# Program files and Spyder

Carver J. Bierson

June 26, 2017

## 1 Introduction

For this class you will turn in homeworks as python files. These are just text files with a .py file extension that can be interpreted by python. Because of this you can write your code in any text editor of your choosing. For scientific applications I recommend using "Spyder" and so will give some examples how to use it.

## 2 Installing Spyder

The information for Spyder is at `http://pythonhosted.org/spyder/`. Note that is a python 2 an python 3 version. Note that all of the install methods for all OS are free.

- **Windows**: The easiest way is to install Anaconda. This will give you a little more than you need but will be much easier than installing python from scratch.
- **Mac**: I recommend just using the dmg installer on the Spyder site. The anaconda installer should also work if you already have that.
- **Linux**: If you have linux your built in package manager (for debian based or red hat at least) should let you directly install Spyder.

If you have trouble with any of these you can stop by office hours or just use the mac computer lab (where class is held).

## 3 Program Files

This is the main method you will use to turn in assignments. A python file is simply a plain text file with the .py file extension. We use program files to be able to reuse code and rerun it with minor modification. When writing code I recommend using a structure of

1. Start of file Block comments
2. Import libraries
3. Define constants and other variables / Read input data

4. Perform calculations / Process data

5. Make plots / write output data

A key part of your program files is comments. **Any file turned in with no comments will receive a 0!** Comments make your code readable by others and yourself years down the road. Here is where I expect comments.

- Header at the start that describes the program, describes important variables, and says who wrote it when.
- At the start of each section of code (input,calculations,output).
- Each time a value is given stating the units as source as needed.
- Each time a new task is being done (calculating x velocity, now y velocity, etc...)

### 3.1 Example file

```
#!/usr/bin/env python
# The above line is optional but lets the code be run as an
    executable on Unix systems
""" -triple quotes starts a multi-line string. Here I am using
this to mimic a bulk comment
ExampleFile.py - Explain what this file does
Some more detail about what it does
# Written for Python 2.7

FirstVar - the first variable in this program
Var2 - the second variable (give units)

Written by Carver J. Bierson
Created Jun-2017
Modified by Example B. Person, Apr. 2019

"""

import numpy as np

#%% Declare variables

# Here I set FirstVar to 5 (units/s)
FirstVar=5

# Var2 is critically important for XX
Var2=10

#%% Do Calculations

#%% Output answer

# Let the use know I am finished
print("I am done") # Python 3 print syntax
```

Listing 1: Python Example script

## 4 Key Spyder Features

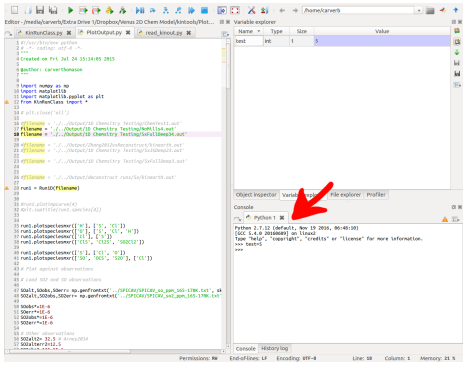

Figure 1: Built in python console.

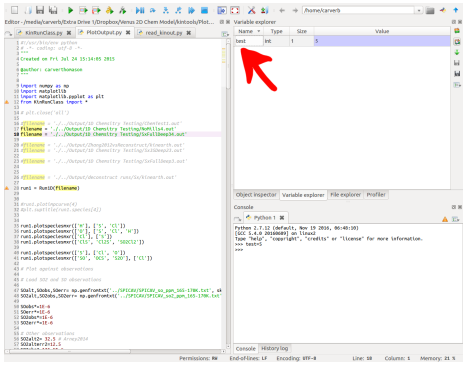

Figure 2: Variables in console workspace.

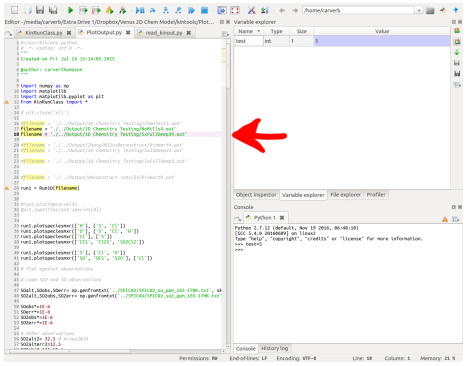

Figure 3: File editor.

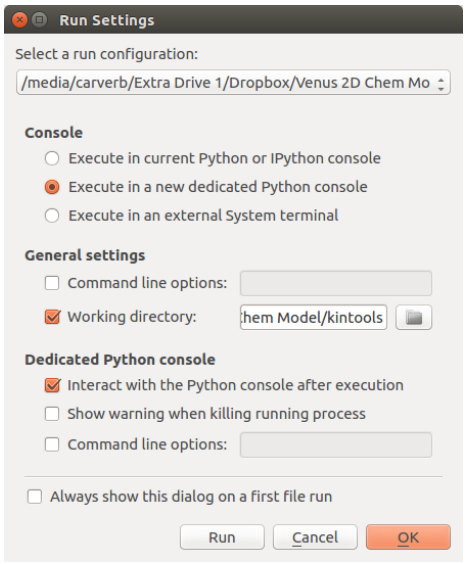


Figure 4: You can access the run options for a file with F6. These are the settings I use for most files. By creating a new terminal for each run and interacting with it after running you can easily check the variables at the end of the run.

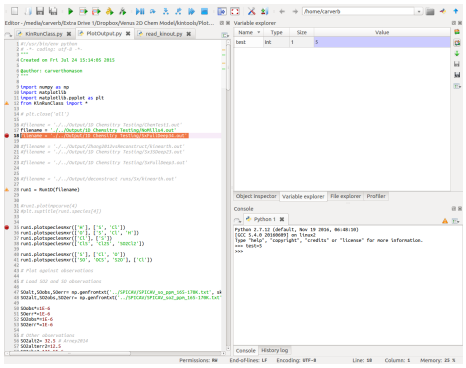

Figure 5: By using the blue play/pause button we can run the file in debug mode. In this case the file will stop at the lines that have a red circle next to them and we can check a variables value. Stop points for debugging are added by clicking next to the line you want to stop at.

## 5 In Class Practice

Write a python script that takes a number as input from the user, than prints out the double and square of that value. *Hint:* lookup the input function.