Python-specific Packaging

setuptools, virtualenv, PyPUG (and a little bit of Conda)

Dale Visser
https://dalevisser.wordpress.com/

Outline

- PyPI, a.k.a., the Cheese Shop
  - Setuptools
  - Sdists, Eggs and Wheels
  - pip and virtualenv
  - Stats
  - Security
- Conda
- Conclusion

Image credit: http://www.clker.com/clipart-cheese-wheel.html

Setuptools (1/2)

```
from setuptools import setup, find_packages
setup(
    name = "HelloWorld",
    version = "0.1",
    packages = find_packages(),
    scripts = ['say_hello.py'],

    # Project uses reStructuredText, so ensure that the docutils get
    # installed or upgraded on the target machine
    install_requires = ['docutils>=0.3'],

    package_data = {
        # If any package contains *.txt or *.rst files, include them:
        '': ['*.txt', '*.rst'],
        # And include any *.msg files found in the 'hello' package, too:
        'hello': ['*.msg'],
    },
```

Source: https://bitbucket.org/pypa/setuptools#basic-use

Setuptools (2/2)

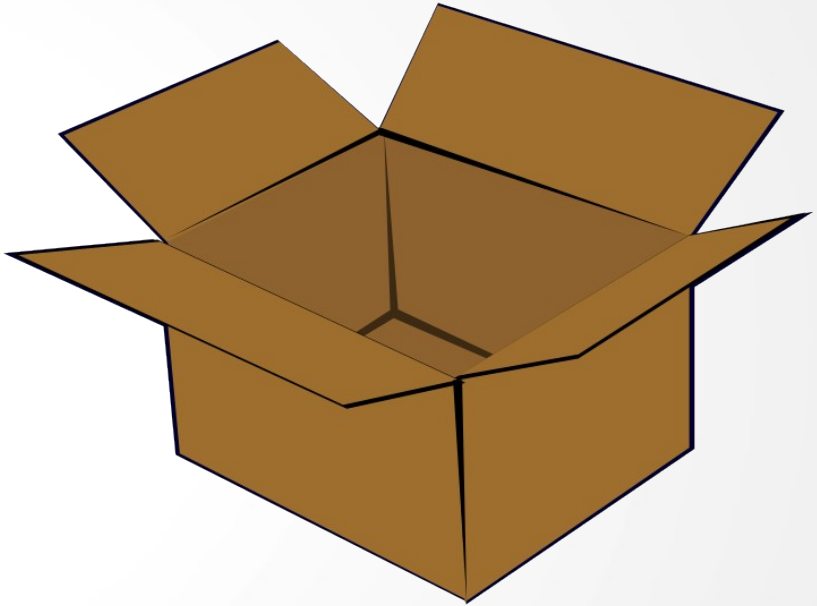
```
    # metadata for upload to PyPI
    author = "Me",
    author_email = "me@example.com",
    description = "This is an Example Package",
    license = "PSF",
    keywords = "hello world example examples",
    url = "http://example.com/HelloWorld/",   # project home page, if any

    # could also include long_description, download_url, classifiers, etc.
)
```

Source: https://bitbucket.org/pypa/setuptools#basic-use

Python Distributions – 2 kinds

- Source Distributions or “sdists”
  - python setup.py sdist
- Built Distributions (“bdists”)
  - Eggs
    - python setup.py bdist or
    - python setup.py bdist_egg
  - Wheels
    - python setup.py bdist_wheel
    - python setup.py bdist_wheel --universal



