python-for-researchers Documentation

Release 0.1

Eric A. McDonald

May 08, 2013

CONTENTS

Outlines and instructional materials for the *Python I: Intro* and *Python II: Data Analysis and Visualization* seminars taught as part of Michigan State University's semiannual Faculty Research Seminars.

Thank you to everyone who attended these seminars. A number of good questions were asked and your feedback at the ends of the sessions was helpful.

CHAPTER

ONE

LICENSE

CHAPTER

TWO

TABLE OF CONTENTS

2.1 Python I: Intro

2.1.1 Expectations

- Previous programming experience recommended. Focus is on using Python as a tool, not on teaching programming.
- Will learn language while learning [hopefully] useful tools.
- Python is not the solution to all your problems. (I.e., this seminar is informational rather than evangelical in nature. Draw your own conclusions.)
- Python is not a magical gift box; it is a programming language. At least some work will be required to reach your goals.

2.1.2 Python Interest Group at MSU?

- People would meet to help each other install Python and various Python packages.
- People would talk about how they use Python at work.
- People would share their experiences with various pieces of Python software.
- Any interest?

2.1.3 Why Python?

- Popularity. You will likely encounter it. Maybe you already did and maybe that is why you are here?
- Compared to many other scripting languages, it has a fairly simple syntax which encourages the writing of readable code and may be easier to learn.
- Compared to other scripting languages, it has one of the most full-featured sets of tools for scientific computing: NumPy, SciPy, pandas, and matplotlib.
- Comparisons and Contrasts
  - Matlab/Octave
    - vs NumPy, SciPy, matplotlib, and scikits
  - R
    - Can embed in IPython

        - vs pandas, StatsModels, and matplotlib
    - Julia
        - Can embed in IPython
    - Perl, Ruby, Scala, Clojure, Haskell, OCaml, etc...
    - Javascript
- Consistency
    - Warts in Ruby and Javascript: https://www.destroyallsoftware.com/talks/wat

2.1.4 Interpreters

- Python
    - Python 2 and Python 3
    - CPython, IronPython, Jython, and Stackless Python
- IPython
    - IPython Terminal
    - IPython QtConsole
    - IPython Notebook
- Interpreters on the Web
    - repl.it: http://repl.it/languages/Python
    - PythonAnywhere: https://www.pythonanywhere.com/
- Running Programs
    - Pitfall Warning: Explicit `print` vs implicit print.
- Compiling Programs
    - No explicit compilation. Performed on-the-fly from source into Python VM bytecode. (Note presence of `.pyc` and `.pyo` files.)
    - PyPy and Nuitka
    - Pyrex and Cython
    - Numba

2.1.5 Everything is an Object

- Almost everything is an object.
    - Don't worry too much about what an object is. Consider it to be a some kind of value which has some associated attributes.
    - Attributes, themselves, are generally objects.
    - Objects are created from types. Types themselves are objects.
    - Don't worry about object-oriented programming, if you're not familiar with it. Existing types are flexible enough that you usually won't have to create your own. (But, it is easy to do so if you have the need!)
- Information about an object and its attributes can be found.

  - `help()`

2.1.6 Built-in Types

- `object`
- `type`
- `module`
- `NoneType`
- `bool`
- `function` (named and anonymous)
- `int`, `long`
- `float`
- `complex`
- `str`, `unicode`, `bytearray`
- `tuple`
- `list`
- `set`
- `dict`

2.1.7 Variables

- Important: Types are associated with values rather than variable names.
- Variable names are references to values.
  - References to values are created by assignment with = statement.
  - References are likewise changed with = statement.
  - References are deleted with `del` statement.
  - Examples.
- Pitfall Warning: Multiple named references to same sequence or mapping.
  - Example.
  - How to make a copy of a sequence? Several ways - more on that later.
- Multiple assignment can be performed using commas as separators.
- Multiple values can be swapped without explicit intermediate variables.
  - Exercise: Try it!

