Python for Managing Your Data

David Favis-Mortlock

Environmental Change Institute
University of Oxford

david.favis-mortlock@ouce.ox.ac.uk

Background

- The aim here is to tackle day-to-day data management tasks, the kind of thing that I and 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, and directly
    - 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 of Python
  - Multi-platform: runs on everything from phones to supercomputers
  - Fast and easy to write (has a fairly unique place in the "ecosystem of programming languages")
  - Largely self-documenting
  - **Clear** error messages !!!!
  - Extensible (big pre-existing libraries)
  - Is faster than a simple interpreted language: Python automatically creates machine-readable bytecode from your source code
  - Modern design: some v. useful data structures e.g. dynamic arrays, lists
- Disadvantages of Python
  - Not as fast in execution as a fully compiled language such as C++

From the Wikipedia page on Python

- Python is a general-purpose, high-level programming language whose design philosophy emphasizes code readability. Python claims to "[combine] remarkable power with very clear syntax", and its standard library is large and comprehensive.
- Its **use of indentation** for block delimiters is unique among popular programming languages.
- An important goal of the Python developers is making Python fun to use. This is reflected in the origin of the name (based on the television series Monty Python's Flying Circus), in the common practice of using Monty Python references in example code, and in an occasionally playful approach to tutorials and reference materials. For example, the metasyntactic variables often used in Python literature are spam and eggs, instead of the traditional foo and bar.

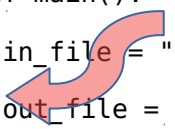
The Tutorial 1

- Me: I'm not a Python expert, by any means. Still learning!
- I am assuming that you have installed:
  - Python
  - some Python libraries ('modules') such as NumPy, MatPlotLib etc.
  - the Spyder IDE
  - a text editor
  - the BSG Windsor example files
- And that you have checked that everything appears to be working correctly.

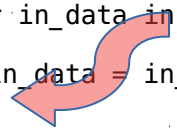
From spreadsheet to CSV file

- We'll start by saving a spreadsheet as a CSV ('Comma Separated Values') file
- CSV files are plain text files, which are very easy to read from and write to using Python
- (It is possible to read from and write to spreadsheet files directly using Python, but this requires a bit more work! See later...)

```
# Example 01

#=============================================================================================================

def main():
  in_file = "some data file.txt"
  out_file = "copy of some data file.txt"

  fp_in = open(in_file, "r")
  fp_out = open(out_file, "w")

  for in_data in fp_in:          # do this section once for every line in the input file
    in_data = in_data.strip()    # remove leading and trailing whitespace

    fp_out.write(in_data)        # write in_data to the output file
    fp_out.write("\n")           # and also write an end-of-line character

  fp_in.close()
  fp_out.close()

  print("File copied, end of run")
#=============================================================================================================

main()
```



