Python for Managing Your Data

(aka Dealing with Large Data Sets, aka Getting your data in to – and out of – models)

Summary

The aim of this short course is to introduce the use of the Python programming language for tackling day-to-day data management tasks, the kind of thing that I and many other geomorphologists do regularly. This includes:

- Format conversion, and data extraction and manipulation, to get your data out of one software package and into another: e.g. to/from spreadsheets using .csv ("Comma Separated Value") files, conversion between common GIS formats, etc.
- Quick and easy data analysis
- The mass production of many similar graphs.

The approach here is deliberately "work-a-day", **not** covering e.g. fancy user interfaces etc.

Why Python?

Advantages:

- multi-platform
- fast, easy to write (has a fairly unique place in the "ecosystem of programming languages")
- clear error messages
- extensible (very many pre-existing libraries)
- is faster than a simple interpreted language: Python automatically creates machine-readable bytecode from your source code
- modern: some very useful data structures e.g. lists, dynamic arrays.

Disadvantages:

- Python is not as fast in execution as a fully compiled language such as C++.

Before You Start

You will need five items of software:

- Python
- some Python library packages (more details below)
- Spyder, an IDE (Integrated Development Environment) for Python
- an editor for plain text (ASCII) files. Don't use a word processing package
- the BSG Windsor Python example files and associated data files.

It is definitely a good idea to install these **before you get to Windsor**!

To install the first three of these, all you have to do is to install the Anaconda package: https://www.anaconda.com. This is available for Windows, Mac, and Linux. Go to 'Downloads' and get version 3.7, then follow the installation instructions for your operating system:

- https://docs.anaconda.com/anaconda/install/windows/
- https://docs.anaconda.com/anaconda/install/mac-os/
- https://docs.anaconda.com/anaconda/install/linux/

Now verify that Anaconda is working correctly: https://docs.anaconda.com/anaconda/install/verify-install/. You might also want to find out more about Spyder: https://www.spyder-ide.org/

Which plain text editor you use depends on your operating system.

- Windows is supplied with a plain-text editor called Notepad, but this has several limitations (for example, it does a poor job of handling the end-of-line characters in text files created on Mac or Linux machines). A better alternative is Wordpad (https://microsoft_wordpad.en.downloadastro.com/). Or try Googling "notepad replacement".
- For Macs, you can use the pre-installed TextEdit app. To work on a file without any text formatting, go to menu "Format > Make Plain Text" (or shortcut 'Command+Shift+T'); to set this to be the default every time you launch the app, go to "Preferences" > and under 'Format' > choose "Plain Text". However, many people prefer Jedit (http://www.jedit.org/) or BBEdit (https://www.barebones.com/products/bbedit/download.html but must be bought after a 30 day evaluation period).
- There are many plain-text editors for Linux, such as Gedit or KDE's Kate.

The final software item that you will need is BSG Windsor Python example files and associated data files. Go to https://github.com/davefavismortlock/BSG-Python and choose 'Clone or download' then 'Download ZIP'. Unpack the zipfile, and put its contents in a folder **and make sure that you know where this folder is**.

Some Background Reading

Having installed the software, it is well worth browsing through Google's Python tutorial (at least as far as 'lists': the sections beyond that might appear a bit daunting…), see https://developers.google.com/edu/python/. (Note that we will be using Python 3, but (at least for beginners) there is not really much difference between Python 3 and Python 2.) If you have time, also watch Google's Python videos. You might also want to download and try some of Google's Python exercises.

It is well worth downloading and skimming through the excellent (and free) 'How To Automate The Boring Stuff With Python', by Al Sweigart: see https://automatetheboringstuff.com/

The Python Tutorial

- **Example 01**: using Python to read and copy a text file
  - Start Spyder and load ‘example_01_copy_file.py’ from the place where you saved the BSG example files
  - Understand it
    - Header line (ignore this for the moment)
    - Comments begin with #
    - Indentation is used to control Python program structure: this approach is unique to Python. It is **VITAL** that you understand this
  - Run it
    - Look at the input and output files using your plain text editor. Are they identical?
  - Modify the example script:
    - Change name of output file
    - Try putting a # in front of the line in the Python script which writes the end-of-line character, then re-run the script and look at the output file. What has changed?
    - Make some deliberate mistakes, then study the error message(s). Correct your errors, get the script working again

- **Example 02**: save a spreadsheet as a CSV (comma separated variables) file, parse the CSV text file, also use of a Python list
  - Understand it
  - Run it
  - Modify it to write out (say) the third data item

- **Example 03**: as example 2 but write out more than one data item
  - Understand it then run it
  - Modify the output format. At present, the two data items are separated by a couple of tab character. Try writing out in a different format, e.g. comma-separated

- **Example 04**: do some calculations
  - Understand then run it
  - Modify it:
    - Try doing some different calculations
    - Write out more than one calculated data item (note: might need to create a new variable to hold the result of the second calculation)