2.1.8 Operators

- `+`, `-`, `*`, `/`, `//`, `%`, `**`
  - Exercises:
    - What happens if you add strings?
    - What happens if you multiply a string, tuple, or list by an integer?
  - Notes on integer division vs true division.
  - Examples of string interpolation.
  - Examples of the assigning variants of these operators.
- `==`, `!=`, `<`, `<=`, `>=`, `>`
- `is`, `is not`, `in`, `not in`
  - Notes on precedence and alternative keyword orders.
- `not`, `and`, `or`
  - Notes on "zeroish" vs "non-zeroish" values.
  - Notes on short-circuiting evaluation.
- `~`, `&`, `|`, `^`, `<<`, `>>`
  - Examples of the assigning variants of these operators.

2.1.9 Working with Objects

- Objects are instances of types.
  - Instances can be created by calling types or factory functions.
  - Examples.
- `dir()`
- `hasattr()`, `getattr()`, `setattr()`
- Dot notation (`.`) is used to access attributes.
  - Exercise: Try to add an attribute to an instance of `object`.
- The `class` statement defines a new type.
  - Inheritance. Old-style and new-style classes.
  - Example of simple class.
  - Exercise: Define a new class. Create an instance of it. Then, try to add a custom attribute to it. If successful, then try accessing that attribute.
- Note on special methods with double underscores.

2.1.10 Working with Strings

- Various kinds of string literals.
- `len()`
- Indexing

  - Note on zero-based indexing.
  - Exercise: What happens if you use a negative index?
- Slicing
  - Colon notation (`:`) for range and stride.
  - Examples.
- `str.strip()`
- `str.lower()` and `str.upper()`
- `str.split()` and `str.join()`
- `str.replace()`
- `str.format()`
  - Examples.
- `str.__sizeof__()`
  - Notes on character width.

2.1.11 Working with Tuples

- Creation of tuples.
- Length, indexing, and slicing like strings.
- Pitfall Warning: Syntactic sugar for 1-element tuple.
- Note on multiple assignment and tuples.

2.1.12 Working with Lists

- Creation of lists.
  - List comprehensions.
  - `range()` and `xrange()`
- Length, indexing, and slicing like strings.
- `list.append()` and `list.insert()`
  - Exercise: Insert an item at the front of a list.
- `list.extend()`
- Item removal.
  - Use `del` with an index or slice.
  - `list.pop()`
  - `list.remove()`
- `list.count()`
- `list.reverse()` and `reversed()`
- `list.sort()` and `sorted()`

2.1.13 Working with Sets

- Creation of sets.
  - Pitfall Warning: The empty set is not `{ }`!
- Length, but no indexing or slicing.
- `set.add()`
- `set.pop()`, `set.remove()`, `set.discard()`
- `set.intersection()`, `set.union()`
  - Updating variants of these methods.
  - Examples.
- `set.difference()`, `set.symmetric_difference()`
  - Updating variants of these methods.
  - Examples.
- Exercise: What do the `-`, `&`, `|`, and `^` operators do with sets?
- Exercise: What about the assigning variants of the same?
- `frozenset`

2.1.14 Working with Dictionaries

- Creation of dictionaries.
  - From a list of key-value pairs.

    `enumerate()`

    `zip()`
  - `dict.fromkeys()`
  - Dictionary comprehensions.
  - Examples.
  - Exercise: Create a dictionary, using a list of letters as keys and a list of numbers as values.
- Indexing by key, but no slicing.
- Value retrieval by indexing vs `dict.get()`.
- Manipulating sequences as dictionary values: `dict.setdefault()`
- Testing for a key with the `in` operator.
- Lists of keys, values, and key-value pairs.
  - Views vs iterators.
- `frozendict`

2.1.15 Flow Control and Modularity

- `pass`
- `def` - `yield` - `return`
  - Docstrings.
  - Functions can return multiple values.
  - Arbitrary numbers of arguments.
  - Keyword arguments.
  - Examples.
- `if` - `elif` - `else`
  - Examples.
- `for` .. `in` - `continue` - `break` - `else`
  - Really? An `else` clause with a loop? Yes.
  - Examples.
- `while` - `continue` - `break` - `else`
  - Examples.
- `try` - `except` - `else` - `finally`
  - The exception hierarchy.
  - Examples.
- `with`
  - Examples later.

