Synoptic Meteorology I: Assignment #1

Due: 18 September 2018

Learning Objectives: Learn how to install a scientific programming language on your computer; learn how to load a programming tool that allows you to type in code and instantaneously see the results of the directive; and learn basic Python 3.x syntax and command structures.

Most assignments this semester focus on learning basics associated with the Python programming language that is rapidly gaining popularity in the atmospheric sciences, with emphasis given to the tools that most readily facilitate synoptic analysis. In this assignment, you are to install the popular Anaconda Python 3.6 distribution on the Atmo Lab (EMS W434) Macs and use it to run several basic commands.

Important note: *The College of Letters and Science, which manages the Atmo Lab, does not allow us to install software across the network so that it works for all users on all machines. This means that the machine on which you complete this assignment is the one on which you will complete all programming-related assignments this semester! Alternatively, you can install Anaconda Python on your personal Windows (remove #macos from the address below) or Mac machine, but you do not need to do this to complete the programming-related course assignments.*

Begin by using one of the web browsers (Chrome, Firefox, or Safari) to open the Anaconda Python download page (https://www.anaconda.com/download/#macos). Click on the leftmost of the green Download buttons to download the Python 3.6 version of the installer. If a pop up asks if you want to register for their newsletter, feel free to dismiss it without signing up.

After the download finishes, click on the downloaded file in the toolbar at the bottom of the screen to launch the installer. In most cases, you can click through the various screens without changing any of the default options. There are two exceptions, however:

- When you reach the Installation Type screen, click on "Change Install Location…" and choose "Install for me only," then click on "Continue."
- When you reach the Microsoft VSCode screen, click on "Continue" without installing the VSCode package.

The installer will take approximately ten minutes to complete. Once it has done so, the system will ask if you want to move the installer package to the trash; feel free to select yes.

After the program has been installed, you will want to create a shortcut to it on your desktop. To do so, double-click on the Macintosh HD icon on the desktop, then enter the Users -> (ePanther ID) -> anaconda3 directory. Control-click on Anaconda-Navigator and choose "Make Alias." Drag the newly created shortcut from the folder to the desktop. To load the software, you simply double-click on the desktop shortcut.

Next, we want to load Anaconda Navigator so that we can enter a Python environment. Double-click on the desktop shortcut you created above. On the first launch, you may be presented with a prompt regarding program use data; select 'Ok, and don't show this again' to continue. By default, Anaconda Navigator opens in the Home pane, which presents you with multiple programs that you can load. We will work in the Spyder development environment. To enter, simply click the Launch button in the Spyder tab. The first time that you load Spyder, it may notify you that a new version is available; uncheck the ‘Check for updates on startup’ box and then move past the notification.

The default Spyder display contains three panes: a large pane on the left half that provides a blank text document into which you can type (and later run) multiple Python commands, and two smaller panes on the right half with an information box (top) and a window in which you can directly type and immediately run Python commands (bottom). In this assignment, we will primarily work with the lower-right pane in this default view. (Note that there are other views for the two right panes: for instance, the top pane can turn into a variable viewer, which is helpful for seeing what variables you’ve defined as well as their types, shapes, and values, whereas the bottom pane can turn into a history of all of the Python commands you’ve typed into the command window.)

For this assignment, you are to get Python to produce the following output:

```
I, (your name here), successfully ran my first Python program at
                 YYYY-MM-DD HH:MM:SS.######!
```

Note that Python code and its output will be depicted in `Courier` font in this and all assignments.

In so doing, you’ll learn some of the basics of how Python variables are declared and typed, how to load modules to provide additional functionality to the basic Python language, and how to write variables and strings out to the screen.

Variables in Python are *dynamically initialized* and *dynamically typed*. In languages such as C++ and Fortran, you usually have to declare a variable and its type (integer, floating-point, string, etc.) before you ever use it. In Python, you don’t have to do this. Thus, let’s say that you want to declare a string variable called `name` that contains your name as a string. This is easy to do, i.e.,

```
name = 'Clark Evans'
```

Although the basic Python language contains great functionality, most scientific work in Python relies on *modules* that provide extended functionality to the Python language. Anaconda Python includes over 200 such modules by default, with many thousands available from other resources. To load a module and all of its functionality, you use the `import` command. For example, to load the `datetime` module, you would use the following command:

```
import datetime
```

In general, the accepted Python syntax for importing modules is to group all such import statements at the top of a program. However, since we are only typing commands directly into Python here, you do not need to start with your module commands; just make sure that the needed modules have all been loaded before you use their functions.

The Spyder iPython console in the lower-right pane offers some useful functionality that helps you to see the functions a module provides and some information about how it is used. For instance, if you want to see what functions the `datetime` module provides, type `datetime.`, then hit Tab on the keyboard. A dropdown menu with the available first-level functions in datetime will appear.

Move down to `datetime.datetime` and hit Return. If you add a `?` to the end of this statement and hit Return, documentation on this function will appear (known in Python as a docstring). Or, if you use the up-arrow to bring back this command, replace the `?` with a `.`, and hit Tab, you will see that `datetime.datetime` also has a number of functions associated with it. Of note, the datetime.datetime.now() function can be used to get the current time.

Variables in Python can also be initialized to the output from a function. For instance, if you want to initialize a variable called `date` with the result of `datetime.datetime.now()`, you type:

```
date = datetime.datetime.now()
```

The value of any Python variable can be printed to the command window using `print()`, i.e.,

```
print(name)
```

would print whatever is stored in the `name` variable to the screen.

The `print()` function can be used to concurrently print multiple variables as a single string so long as all of the variables are strings. For example, imagine that you have defined two variables (`name` and `date`, with values matching those in the examples above) and wish to print them both at once separated by a space. In general, this can be done as follows:

```
print(name + ' ' + date)
```

where the + operator is used to concatenate, or merge, strings on either side of the operator.

However, if you try this command, you'll find that it doesn't work as you might expect. Instead, you get the following error:

```
Traceback (most recent call last):
File "<ipython-input-13-479d1808338e>", line 1, in <module>
print(name + ' ' + date)
TypeError: must be str, not datetime.datetime
```

Python is generally user-friendly with respect to its error messages. This is one such example. In this case, it is telling us that we have a data type error, that one of our variables in the `print()` statement is not a string but has the `datetime.datetime` data type. Fortunately, Python has a way to *typecast* variables. For instance, the `str()` function can be used to convert selected data types to strings.

Given all of this, you are to use the Python console to load the needed modules, declare the needed variables, and execute the needed print statement to produce the output on the previous page. Once done, take a screenshot (a picture of the screen taken with your smartphone will suffice) and print it (to submit in class) or send it to me via e-mail (prior to the start of class on the due date).