Computational Linguistics Seminar
Packaging

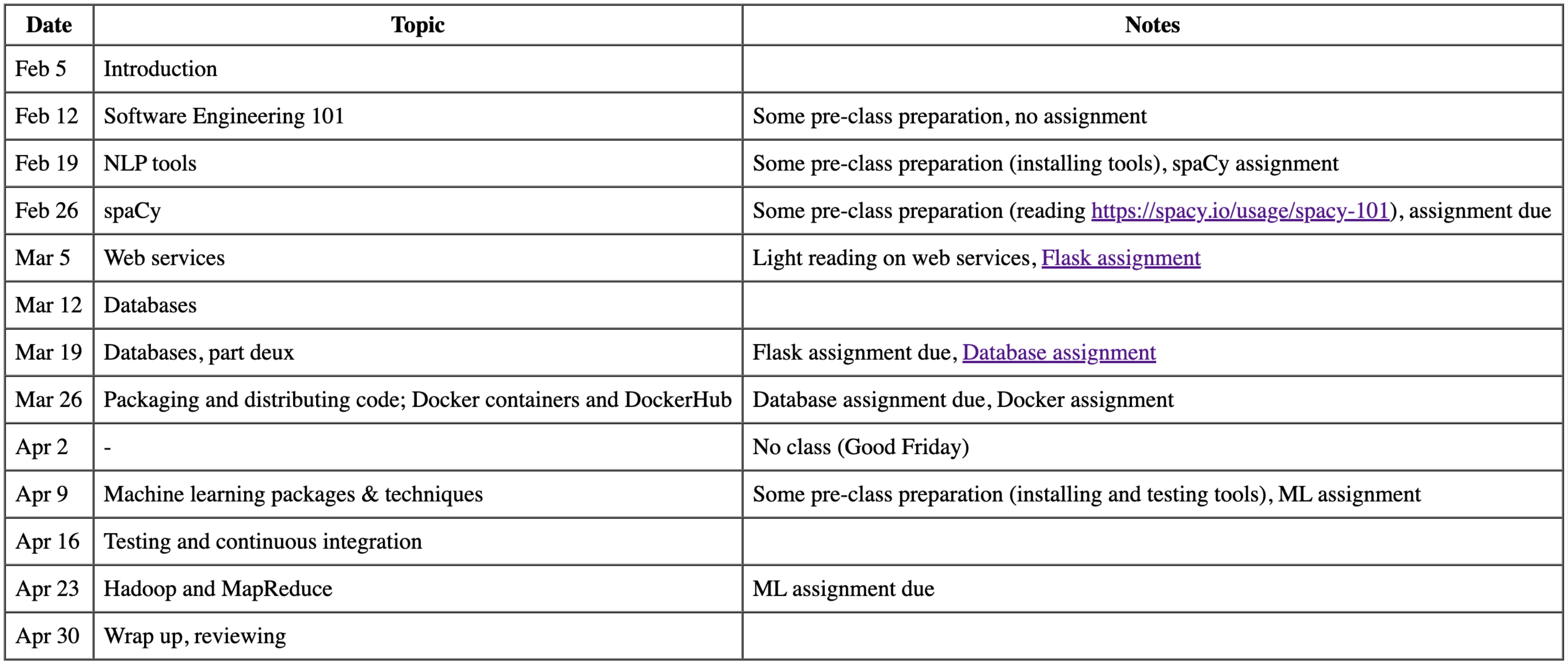
Marc Verhagen
Brandeis University
Spring 2021

The banner image is a fragment of Primordial Soup at https://regenaxe.com/2017/01/17/primordial-soup/

Overview

- Flask assignment solution
- Database assignment
- Python virtual environments
- Github releases, GitHub pages for manuals & Read the Docs
- Packaging
  - PyPI - Python Package Index
  - Trying it yourself

