Introduction to Python
General Introduction, Basic Data Types, Functions

Christopher Barker

UW Continuing Education / Isilon

June 20, 2012

Table of Contents

Instuctors

Christopher Barker: PythonCHB@gmail.com

Dan Rutz: danrutz@hotmail.com

Chris' History

First computer:

- Commodore Pet – 8k RAM
  - Basic

High School:

- PDP 11 – paper printer terminal 200baud modem
  - Basic

Chris' History

First computer:

- Commodore Pet – 8k RAM
  - Basic

High School:

- PDP 11 – paper printer terminal 200baud modem
  - Basic

College:

- Pascal: VAX/VMS 750
- Scheme: Unix VAX 780

Chris' History

First computer:

- Commodore Pet – 8k RAM
  - Basic

High School:

- PDP 11 – paper printer terminal 200baud modem
  - Basic

College:

- Pascal: VAX/VMS 750
- Scheme: Unix VAX 780

Then a long Break: Theater Arts Major, Scenery, Lighting...

Chris' History

Back to School: PhD Engineering

- DOS / Windows 3.1
  - FORTRAN
  - MATLAB
  - Discovered Linux (RedHat 2.0)

Chris' History

Back to School: PhD Engineering

- DOS / Windows 3.1
  - FORTRAN
  - MATLAB
  - Discovered Linux (RedHat 2.0)

Now:

- Oceanographer for NOAA
- Oil Spill Modeling
- Software Development

Chris' History

Back to School: PhD Engineering

- DOS / Windows 3.1
  - FORTRAN
  - MATLAB
  - Discovered Linux (RedHat 2.0)

Now:

- Oceanographer for NOAA
- Oil Spill Modeling
- Software Development

Gave TCL a try.....

Chris' History

Back to School: PhD Engineering

- DOS / Windows 3.1
  - FORTRAN
  - MATLAB
  - Discovered Linux (RedHat 2.0)

Now:

- Oceanographer for NOAA
- Oil Spill Modeling
- Software Development

Gave TCL a try.....

Gave Perl a try.....

Chris' History

Discovered Python in 1998

- It could do what Perl could do,
  - what TCL could do, what MATLAB could do,
- But I liked it – it fit my brain

My Python use now:

- Lots of text file crunching
- Lots of data processing scripts
- Automation of downloading, processing data
- Desktop GUIs (wxPython)
- computational code
- wrapping C/C++ code
- web apps (Pylons, Pyramid)
- GIS processing
- Ask me about "BILS"

Who are you?

A bit about you:

- name
- What do you do at Islion?
- programing background (languages)

Class Structure

github project

`https://github.com/PythonCHB/PythonIntroClass`

Syllabus:

`github.com/PythonCHB/PythonIntroClass/wiki/Syllabus`

Code, etc:

git:

`https://github.com/PythonCHB/PythonIntroClass.git`

svn:

```
svn co https://github.com/PythonCHB/PythonIntroClass
```

Lightning Talks

Lightning talks

- 5 minutes (including setup) - no kidding!
- Every student will give one
- Purposes: introduce yourself, share interests, also show Python applications
- Any topic you like, that is related to Python – according to you!

What is Python?

- Dynamic
- Object oriented
- Byte-compiled
- interpreted
- ....

Python Ecosystem

Used for:

- CS education (this course!)
- Application scripting (GIS, GNU Radio, Blender...)
- Systems administration and "glue"
- Web applications (Django etc. etc. etc.)
- Scientific/technical computing (a la MATLAB, Mathematica, also BioPython etc. ..)
- Software tools (automated software testing, distributed version control, ...)
- Research (natural language, graph theory, distributed computing, ...)

An unusually large number of niches – versatile

Python Ecosystem

Used by:

- Beginners
- Professional software developers, computer system administrators, ...
- Professionals OTHER THAN computer specialists: biologists, urban planners, ...

An unusually large number of types of users – versatile

You can be productive in Python WITHOUT full-time immersion!

Python Features

Gets many things right:

- Readable – looks nice, makes sense
- No ideology about best way to program – object-oriented programming, functional, etc.
- No platform preference – Windows, Mac, Linux, ...
- Easy to connect to other languages – C, Fortran - essential for science/math
- Large standard library
- Even larger network of external packages
- Countless conveniences, large and small, make it pleasant to work with

Python Features

Features:

- Unlike C, C++, C#, Java ... More like Ruby, Lisp, Perl, Matlab, Mathematica ...
- Dynamic - no type declarations
  - programs are shorter
  - programs are more flexible
  - less code means fewer bugs
- Interpreted - no separate compile, build steps - programming process is simpler

Python Versions

Python 2.*

"Classic" Python – evolved from original

Python 3.* ("py3k")

Updated version – removed the "warts" allowed to break code (but really not all that different). Not all that well adopted yet – many packages not supported.

This class uses Python 2.7 not Python 3

Implementations

- Jython (JVM)
- Iron Python (.NET)
- PyPy – Python written in Python (actually RPy...)

We will use CPython 2.7 from python.org for this course.

A Tiny Bit of History

Invented/developed by Guido van Rossum in 1989 – first version was written on a Mac. Time of origin similar to TCL and Perl.

Date	Version
Dec 1989	started
Feb 1991	0.9.0
Jan 1994	1.0.0
Apr 1999	1.5.2
Sept 2006	2.5
Dec 2008	3.0
Jul 2010	2.7

GvR at Google – still the BDFL

Code swarm for Python history: `http://vimeo.com/1093745`

What's a Dynamic language

Strong, Dynamic typing.
- Type checking and dispatch happen at run-time

```
X = A+B
```

What's a Dynamic language

Strong, Dynamic typing.
- Type checking and dispatch happen at run-time

```
X = A+B
```

- What is A?
- What is B?
- What does is mean to add them?

What's a Dynamic language

Strong, Dynamic typing.
- Type checking and dispatch happen at run-time

```
X = A+B
```

- What is A?
- What is B?
- What does is mean to add them?

- A and B can change at any time before this process

Using Python

All you need for Python:

- A good programmer's text editor
  - Good Python mode
  - Particularly indentation!
- The command line to run code
- The interactive shell
  - regular interpreter
  - `IPython` is an excellent enhancement
    `http://ipython.org/`

There are lots of Editors, IDES, etc.:
maybe you'll find one you like.

Running Python Code

- At an interpreter prompt:

```
$ python
>>> print 'Hello, world!'
Hello, world!
```

Lab

Getting everyone on-line and at a command line.

- Log in
- Do a git or SVN checkout of the project
- Start up the Python interpreter:
  `$ python` (`ctrl+D` to exit)
- create a file in your editor and save it
- Start up `IPython`
  `$ ipython` (also `ctrl+D` to exit)