2.1.16 Functional Programming

- `lambda`
- Biconditional expressions.
- `all()` and `any()`
- `map()`
- `filter()`
- `reduce()`
- `sum()`, `min()`, `max()`

2.1.17 Working with Files

- `open()` and `with` context handler
  - Modes
- Iteration over lines of text file.
- `file.read()`, `file.readline()`

- `file.write()`, `file.writeline()`

2.1.18 Miscellany

- `int()`
  - Number bases.
- `float()`, `complex()`
  - Infinities and not-a-number (NaN).
- `str()`, `unicode()`, `repr()`
- `raw_input()`
- `chr()`, `unichr()`, `ord()`
- `eval()`
- `exec()`, `execfile()`
- `__builtins__`, `__builtin__`, `builtins`
- Decorators
- Properties
- Generator Expressions

2.1.19 Standard Library

Namespaces, Scopes, and Modules

- `vars()`
- `locals()` and `globals()`
- `import`
- `from .. import ..`
- Aliasing with `as`.
- Multiple selective imports.

Back to the Future

- `__future__`
  - `division`
  - `print_function`
  - `absolute_import`

Python Sundries

- `sys`
  - `stdin`, `stdout`, `stderr`
  - `version`, `version_info`
  - `modules`
- `collections`
  - `collections.namedtuple()` (type definitions for the lazy)
  - `collections.defaultdict()`
  - `collections.OrderedDict()`

Human-Readable Data

- `pprint`

Math and Statistics

- `math`, `cmath`
  - `pi` and `e`
  - Exercise: Quadruple the arc tangent of 1.
  - Exercise: Investigate the difference between the built-in functions `int()` and `round()` and the `math` functions `math.floor()`, `math.ceil()`, and `math.trunc()`.
- `decimal`
  - Examples.
- `fractions`
  - Examples (including classic 1.1 from float and from Decimal)
- `random`
  - `random.choice()`, `random.sample()`
  - `random.shuffle()`
  - Samples from assorted distributions.
  - Examples.
  - Exercise: Generate a list of 10 random samples from a Gaussian distribution.

Gathering Data

- `csv`
  - Can handle other separators besides commas.
  - Can ignore header lines.
  - Examples.
- `urllib`, `urllib2`

  - FTP and HTTP retrieval of data.
  - Can scrape web pages for data. Use in conjunction with something like BeautifulSoup. For example, see this Stack Overflow question.
  - Examples.

Data Persistence

- `pickle`
  - `pickle.dump()`
  - `pickle.load()`
  - Exercise: Create a dictionary and set. Dump them to a file. Load them from the file.

Raking Data

- `operator`
  - `operator.itemgetter()`
  - `operator.attrgetter()`
  - Functional forms of built-in operators.
  - Examples.
- `re`
  - `re.compile()`
  - `re.findall()`
  - Examples.

Files, Directories, and Subprocesses

- `os`, `subprocess`
  - `os.getcwd()`
  - `os.getpid()`
  - `subprocess.Popen`
- `os.path`, `glob`, `shutil`

2.2 Python II: Data Analysis and Visualization

2.2.1 Expectations

- Previous Python experience *required*. Focus is on using Python as a tool, not on learning the language.
- Python is not the solution to all your problems. (I.e., this seminar is informational rather than evangelical in nature. Draw your own conclusions.)

- Python is not a magical gift box; it is a programming language. At least some work will be required to reach your goals.
- Consider me to be a tour guide rather than an expert.
  - Will highlight capabilities of various packages, but have very little experience with most of them.

2.2.2 Useful Resources

- Stack Overflow: http://stackoverflow.com (Someone has probably already asked your question and received an answer or a dozen answers. If not, you can ask questions here too.)
- Python 2 Reference: http://docs.python.org/2
- Python 3 Reference: http://docs.python.org/3
- IPython Notebook Gallery: http://nbviewer.ipython.org/
- IPython Reference: a collection of IPython tutorials

2.2.3 IPython

- Copy and paste of example output.
- Pop-up help.
- Tab completions.
- Automatic pretty-printing.
- Persistent history.
- Logging sessions.
- Demo mode.
- Loading Python files.
- Pylab
  - vs SAGE

2.2.4 NumPy

- NumPy arrays vs Python lists.
- Creation of arrays.
  - `array`
  - `arange`
  - `linspace`
  - `zeros`, `ones`
- Reshaping arrays.
- `eye`
- Element-wise operations.
- Simple linear algebra.