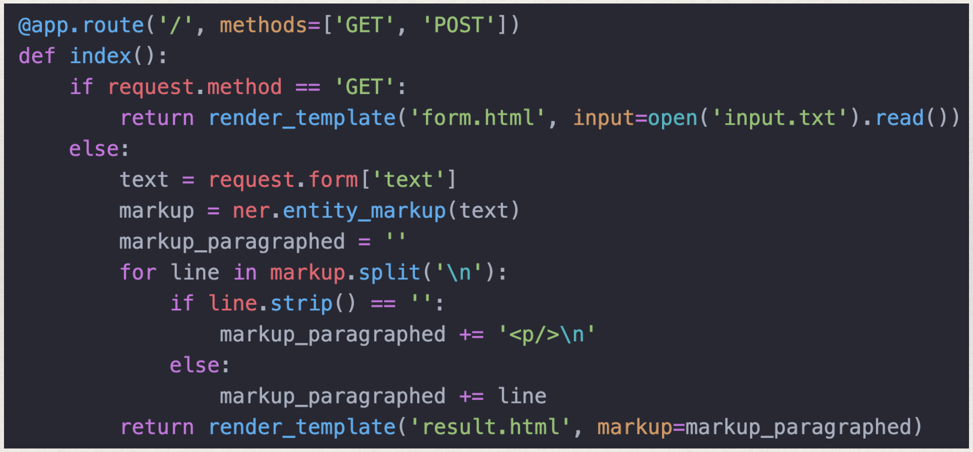
Mini-poll



Date	Topic	Notes
Feb 5	Introduction	
Feb 12	Software Engineering 101	Some pre-class preparation, no assignment
Feb 19	NLP tools	Some pre-class preparation (installing tools), spaCy assignment
Feb 26	spaCy	Some pre-class preparation (reading https://spacy.io/usage/spacy-101), assignment due
Mar 5	Web services	Light reading on web services, Flask assignment
Mar 12	Databases	
Mar 19	Databases, part deux	Flask assignment due, Database assignment
Mar 26	Packaging and distributing code; Docker containers and DockerHub	Database assignment due, Docker assignment
Apr 2	-	No class (Good Friday)
Apr 9	Machine learning packages & techniques	Some pre-class preparation (installing and testing tools), ML assignment
Apr 16	Testing and continuous integration	
Apr 23	Hadoop and MapReduce	ML assignment due
Apr 30	Wrap up, reviewing	

Flask Assignment

```
.
├── app.py
├── input.txt
├── ner.py
├── static
│   └── css
│       └── main.css
└── templates
    ├── form.html
    └── result.html
```



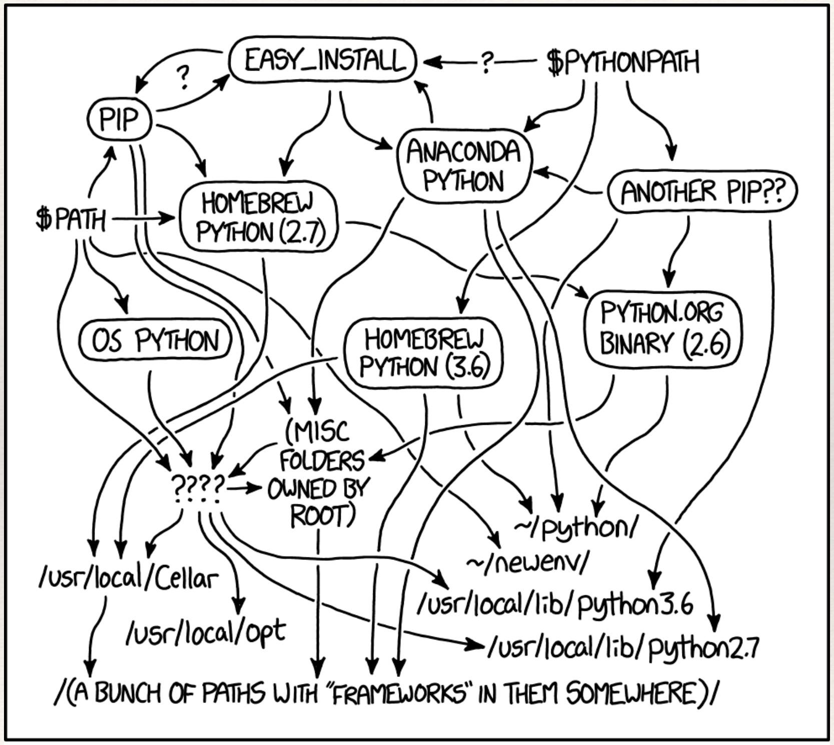
```python
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'GET':
        return render_template('form.html', input=open('input.txt').read())
    else:
        text = request.form['text']
        markup = ner.entity_markup(text)
        markup_paragraphed = ''
        for line in markup.split('\n'):
            if line.strip() == '':
                markup_paragraphed += '<p/>\n'
            else:
                markup_paragraphed += line
        return render_template('result.html', markup=markup_paragraphed)
```