Indent, first level



Indent, second level

The Tutorial 2

- Exercise 01 onwards... at your own pace
- I will stop again in 5-10 minutes to say a bit more about Exercise 02 etc.

```
# Example 02

#=====================================================================================================================
def main():
  in_file = "data_file.csv"
  out_file = "new_data_file_1.txt"

  fp_in = open(in_file, "r")
  fp_out = open(out_file, "w")

  for in_data in fp_in:
    in_data = in_data.strip()             # remove leading and trailing whitespace

    data_list = in_data.split(",")        # make a list (called data_list) from the comma-separated data items in in_data

    fp_out.write(data_list[0])            # write out only the first item in the list (which starts with item zero)
    fp_out.write("\n")

  fp_in.close()
  fp_out.close()

  print("File 1 created, end of run")
#=====================================================================================================================

main()
```

```
#=================================================================================================================
def main():
  in_file = "data_file.csv"
  out_file = "new_data_file_2.txt"

  fp_in = open(in_file, "r")
  fp_out = open(out_file, "w")

  for in_data in fp_in:
    in_data = in_data.strip()             # remove leading and trailing whitespace

    data_list = in_data.split(",")        # make a list (data_list) from the comma-separated data items in in_data

    fp_out.write(data_list[0])            # write out the first item in the list (which starts with item zero)
    fp_out.write("\t\t")                  # write two tab characters next
    fp_out.write(data_list[1])            # write out the second item in the list
    fp_out.write("\n")                    # write an end-of-line-character

  fp_in.close()
  fp_out.close()

  print("File 2 created, end of run")
#=================================================================================================================

main()
```

```
#=================================================================================================================
def main():
  in_file = "data_file.csv"
  out_file = "new_data_file_3.txt"

  fp_in = open(in_file, "r")
  fp_out = open(out_file, "w")

  for in_data in fp_in:
    in_data = in_data.strip()             # remove leading and trailing whitespace

    data_list = in_data.split(",")        # make a list (data_list) from the comma-separated data items in in_data

    fp_out.write(data_list[0])            # write out the first item in the list (starts with item zero)
    fp_out.write("\t\t")

    ndata = float(data_list[2])           # ndata is the 3rd item in the list, in the form of a floating point number
    ndata = ndata * 35.567

    fp_out.write("%7.6f" % ndata )        # write out the newly-calculated number
    fp_out.write("\n")

  fp_in.close()
  fp_out.close()

  print("File 3 created, end of run")
#=================================================================================================================

main()
```

```
import numpy

#=============================================================================================================
def main():
  in_file = "data_file.csv"
  fp_in = open(in_file, "r")

  ndata_list = []

  for in_data in fp_in:
    in_data = in_data.strip()             # remove leading and trailing whitespace

    data_list = in_data.split(",")        # make a list (indata) from the comma-separated data items

    ndata = float(data_list[2])           # ndata is the third item in the list, in the form of a floating point
number
    ndata = ndata * 10

    ndata_list.append(ndata)

  fp_in.close()

  # we have our list of data ready, so do some binning
  # see https://docs.scipy.org/doc/numpy-1.15.1/reference/generated/numpy.histogram.html
  number_of_bins = 6
  bins = numpy.histogram(ndata_list, number_of_bins)

  print(bins)

  print("Calculated stats, end of run")
#=============================================================================================================

main()
```

Example 05a

```
import numpy

#=============================================================================================================================
def main():
  in_file = "data_file.csv"
  fp_in = open(in_file, "r")

  ndata_list = []

  for in_data in fp_in:
    in_data = in_data.strip()             # remove leading and trailing whitespace

    data_list = in_data.split(",")        # make a list (data_list) from the comma-separated data items in in_data

    ndata = float(data_list[2])           # ndata is the third item in the list, in the form of a floating point number
    ndata = ndata * 10

    ndata_list.append(ndata)

  fp_in.close()

  # we have our list of data ready, so do some binning
  # see https://docs.scipy.org/doc/numpy-1.15.1/reference/generated/numpy.histogram.html
  number_of_bins = 12
  (bins, bin_edges)  = numpy.histogram(ndata_list, number_of_bins)

  out_file = "binned_data.txt"            # setup and open the output file
  fp_out = open(out_file, "w")

  print_text = []                         # create an empty list
  for i in range(len(bins)):              # go through the bins array