Egg and Wheel Distribution Formats

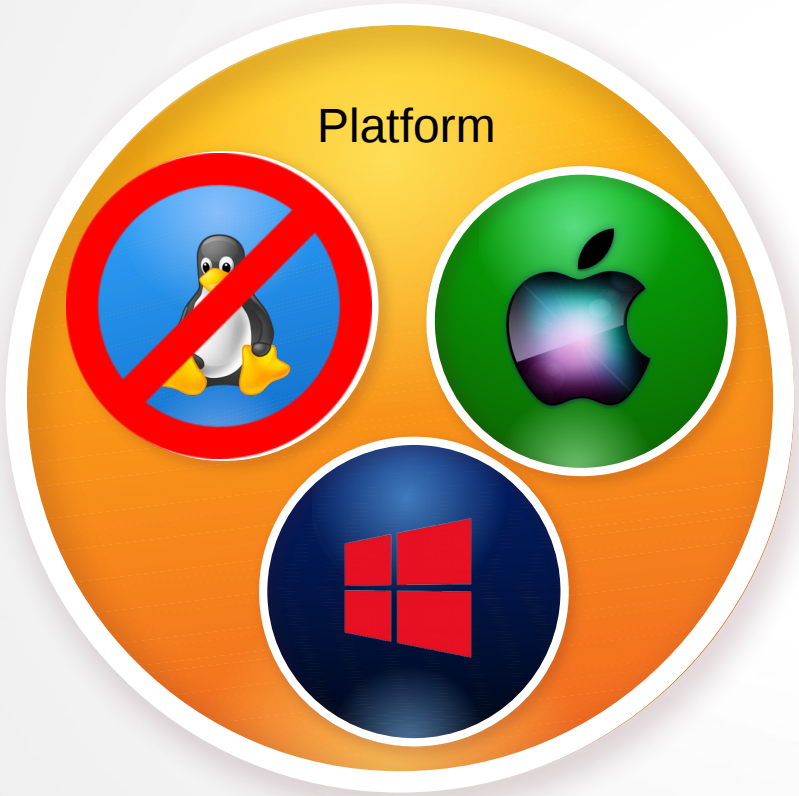
- Eggs - classic
  - .egg – zip archive or folder containing package and metadata
  - .egg-info – metadata folder that sits *alongside* installed package folder
  - Lacks *formal* specification, but *is* well-described: https://bit.ly/egg_format
- Wheels – pip and PyPI preferred
  - .whl – zip archive containing package and metadata
  - .dist-info – metadatafolder that sits *alongside* installed package folder
  - Relevant specifications:
    - PEP-376 - Database of Installed Python Distributions
    - PEP-426 - Metadata for Python Software Packages 2.0
    - PEP-427 - The Wheel Binary Package Format 1.0

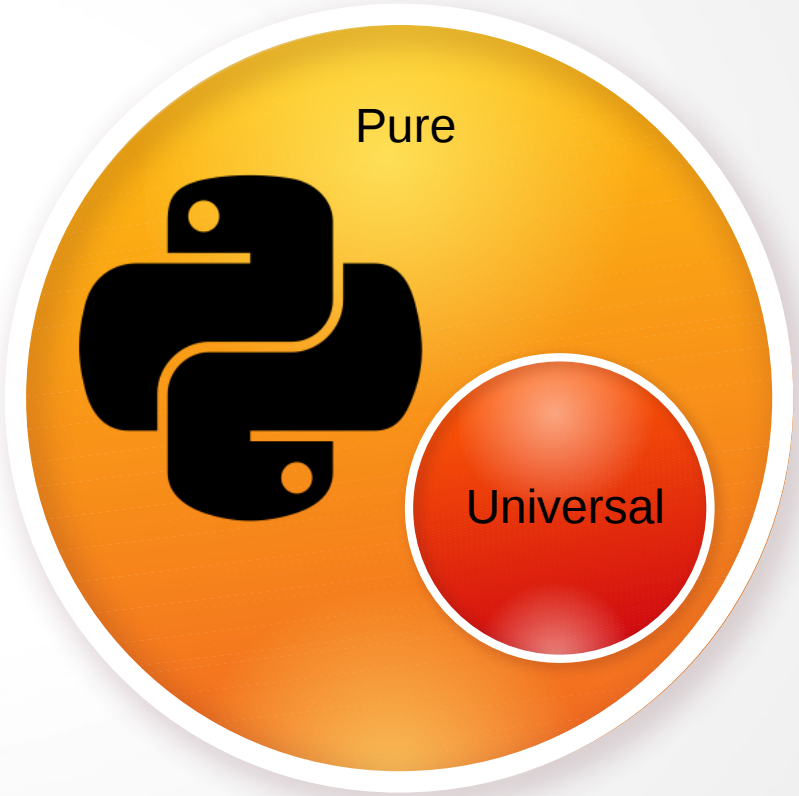
Debian/Ubuntu Conventions

- PyPI packages go into a `site-packages` folder on the local system.
- Debian (and derivatives) also distribute some python packages via APT (Advanced Package Tool).
  - From https://wiki.debian.org/Python#Deviations_from_upstream

    "Third party Python software installed from Debian packages goes into `dist-packages`, not `site-packages`."
- See `/usr/local/lib/pythonX.Y/`

