The SysAdmin's Guide to Python

by Daniel Mikusa
Software Support Engineer @ Pivotal

2015 MACADMINS CONFERENCE AT PENN STATE

python™

About Me

- Daniel Mikusa
  - Blog / Website
  - Twitter / Google Plus
  - Github (Work) / Github (Personal)
- Long time Mac user
- Professional software developer
- Have used Python for the last decade
- Python helped me to build everything from scripts, to IVR & web apps

Agenda

- Introduction
- Installing Python
- Developing with Python
- Batteries Included: the Standard Library
- Everything Else: Third Party Libraries
- Distributing Your Code

Introduction

- Goals
  - Dive into the world of Python development
  - Show common & good practices for coding
  - Show tools useful to make life easier
  - Showcase why Python is great for SysAdmins
  - Show how to package up your code
- Out of scope
  - Introduction to Python / Python the language
  - Web Development w/Python

Installing Python

- Hey, that's easy right? Included w/the OS.
- Well...
  - What if you want the latest version?
  - What if you want a specific version?
  - What if you have two apps with different sets of libs?
  - What if you want Python 3?
- Options:
  - Use the system version
  - Install from python.org
  - Pyenv - github.com/yyuu/pyenv

Developing with Python

Coding Styles

- Good Style is Important
  - Make code better, more readable, more maintainable and it helps to squash bugs
- It's easy w/Python!
  - PEP-8 Style Guide for Python Code
  - PEP-20 Zen of Python
  - flake8 Linter & automated style check
- Integrate flake8 w/your text editor or VCS
  - Git & Mecurial
  - VIM, Sublime Text, Atom & others all support it

Text Editors & IDEs

- VIM works great (my preference)
  - Supports: snippets, syntax highlights, validation (flake8), file browser and code completion
  - python-mode a great place to start
- SublimeText work great too (I hear)
- There are some IDE's too: PyCharm, PyDev & NINJA-IDE.
- No right or wrong answer, pick what works best for you!

Virtual Environments

- Separate environments for each project
- No dependency overlap / mismatch
- Two Options:
  - virtualenv
  - virtualenvwrapper
- Short how-to guide on each
- One of the few things I install globally
- Pyenv has plugins for both
- Pick one, use it.

Testing Tools

- Python is not compiled, so it's extra important to test your code
- Standard library has support the unittest library, great place to start
- Running tests:
  - *python -m unittest <test>*
  - *nosetests* or *nosetests <file>*
  - Many other options
- Integrate with text editor, VCS or run when files change (tdaemon) to automate the process

Project Structure

- No real requirements, can be as little as a single file or script
- Suggestion:
  - project_root
    - <module_name>/
      - __init__.py, <name>.py
    - bin/
    - docs/
    - setup.py
    - tests/
      - __init__.py, <name>_test.py
    - scripts/
    - README.md

Other Odds & Ends

- Source control
- Terminal
- Interactive Python Shell
  - default is just python
  - ipython & bpython are alternatives
- Sphinx for docs

Demo: Project Setup

Development Workflow

Naive Workflow

- Edit code, run it, use it, find problems, fix
- Strengths
  - get started quickly
  - write small or simple scripts quickly
  - helpful with prototypes, throw-away code or when you’re trying to figure out an API
- Problems
  - small projects don’t always stay small
  - complexity increases time to find problems
  - using code may not thoroughly test all of it
  - regressions can happen

Test Driven Workflow

- Write tests, tests fail, write code to fix tests
- Strengths
  - fast iteration & feedback
  - test guarantee fitness of all code
  - tests informally document behavior of code
  - maintenance of code is easier
- Problems
  - need to write more code
  - slower to get started
  - some things are hard to test (file systems, networks)

Demo: Dev Workflow

Batteries Included: The Standard Library

Intro

- Standard library provide much of the capabilities of Python
- Extensive list of libraries some written in Python & some written in C
- Integrates with the OS to provide platform neutral APIs
- Nothing to install, it's there out-of-the-box
- Full Docs

Fundamental Libraries

- Provide basic functionality. Used by tons of scripts, libraries and applications.
  - re - regular expressions
  - datetime, calendar & time - time & date functionality
  - random - for non-secure randomness
  - itertools - helpers for making fast, efficient iterators (ifilter, imap, izip)
  - sys - system specific functionality, specifically access to command line args, python path & exit
  - os - operating system specific apis. Access to environment variables, user / group info and most of the file system access

File APIs

- These parts of the standard library allow you to interact with files & the file system.
  - os - provides basics like open, mkdir, stat, rmdir, remove and walk
  - os.path - everything needed for path manipulation, including join, dirname, basename & exists
  - tempfile - create temporary files & directories
  - glob - unix style pattern matching (i.e. *.gif)
  - shutil - high level file ops like copy, move, copytree and rmtree

Parsing APIs

- Allow you to easily parse info, strings & files
  - argparse, optparse & getopt - command line argument parsing libraries. argparse is preferred.
  - json - parse & writes json strings & files
  - csv - read & write csv files
  - base64 - RFC 3548 encoders
  - codecs - text encoding
  - pickle, cPickle - Python object serialization
  - there's a host of others, parsing for HTML, XML (DOM & SAX) and email

Debugging & Profiling Code

- The old standby, print and the pprint module
- Break points and stepping through code
  - pdb - the python debugger, similar to gdb
  - pudb - an enhanced visual debugger
- Profiling Code
  - timeit - measure execution time of code
  - profile / cProfile - deterministic profiles for code

Other APIs

- Compression
  - zipfile, gzip, bz2 and tar
- Crypto
  - hashlib - secure hashes and digests
  - hmac - keyed hashing for messages
- Logging - app logging & logging config
- Subprocess - spawning subprocesses
- Signal - signal handling
- Socket - low-level socket api
- urllib / urllib2 - send HTTP requests

Everything Else: Third Party Libraries

Improvements

Libraries that improve on parts of the standard library.

- requests - http for humans
- wrapt - easy & correct decorators
- pytz - timezone handling
- delorean - Enhanced date & time library
- pycrypto - Cryptographic toolkit
- sh - Easy subprocess launching
- docopt & click - Processing command line arguments

New Stuff

Libraries that add new functionality.

- paramiki - SSH / SFTP library
- PyYaml - Yaml library for Python
- matplotlib & pygal - Graph & plotting library
- reportlab - PDF generation
- Other Libraries

Demo: Pulling it Together

Distributing Your Code

Distutils

- Standard way to package up your library or scripts
- Great for installing libraries and command line scripts
- Can publish to PyPi or a private package repository
- Install
  - From source: *python setup.py install*
  - From repo: *pip install <pkg-name>*

py2app

- Can be used to create a MacOS app from your Python code
- Mostly useful when developing GUIs
- Usage is straightforward, RTFM
  - pythonhosted.org/py2app/

Demo: Distutils

Summary

- Python is a great language for Sys Admins
  - installed on many systems by default
  - tons of libraries included out-of-the-box
  - great development tools for being productive
  - easy to package code for distribution & sharing

Questions

Feedback

http://j.mp/psumac2015-101