Schedule individual 15-minute walkthroughs

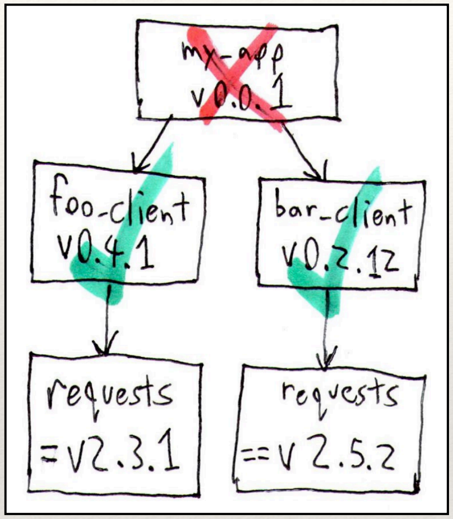
Database assignment

- Questions?

Python installations and modules

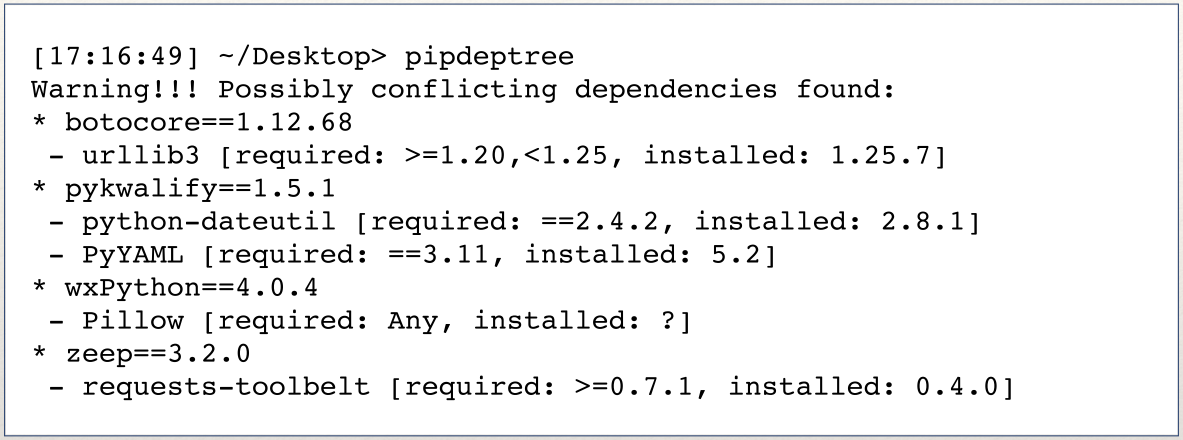


MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.



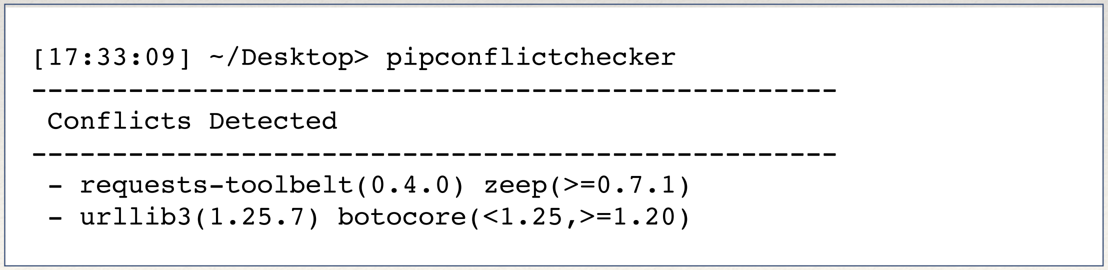
https://medium.com/knerd/the-nine-circles-of-python-dependency-hell-481d53e3e025