Wheels





pip and virtualenv demo

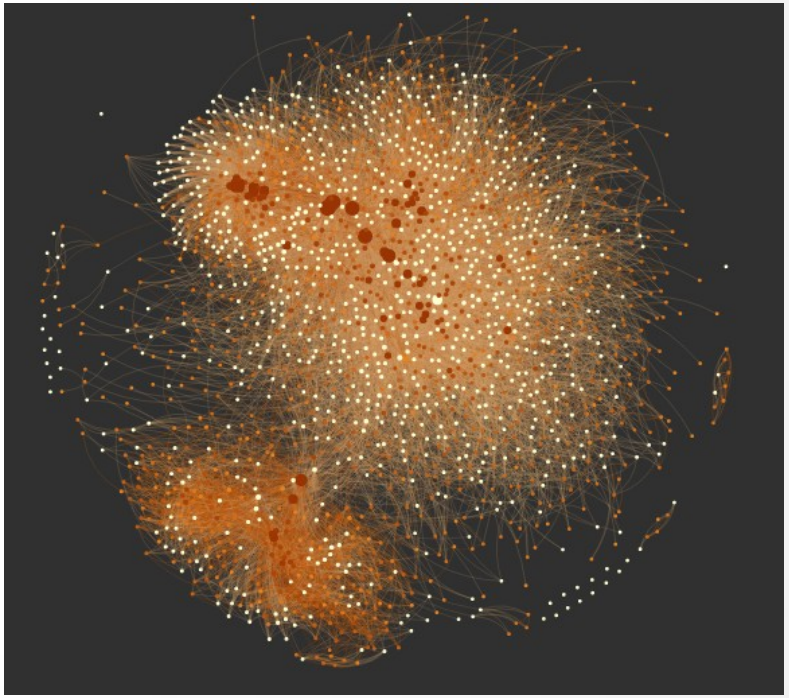
Online Resources

- Your first, best resource is this: https://packaging.python.org/
  - It is a collaborative resource created by the PyPI people. E.g., find out about *twine*
- Also, you can “pip install” from git repos!: https://bit.ly/pip_install_vcs

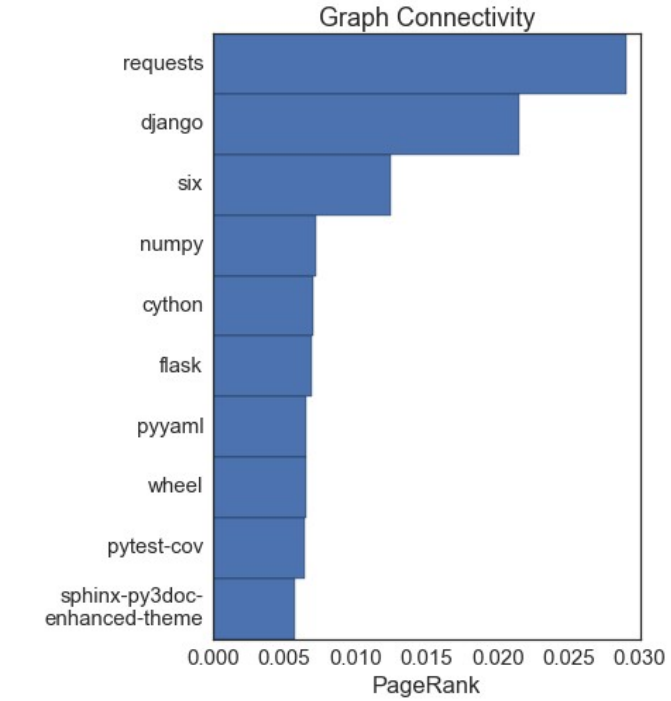
Image credit: http://www.clker.com/clipart-9829.html