- **Example 05**: use the NumPy library to bin data
  - If you get an error message re. the ‘import numpy’ line of the script, then you do not have the NumPy library installed

- Read and understand it!
    - Notice how we receive the output of numpy.histogram() in the bins variable
- Run it and look at the output. It is a good idea to read https://docs.scipy.org/doc/numpy-1.15.1/reference/generated/numpy.histogram.html which tells you more about the NumPy histogram() function
- The output is rather ugly. So also try **Example 05a**, which writes the output to a new file. Notice here that we receive the output of numpy.histogram() into several variables

- **Example 06**: removing lines of data that have an error in them
    - Understand it:
        - This is just example_1 with a couple of extra lines. In the first new line, we look for the word "Updating" (this could be any string of text, of course). If it is found, then we skip the rest of the loop and continue with the next iteration of the loop. So only the lines which do **not** contain the word "Updating" get written to the new file
    - Run it
    - Modify it:
        - Try changing the program so that only lines which **do** contain the word “Updating” get written to the file. Hint: could you put something instead of 'continue'?

- **Example 07**: similar to Example 6, but illustrates the use of the == (two equals signs) in a test for equality
    - Understand it:
        - In Python, think of **one equals sign** (i.e. =) as meaning "becomes equal to" and **two equals signs** (i.e. ==) as meaning "is equal".
        - Here, if the second number on the line is equal to 0.0005, then that that line is skipped. So the output file consists only of those lines that **do not** have 0.0005 as their second value
    - Run it

- **Example 08**: plots a simple histogram from a datafile
    - For this you will need to have the MatPlotlib library installed. For more about this, see https://matplotlib.org/

- **Example 09**: plots a fancier histogram, which has a log-scale y axis
    - Try changing some of the options documented at https://matplotlib.org/api/_as_gen/matplotlib.pyplot.html#module-matplotlib.pyplot
    - For example, set log = False, normed = True, and cumulative = True.
    - Also try histtype = 'step'. Try histtype = 'stepfilled' too, etc.

- **Example 10**: also uses the amazing Geospatial Data Abstraction Library (GDAL) to read a

GIS input file, and display a single layer from this file. As well as MatPlotLib, you will need to have GDAL installed (http://www.gdal.org/)

- This program reads the input filename from the command line. First run it with no command-line argument to get a "Usage: ..." message. See how this is done in the Python code?
- Now run it using the example GIS file: in Spyder, go to 'Run' → 'Configuration per file' → tick 'Command line options' then write 'delev29.img' in the box.
- Use a GIS file of your choice.

- **Example 11**: again uses GDAL, this time to produce a histogram from a GIS input file. It also writes the GIS data to a text file, e.g. for further analysis. Similarly to the previous example, this program reads the input and output filenames from the command line, so put 'delev29.img out.txt' in Spyder's 'Command line options' box.
  - Again, try using a GIS file of your choice.
  - How easy would it be to write a new Python program which reads the output from this program?

Final thoughts

- Are these scripts or programs?
- Being a programmer vs being an analyst/programmer
- Designing your own programs
  - Pseudo code vs flowcharts
  - Converting pseudo code to Python
  - Just get started! Does not have to be a complete program to test it, start simple and add to it
- The time trade-off: the hard work is almost all **up-front** when programming
- Where to find help on Python specifics: http://www.python.org/doc/
- Errors and debugging
  - Use the Python command **print** liberally, comment out when no longer needed
  - Oh! and OH!!! errors (i.e. coding and design errors)
- More Python
  - Finding more Python library modules: Python Package Index http://pypi.python.org/pypi (over one million, last time I looked)
    - Try http://docs.enthought.com/mayavi/mayavi/ for 3D graphing in Python
  - There is information about installing Python packages at https://packaging.python.org/tutorials/installing-packages/
- As well as the Google Python course, there are many, many online Python courses (not all free), e.g.
  - This one from CodeAcademy: https://www.codecademy.com/learn/python
  - LearnPython: http://www.learnpython.org/
  - There are also lots of IOS and Android apps for learning Python. Not all of them are useful, unfortunately.
- As well as *How To Automate The Boring Stuff With Python*, there are many other excellent online Python books. The following are by Allen B. Downey:

    - *Think Stats: Probability and Statistics for Programmers* at http://greenteapress.com/thinkstats/index.html
    - *Think Python: How to Think Like a Computer Scientist* at http://greenteapress.com/thinkpython/
    - *Think Complexity: Exploring Complexity Science With Python* at http://greenteapress.com/complexity/
    - A really good specialist Python book is: *Python Geospatial Development*, by Erik Westra (I think it is cheapest from http://www.packtpub.com/python-geospatial-development/book)
- Python blogs are often very informative e.g.
    - *Pythonic Perambulations* at http://jakevdp.github.com/
    - *The Glowing Python* at http://glowingpython.blogspot.fr/

Good luck!

Dave Favis-Mortlock, 29 November 2018