- Simple stats.

Note: If you were in the morning session, then you want to `np.resize` to resize a NumPy array and not `np.reshape` as I accidentally wrote. Sorry for the problem.

2.2.5 SciPy

2.2.6 matplotlib

2.2.7 pandas

2.2.8 StatsModels

2.2.9 NetworkX

2.2.10 NLTK

2.2.11 scikits

- scikit-learn
- scikit-image

2.2.12 SymPy

- Running `isympy`.
- Using SymPy from within an IPython GUI.
  - Example.

2.2.13 StarCluster

2.2.14 Miscellany

- mpi4py
- IPython parallelism
- Cython
- Numba
- PyCUDA

2.3 Downloads for Presentation

`Intro Demo`

`Standard Library Demo`

`NumPy Demo`

`Pandas Demo`

`matplotlib Demo` `matplotlib Example Script`

To use in IPython:

```
from IPython.lib.demo import Demo
demo = Demo( "intro-demo.py" )  # for example
demo( )
```

Presenter: Eric A. McDonald < em [at] msu [dot] edu >

2.4 Python Distributions

A Python distribution is a software bundle, which contains a Python interpreter and the Python standard library. Installer programs for common operating systems are a frequent mode of distribution. Many Python distributions also have package managers so that you can install or upgrade various Python packages.

Some of the most popular distributions are listed below. Distributions which are marked as "scientific" are ones which come with *IPython*, *numpy*, *pandas*, and *matplotlib*, at a minimum. All of the distributions provide at least one integrated development environment (IDE) for free. A Python IDE provides a Python-aware code editor integrated with the ability to run code from that editor.

2.4.1 ActiveState ActivePython

Scientific: No, but many scientific packages can be added via the package manager

Platform: AIX, HP-UX, Linux, MacOS X, Solaris, Windows

Overview: http://www.activestate.com/activepython

Downloads: http://www.activestate.com/compare-editions

Package List: too many to list - use search page: http://code.activestate.com/pypm/search:/?tab=name

Package Manager: PyPM

IDE: *IDLE*, *Komodo* (must be purchased separately from ActiveState)

Note: ActivePython is one of the oldest Python distributions, but is not particularly geared towards science.

2.4.2 Continuum Analytics Anaconda

Scientific: Yes

Platform: Linux, MacOS X, Windows

Overview: https://store.continuum.io/

Downloads: [Note: Complete distribution is available for free, but requires registration first. Also, academics can get several powerful commercial add-on products for free with proof of affiliation with an educational institution.]

Package List: http://docs.continuum.io/anaconda/pkgs.html

Package Manager: conda

IDE: *Spyder*

Note: Continuum provides a Python compiler, called Numba, as part of its distribution. This can compile Python code down to machine code and is aware of how to optimize with special consideration for the popular *numpy*.

Note: Commercial add-on tools are linked against the Intel Math Kernel Library (MKL) for improved numerical performance.

2.4.3 Enthought Canopy

Scientific: Yes

Platform: Linux, MacOS X, Windows

Overview: https://www.enthought.com/products/canopy/

Downloads: [Note: Academics can get the professional version for free by registering for an academic license at https://www.enthought.com/products/canopy/academic/ .]

Package List: https://www.enthought.com/products/canopy/package-index/

Package Manager: Canopy Package Manager

IDE: *IDLE*, *SciTE*

Note: Professional version is linked against the Intel Math Kernel Library (MKL) for improved numerical performance.

2.4.4 Python(x,y)

Scientific: Yes

Platform: Windows

Overview: https://code.google.com/p/pythonxy/

Downloads: https://code.google.com/p/pythonxy/wiki/Downloads

Package List: https://code.google.com/p/pythonxy/wiki/StandardPlugins

IDE: *SciTE*, *Spyder*

Other Tools: Console (enhanced Windows command line window), WinMerge (differencing and merging of files on Windows)

Note: A variant of this distribution is also available for Linux; please see pythonxy-linux.