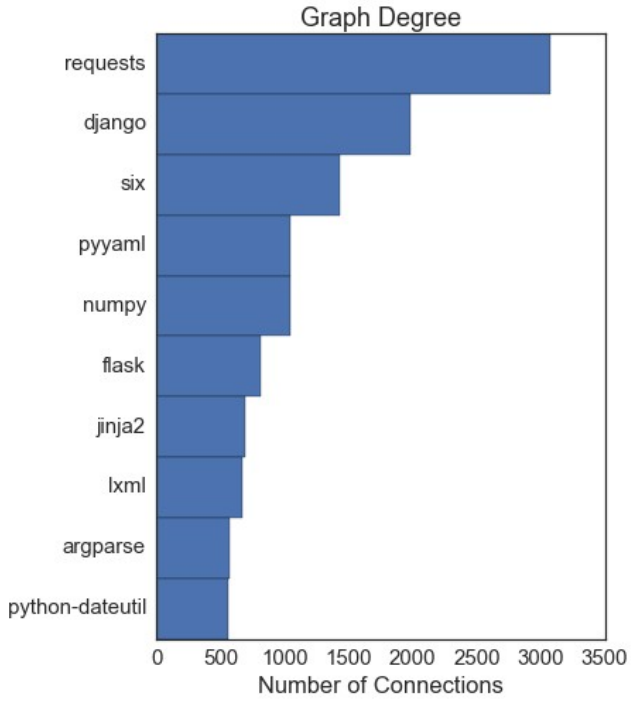
Dependency stats – overall graph

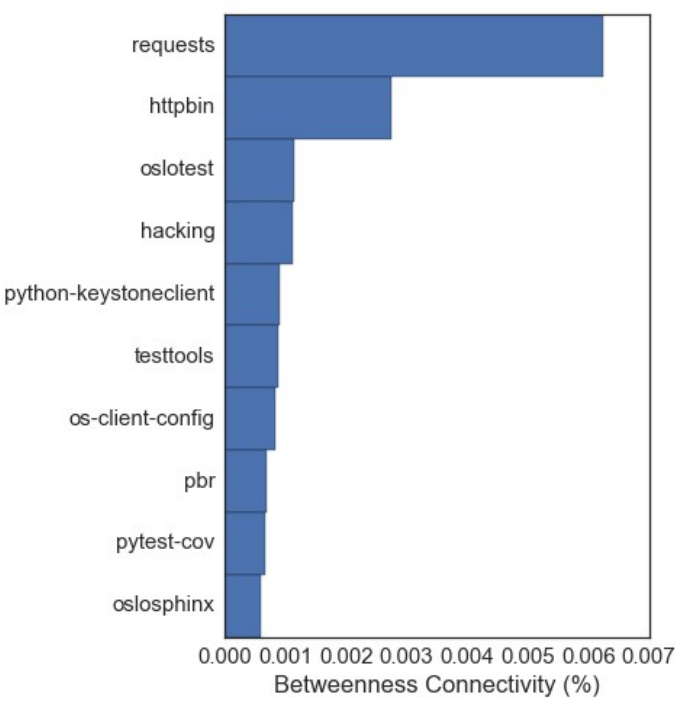
- https://kgullikson88.github.io/blog/pypi-analysis.html
- Biggest nodes
  - requests
  - zope



