Downloads/ and in that folder/directory would be the download *.nc* file.

Code and things to turn in:

* Nothing

# Data and Calculations Notebook

Ah, here comes the meat and potatoes of your lab. We will be spending a lot of time in this notebook, so much you will probably be dreaming about it! (Hopefully not…)

Let’s jump right in with the data file we will pull the Geopotential Heights from. In the atmospheric sciences, many different file types are used to store data. In IDV we use GEMPAK (*.gmk*) files, GFS and other model data is in GRIB (*.grb*), METEOSAT satellite is in HDF5 (*.hdf*), and GOES satellite data is in netCDF (*.nc*), just to name a few. The netCDF file is what our reanalysis data from NCEP is in.

The NCEP reanalysis data is for the entire globe for the full 2016 year, 4x daily. It has geopotential heights as its main variable along with the time steps, latitudes and longitudes. The *.nc* file will have metadata (info about the data set) and data (actual values). Each one of the 4 variables mentioned is stored as a *variable* in the *.nc* file.

First you must make sure you have downloaded the *hgt.2016.nc* file from Blackboard. This would’ve automatically saved to your *Downloads/* folder.

As we saw with the Intro Notebook, we can save almost anything in Python to a *variable*. This will become something you do over and over again, so make such you understand exactly what you’re doing when you *declare* a variable.

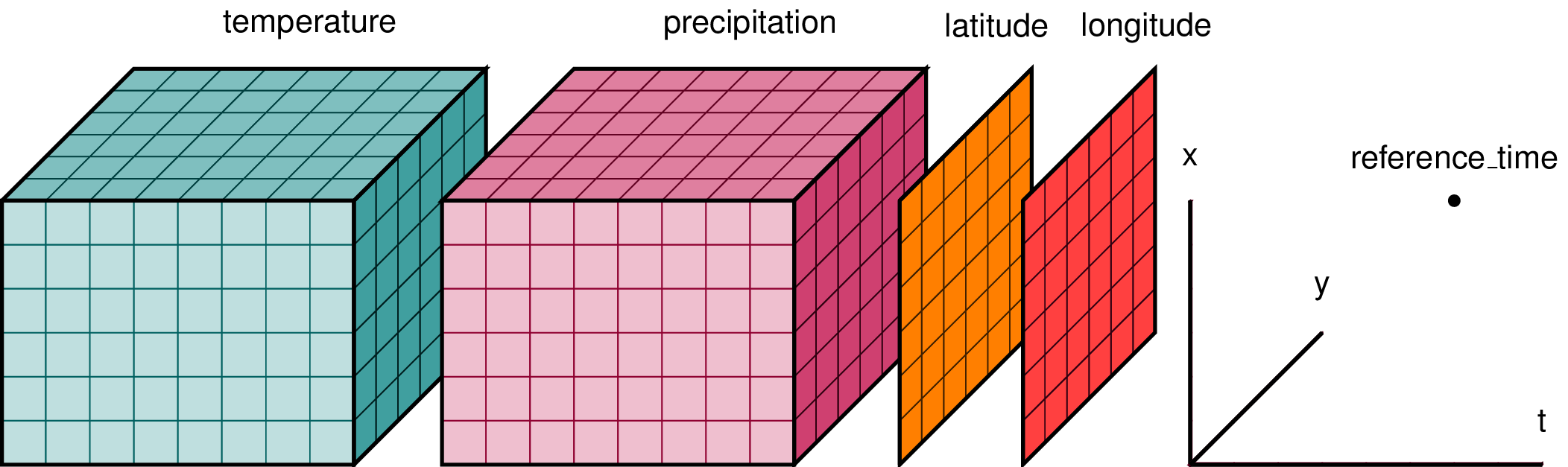
## To Do Checklist

* Open the *.nc* file using what you did in the intro notebook and save the information as a Python *variable*, ***ds***
* Explore the metadata and the variables aka *keys*
* Use the *user input function* to find the **index number** of Denver’s lat and lon in their respective arrays. Do the same thing with the 500mb pressure level. Finally, figure out which number the time 12Z of February 2nd corresponds to. Hint: the 00Z hour of January 1st is index number 0, and each day has 4 times 00, 06, 12, and 18Z.

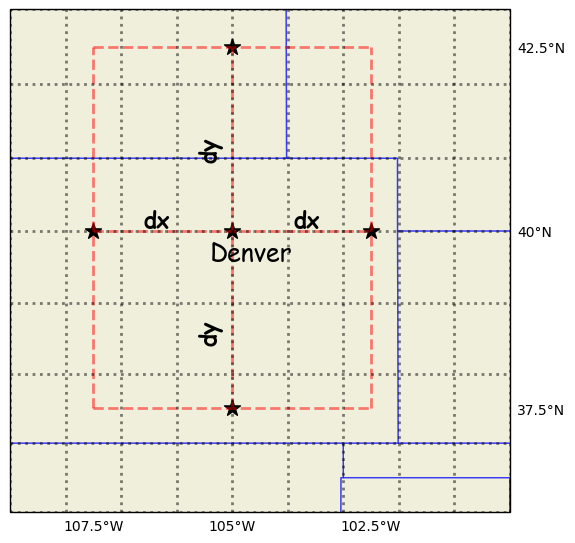
Call these variables:

* Denlat\_index
* Denlon\_index
* mytime\_index
* mylev\_index
* Asf

You will learn to pull that data and store it in a Python *array*. An array can be thought of simply as a multidimensional matrix, or more simply: a chest of drawers.



When we think about taking a value at a point with our calculations, we need to take the *difference* of Geopotential Height *around* that spot, say around Denver, in every direction dictated by the equations. For the gridded data, each data point is both longitude (x) and latitude (y), and the data point is from the previous point, for both directions. See the image below:



\*\* Python reads the y-direction from top to bottom, so we’ll need to address this in our functions!!\*\*

So, the and and for the gridded data, and . So, what are and ? You’ll need them for your calculations!

### Pressure Gradient Force (PGF)

w/

Code and things to turn in:

* Functions
* Data arrays
* New netCDF file with calculations
* Comments!!!

For the PGF we are looking to take the change of heights around Denver:

and