    thisLine = str(bin_edges[i])          # and create a string containing this value of bin_edges
    thisLine += ",    "                   # append a comma and some spaces to the string
    thisLine += str(bins[i])              # append this value of bins to the string

    print_text.append(thisLine)           # append the string to the print_text list

  print_text.append(str(bin_edges[-1]))   # after the loop, append the last value of bin_edges to the print_text list

  for line in print_text:                 # go through the print_text list, item by item
    fp_out.write(line)                    # and write this item to the output file
    fp_out.write("\n")                    # write an end-of-line character to the output file

  fp_out.close()

  print("Data binned, end of run")
#=============================================================================================================================

main()
```

```
#=============================================================================================================
def main():
  in_file = "some data file.txt"
  out_file = "error-free data file.txt"

  fp_in = open(in_file, "r")
  fp_out = open(out_file, "w")

  for in_data in fp_in:
    in_data = in_data.strip()             # remove leading and trailing whitespace

    if in_data.find("Updating") >= 0:     # look for the word "Updating" in in_data
      continue                            # if it is found, continue with the next iteration of the loop

    fp_out.write(in_data)                 # not found, so write out in_data
    fp_out.write("\n")

  fp_in.close()
  fp_out.close()

  print("Errors removed, end of run")
#=============================================================================================================

main()
```

```
#=============================================================================================================
def main():
  in_file = "data_file.csv"
  out_file = "no_0.0005_data_file.txt"

  fp_in = open(in_file, "r")
  fp_out = open(out_file, "w")

  for in_data in fp_in:
    in_data = in_data.strip()             # remove leading and trailing whitespace

    data_list = in_data.split(",")        # make a list (data_list) from the comma-separated data items

    ntest = float(data_list[1])           # read the second item in data_list as a floating-point number
    if ntest == 0.0005:                   # is this number equal to 0.0005?
      continue                            # it is, so skip the rest of the loop

    fp_out.write(data_list[0])            # it isn't, so write out the first item in the list
    fp_out.write("\n")

  fp_in.close()
  fp_out.close()

  print("Errors removed, end of run")
#=============================================================================================================

main()
```

```
import matplotlib.pyplot as plt


#=============================================================================================================================
def main():
  in_file = "data_file.csv"
  fp_in = open(in_file, "r")

  ndata_list = []

  for in_data in fp_in:
    in_data = in_data.strip()             # remove leading and trailing whitespace

    data_list = in_data.split(",")        # make a list (indata) from the comma-separated data items

    ndata = float(data_list[2])           # ndata is the third item in the list, in the form of a floating point number
    ndata = ndata * 1000                  # make the numbers a bit bigger (so it is easier to read labels on graph)

    ndata_list.append(ndata)

  fp_in.close()

  n, bins, patches = plt.hist(ndata_list, 50)   # set up a histogram using data_list, with 50 bins
  plt.show()                                    # and plot it

  print("Data graphed, end of run")
#=============================================================================================================================

main()
```

```
import matplotlib.pyplot as plt

#=====================================================================================================================
def main():
  in_file = "data_file.csv"
  fp_in = open(in_file, "r")

  ndata_list = []

  for in_data in fp_in:
    in_data = in_data.strip()             # remove leading and trailing whitespace

    data_list = in_data.split(",")        # make a list (indata) from the comma-separated data items

    ndata = float(data_list[2])           # ndata is the third item in the list, in the form of a floating point number
    ndata = ndata * 1000                  # make the numbers a bit bigger (so it is easier to read labels on graph)

    ndata_list.append(ndata)

  fp_in.close()

  # set up a histogram using data_list and various options
  n, bins, patches = plt.hist(ndata_list, 50, normed = False, cumulative = False, histtype = 'bar', log = True, facecolor =
'green', alpha = 0.75)
  plt.xlabel('Runoff')                    # label the x axis
  plt.ylabel('Frequency')                 # label the y axis
  plt.grid(True)                          # put a grid on it
  plt.show()                              # plot the graph

  print("Data nicely graphed, end of run")
#=====================================================================================================================

main()
```

