Introduction to Python

Jeffrey R. de Wet

August, 2015

This is an outline of what we will be covering in class on the Python programming language. We will not be looking at a slide deck. Instead, each student will run an iPython notebook which will allow them to try the code snippets for themselves, experiment and change values as they like, and end up with a record of what they did that can be revisited and rerun at any time using the iPython notebook system. There are also repositories of iPython notebooks on the internet that can be downloaded and used as a guide to certain topics or experimented with.

A. Python is a very high level object oriented interpreted programming language
    a. Code is not compiled ahead of time
    b. Read and execute by the interpreter at run time
    c. Python takes care of memory management for you
    d. python.org and documentation
    e. Start an ipython notebook
        i. What you can do with an iPython notebook
    f. What's an object?
        i. Everything you work with in Python is an object
        ii. It allows a close association between data and methods (functions) that operate on that data in some function
        iii. Example with a str (string) object
            1. dir function, console documentation
            2. help function, console documentation
            3. Accessing methods with the . (dot) operator
            4. Two important methods: strip() and split()
    g. Loading a module
        i. Python standard modules
        ii. Large number of specialized data analysis modules added by anaconda
        iii. Example of standard math module
            1. import math
            2. math functions are now in the math namespace
            3. math.sin(), math.cos(), etc.
            4. namespaces reduce the possibility of name conflicts
        iv. Accessing module functions with the . (dot) operator
    h. Importing some Python 3 behavior from __future__
        i. Printing and integer division

1. from __future__ import division, print_function

B. Python built-in data and structure objects
    a. Python variables are references – names for things (objects) we want to be able to use, but a reference has no type or properties, it is just a name
        i. Reference name must start with an alphabetic character or _, but may contain numerical digits.
        ii. Choose meaningful or descriptive names for you references
    b. = is the assignment operator
        i. Try some simple examples
    c. Two types of built-in objects: immutable and mutable
        i. Immutable
            1. int - integer, arbitrarily long
            2. float – double precision, 64 bit floats
                a. standard math operators
                    i. +, -, *, /, ** (power)
                b. In-place math operators
                    i. +=, -=, *=, /=,
            3. str – character strings, iterable (see for loop), can be sliced
                a. May use ‘ or “, must be paired
                b. “””Triple quoted string
                May extend over multiple lines”””
                c. Triple quoted string can also be used to add help strings to your code that are accessible to the Python console help system
            4. bool – True or False
                a. logical operators
                b. comparison operators
            5. None – special null object
            6. tuple – similar to a list in that it is an ordered collection of objects; once made, its contents can’t be changed; iterable (see for loop), can be sliced
            7. Conversion of types
        ii. Mutable – think of them as containers that can hold other objects
            1. list – Contents accessed by index (position) , iterable (see for loop), can be sliced, first position is 0
                a. range() – make a list of integers
            2. set – Unordered collection of unique items that can perform set operations on the contents, iterable (see for loop)

3. dict – Dictionary, a mapping type that stores key : value pairs
    a. Very fast look-ups
    b. Iterable (see for loop)
4. len() function – how big is my collection?

iii. More data structures are available in the collections module

C. Loops, Logic and Branching – program flow control
    a. Leading white space is used to indicate blocks of code that are controlled by a loop or if-elif-else construct
        i. Spaces or tabs may be used, but choose one and stick with it – use a width that is comfortable – I recommend the equivalent of 4 spaces
    b. for loop – works with iterables
        i. for var_name in iterable_object:
                controlled code block
        ii. xrange – like range except it never makes a list of integers, generates the next integer when needed
    c. while loop – operates while a test condition is True
        i. while test_is_true:
                controlled code block
    d. if elif else – first true test is the one that happens
        i. if test_is_true:
                controlled code block
            elif next_test_is_true:
                controlled code block
            else:
                if none of above are true, do this
    e. logical operators
        i. Boolean operators: and, or, not
        ii. Comparison operators: ==, >, <, >=, <=, !=
            1. == means test for same value, = means assign
        iii. in – operator that tests for membership
        iv. empty structures evaluate to False
        v. None evaluates to False
        vi. 0 evaluates to False

D. Opening and reading files
    a. The file object is iterable – when placed in a for loop, it gives one line each pass through the loop
    b. Can also read the entire file as one string
    c. Can get a list of all of the lines in a file

d. Can move around in a file by byte locations

E. Functions – modularity and grouping of related segments of code
   a. Increase readability
   b. Reusability of code
   c. Easier debugging
   d. Rule of thumb – all of the code in a function should be viewable without scrolling
   e. Note: a collection of functions can be saved in a file and imported into a program just as you import a standard, built-in module
   f. A function to convert sequence information to fasta format

Resources

https://www.python.org/ Python home, source and documentation

http://www.continuum.io/ Home of the Anaconda distribution of Python and several innovative data analysis and visualization packages

http://scipy.org/ Home of SciPy and Numpy (N-dimensional arrays with vectorized operations), advanced Python scientific computing software and resources

http://pandas.pydata.org/ Home of Pandas, a Python data analysis library

http://stanford.edu/~mwaskom/software/seaborn/ Seaborn – Statistical Data Visualization Software

http://matplotlib.org/ matplotlib – Python plotting package, used by Seaborn

http://ipython.org/ iPython

https://github.com/ipython/ipython/wiki/A-gallery-of-interesting-IPython-Notebooks iPython Notebook Gallery

Books

http://greenteapress.com/wp/ Free books about Python or using Python by Allen Downey

http://learnpythonthehardway.org/book/ Learn Python the Hard Way, online, html version

http://www.swaroopch.com/notes/python/ A Byte of Python, by Swaroop CH, Python 2 version

http://rgruet.free.fr/ Python Quick Reference

Tutorials

https://docs.python.org/2/tutorial/ The official Python tutorial

https://www.codementor.io/python/tutorial/ Python tutorials at Codementor

https://openhatch.org/wiki/Boston_Python_Workshop/archive Boston Python workshop tutorials

http://www.python-course.eu/course.php An online Python tutorial

http://pyvideo.org/ List of many videos primarily from Pycon. Some are tutorials

https://www.youtube.com/user/PyDataTV Videos from more recent PyData conferences

Also, search for Scipy videos

Tools

http://www.continuum.io/ Anaconda Python distribution, includes a large set of data analysis, scientific and plotting libraries plus easy access to many additional Python packages

http://komodoide.com/download/#edit Komodo edit – powerful multi-language code editor

https://www.jetbrains.com/pycharm/ Pycharm – a Python IDE (interactive development environment, has a free, community edition

Exporting Notebooks as pdfs

I have only done this on windows so far, although the setup is probably similar on Macs. If you would like to be able to export iPython notebooks as pdf documents you will need to install the following in the given order and install for all users:

MiKTEX http://miktex.org/
Pandoc http://pandoc.org/

The first time you try to export as a pdf, you may get a blank window in the browser and it will say waiting for server. Look for a dialog boxes behind the browser that are asking for permission to install missing packages. I got 5 or 6 of those. It may all get done automatically if you didn't request notification of missing packages during the original installation process.