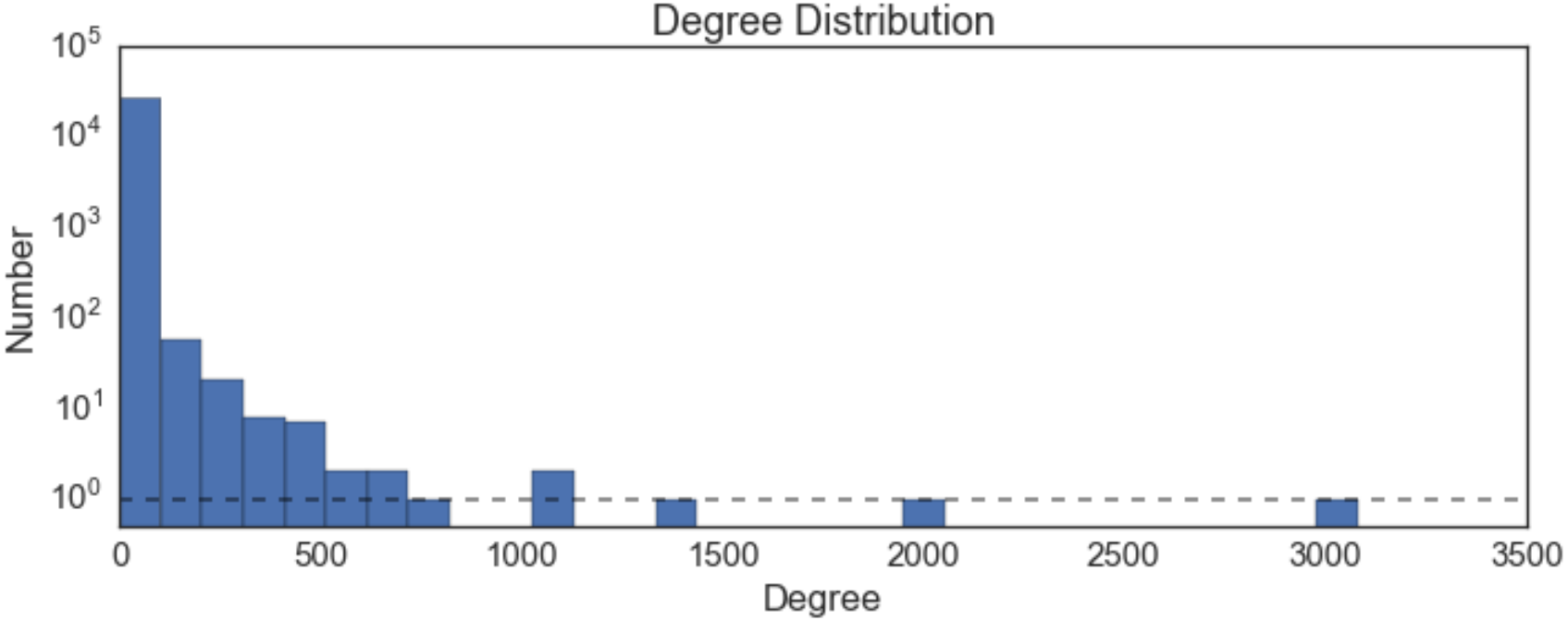
Stats: Measures of package importance





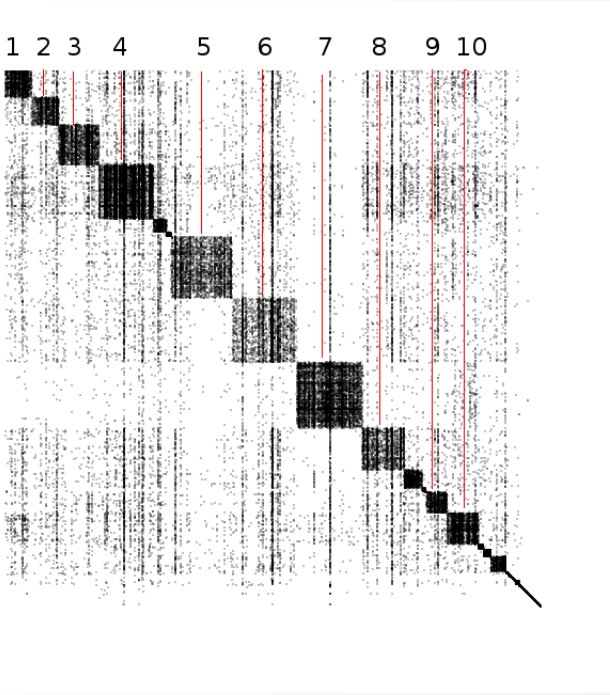


Stats - # other packages depended on



Stats (Development Communities)

1) Flask, bottle
2) Redis, tornado, pyzmq
3) Numpy, scipy, matplotlib, pandas
4) Testing/documentation packages
5) Django
6) Requests
7) Distribute (i.e., Zope)
8) Static website dev (e.g., pyyaml, jinja2)
9) Argparse, decorator, pyparsing, et al.
10) Various, most important: sqlalchemy



Keep your library dependencies secure

Have a look at OWASP Dependency Check.

- PyPI
  - Looks at: Python source files (*.py); Package metadata files (PKG-INFO, METADATA); Package Distribution Files (*.whl, *.egg, *.zip)
  - Analyses using: Regex scan of Python source files for setuptools metadata; Parse RFC822 header format for metadata in all other artifacts.
- Also scans archive files, .NET assmblies, autoconf, Maven/Nexus, Cmake, PHP composer.lock, .jar, .war, NPM package.json, Nuget *.nuspec, Ruby *.gemspec

Conda

- I've only played a little with it since the last meeting…
- Combines pip/virtualenv capabilities
- Allows to work with specific versions of Python, independent of what's available from your OS
- For packages not on anaconda.org, its environments can interoperate with “pip install”
- Perhaps someone else would like to give a talk? ☺