The Tutorial 3

- OK, I'll come clean: you are not going to become an expert in Python – or even just proficient – by working through this tutorial.
- But it will, hopefully, give you an idea of how even small, easily-written Python programs can be very valuable for everyday data management tasks.

Python snippets 1

```
import xlrd       # see http://xlrd.readthedocs.io/en/latest/api.html

[…]

def main():
   # open the spreadsheet
   ssfile = xlrd.open_workbook("/home/davefm/documents/Research/0 To Do/01 JB Karoo erosion
    pins/Spatial/Clinometer/All_sites_clinometer_measurements.xls")

   # and get a list of worksheet names (one for each site)
   sheetnames = ssfile.sheet_names()

   sites = []
   for i in range(len(sheetnames)):
      print("Processing " + sheetnames[i])
      s = Site(sheetnames[i])

      # get the worksheet object
      thissheet = ssfile.sheet_by_index(i)
      # now get the cell values, starting in cell B2 (which is the clinometer value for pin A1) and ending in cell F6 (which is
    the clinometer value for pin E5)

      n = 0
      for row in range(1, 6):       # i.e. row B to F
         for col in range(1, 6):    # i.e. col 2 to 6
            s.pins[n].gradient = thissheet.cell_value(row, col)
            n += 1
```

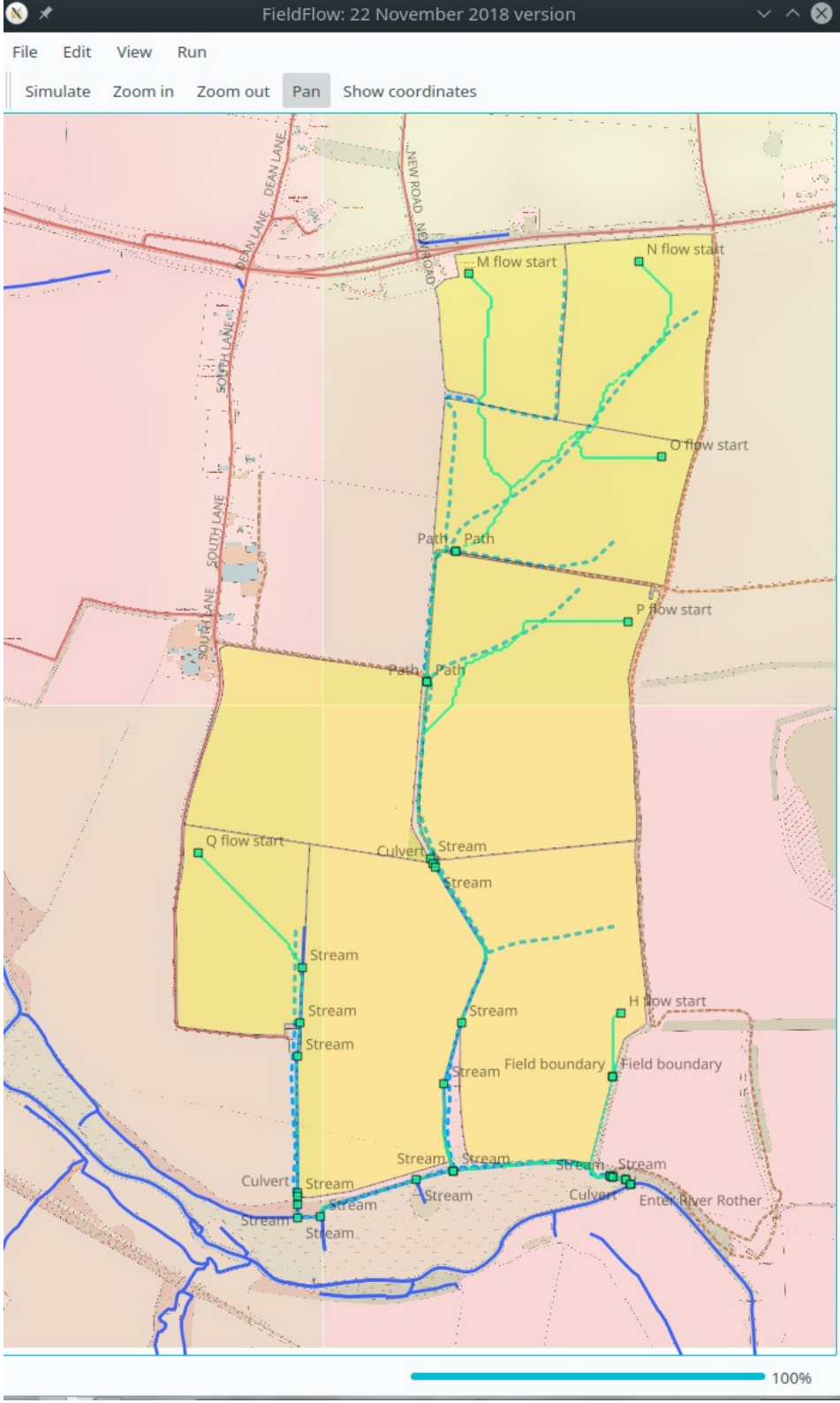
As used in Favis-Mortlock, D.T., Boardman, J., Foster, I.D.L. and Greenwood, P. (2018) ‘Local gradient’ and between-site variability of erosion rate on badlands in the Karoo, South Africa. Earth Surface Processes and Landforms 43(4), 871-883. https://doi.org/10.1002/esp.4293

Python snippets 2

```
#!/usr/bin/python2

from qgis.core import QgsProject
from qgis.gui import QgsMapCanvas, QgsLayerTreeMapCanvasBridge
from qgis.core.contextmanagers import qgisapp
from PyQt4.QtCore import QFileInfo

with qgisapp():
   # note that this must be an absolute path
   project_path = '/home/davefm/Documents/Teaching/Postgrad/BSG
    Windsor/Python_for_Managing_Your_Data/examples/QGIS_examples/TEST.qgs'

   canvas = QgsMapCanvas(None)  # will reparent it to widget via layout

   # load the  project
   bridge = QgsLayerTreeMapCanvasBridge(QgsProject.instance().layerTreeRoot(), canvas)
   QgsProject.instance().read(QFileInfo(project_path))

   # and show the canvas
   canvas.show()
```