2.4.5 WinPython

Scientific: Yes

Platform: Windows

Overview: https://code.google.com/p/winpython/

Downloads: https://code.google.com/p/winpython/downloads/list

Package List: https://code.google.com/p/winpython/wiki/PackageIndex

Package Manager: WinPython Package Manager (WPPM)

IDE: *SciTE*, *Spyder*

Other Tools: TortoiseHG (Mercurial version control system integrated into Windows Explorer)

2.5 Additional Python Packages

Some Python packages may not be a part of some distributions, but contain files which must be compiled (i.e., they are not "pure Python"). As it can be difficult to compile these files, especially on Windows, there exist third-party repositories of precompiled packages.

2.5.1 Christoph Gohlke's Windows Binaries

Overview: http://www.lfd.uci.edu/~gohlke/pythonlibs/

Downloads: http://www.lfd.uci.edu/~gohlke/pythonlibs/

Package List: http://www.lfd.uci.edu/~gohlke/pythonlibs/

Platform: Windows

2.6 Python Packages

2.6.1 IPython

Overview: http://ipython.org/

Documentation: http://ipython.org/documentation.html

Examples Gallery: http://nbviewer.ipython.org/

2.6.2 NumPy

Overview: http://www.numpy.org/

Documentation: http://docs.scipy.org/doc/numpy/reference/

Tutorial: http://www.scipy.org/Tentative_NumPy_Tutorial

2.6.3 SciPy

Overview: http://www.scipy.org/

Documentation: http://docs.scipy.org/doc/

Tutorial: http://docs.scipy.org/doc/scipy/reference/tutorial/

Recipes: http://www.scipy.org/Cookbook

2.6.4 matplotlib

Overview: http://matplotlib.org/

Documentation: http://matplotlib.org/contents.html

Examples Index: http://matplotlib.org/examples/index.html

Examples Gallery: http://matplotlib.org/gallery.html

2.6.5 pandas

Overview: http://pandas.pydata.org/

Documentation: http://pandas.pydata.org/pandas-docs/stable/

Recipes: http://pandas.pydata.org/pandas-docs/stable/cookbook.html

2.6.6 StatsModels

Overview: http://statsmodels.sourceforge.net/

Documentation: http://statsmodels.sourceforge.net/stable/index.html

2.6.7 NetworkX

Overview: http://networkx.github.io/

Documentation: http://networkx.github.io/documentation/latest/contents.html

Tutorial: http://networkx.github.io/documentation/latest/tutorial/

Examples: http://networkx.github.io/documentation/latest/examples/

2.6.8 Natural Language Toolkit (NLTK)

Overview: http://nltk.org/

Documentation: http://nltk.org/api/nltk.html

Tutorial: http://nltk.org/book/

2.6.9 scikits

Overview: http://scikits.appspot.com/

Index of Kits: http://scikits.appspot.com/scikits

scikit-learn

Overview: http://scikit-learn.org/stable/

Documentation: http://scikit-learn.org/stable/modules/classes.html

Tutorial: http://scikit-learn.org/stable/tutorial/index.html

Examples Gallery: http://scikit-learn.org/stable/auto_examples/index.html

scikit-image

Overview: http://scikit-image.org/

Documentation: http://scikit-image.org/docs/dev/

Examples Gallery: http://scikit-image.org/docs/dev/auto_examples/

2.6.10 SymPy

Overview: http://sympy.org/en/index.html

Documentation: http://docs.sympy.org/

Online Interpreter: http://live.sympy.org/

2.6.11 StarCluster

Overview: http://star.mit.edu/cluster/

Documentation: http://star.mit.edu/cluster/docs/latest/manual/index.html

Tutorial: http://star.mit.edu/cluster/docs/latest/quickstart.html

Note: You need to setup an Amazon AWS account before you can do anything useful with this software. The account is free and you are given an initial number of compute hours for free as well. However, you will be charged beyond a certain point or for using a certain class of computing resources.

2.6.12 Cython

Overview: http://cython.org/

Documentation: http://docs.cython.org/

Tutorial: http://docs.cython.org/src/tutorial/index.html

2.6.13 Numba

Overview: http://numba.pydata.org/

Documentation: http://numba.pydata.org/numba-doc/0.8/index.html

Examples: http://numba.pydata.org/numba-doc/0.8/examples.html