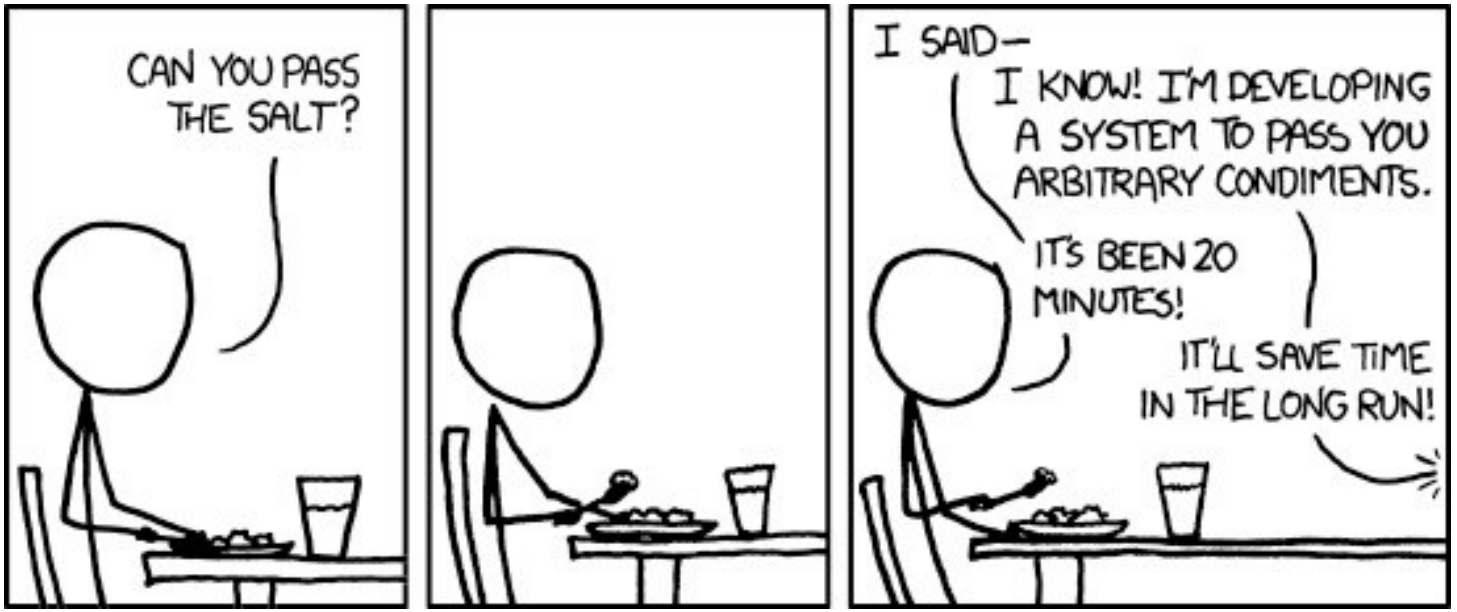
And a larger Python project

‘FieldFlow’, see https://github.com/davefavismortlock/FieldFlow



Final thoughts 1

- Are these scripts or programs? Dunno!
- Being a programmer vs being an analyst-programmer
- How to construct your own Python programs? These are your options:
  - Write it out in “pseudo code” (or as a flowchart). Then convert your pseudo code to Python
  - Just get started! You do not need to have a complete program to test what you have... Start simple and then add to it.
- Or do I need a program at all? The time trade-off: hard work is almost all up-front when designing/writing programs.



(Thanks, Brian!)

Final thoughts 2

- 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)
- Where to find help on Python specifics: http://www.python.org/doc/ or just Google it!
- PyLint

Final thoughts 3

- More Python
  - Finding more modules: Python Package Index http://pypi.python.org/pypi
    - Try http://docs.enthought.com/mayavi/mayavi/ for 3D graphing in Python
  - There are many, many online Python courses. Also IOS and Android apps (not all of them useful, however).

Final thoughts 4

- There are some very good (free!) online Python books e.g.
  - *How To Automate The Boring Stuff With Python* by Al Sweigart
  - Several by Allen B. Downey at http://greenteapress.com
- Python Geospatial Development book, by Erik Westra
- Python blogs are often very informative e.g.
  - "Pythonic Perambulations" at http://jakevdp.github.com/
  - "The Glowing Python" at http://glowingpython.blogspot.fr/

Final thoughts 5

- My “after sales service”: david.favis-mortlock@ouce.ox.ac.uk
- Good luck!