Python installations and modules



```
[17:16:49] ~/Desktop> pipdeptree
Warning!!! Possibly conflicting dependencies found:
* botocore==1.12.68
 - urllib3 [required: >=1.20,<1.25, installed: 1.25.7]
* pykwalify==1.5.1
 - python-dateutil [required: ==2.4.2, installed: 2.8.1]
 - PyYAML [required: ==3.11, installed: 5.2]
* wxPython==4.0.4
 - Pillow [required: Any, installed: ?]
* zeep==3.2.0
 - requests-toolbelt [required: >=0.7.1, installed: 0.4.0]
```

Inconsistencies from my own Python installation



```
[17:33:09] ~/Desktop> pipconflictchecker
--------------------------------------------------
 Conflicts Detected
--------------------------------------------------
 - requests-toolbelt(0.4.0) zeep(>=0.7.1)
 - urllib3(1.25.7) botocore(<1.25,>=1.20)
```

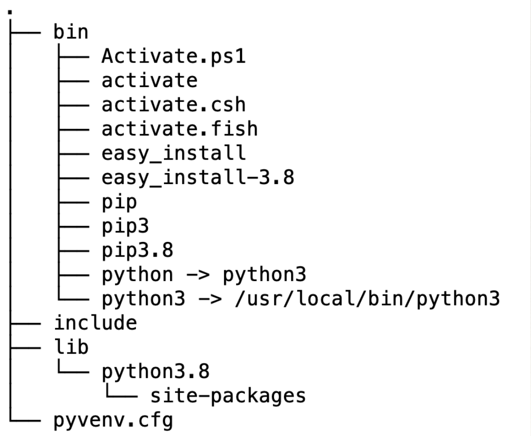
Virtual Environment

- Resolves many dependency issues since you can use different versions of a package for different projects.
- Makes your project self-contained and reproducible by capturing all package dependencies in a requirements file.
- Other benefits
  - You can install packages on a host on which you do not have admin privileges. (not a big deal, we do have "pip install —user")
  - Keep your global site-packages tidy by not including packages needed for one project.

Source: https://towardsdatascience.com/virtual-environments-104c62d48c54

Virtual Environment

- A self-contained directory tree with
  - a Python installation for a particular version of Python, using symbolic links to Python executables on your system
  - additional packages in a site-packages folder where third party libraries are installed
  - some scripts to set environment variables



```
.
├── bin
│   ├── Activate.ps1
│   ├── activate
│   ├── activate.csh
│   ├── activate.fish
│   ├── easy_install
│   ├── easy_install-3.8
│   ├── pip
│   ├── pip3
│   ├── pip3.8
│   ├── python -> python3
│   └── python3 -> /usr/local/bin/python3
├── include
├── lib
│   └── python3.8
│       └── site-packages
└── pyvenv.cfg
```

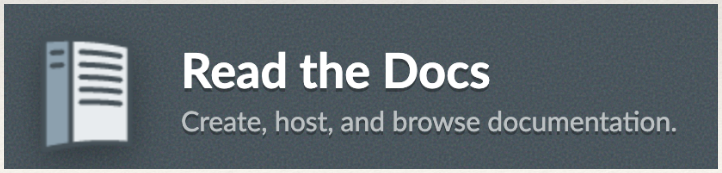
Releasing your code

- Use Github tags and releases
- Documentation
  - GitHub pages
    - https://pages.github.com/
    - Markdown, Jekyll
    - website for your project
  - Read the Docs
    - https://readthedocs.org/
    - Sphinx, Markdown and reStructuredText
    - hosts documentation

GitHub Pages

Websites for you and your projects.

Hosted directly from your GitHub repository. Just edit, push, and your changes are live.

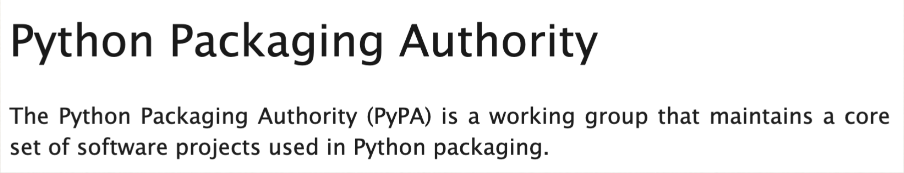


Read the Docs

Create, host, and browse documentation.

PyPI - Python Package Index

- PyPA - Python Packaging Authority



Python Packaging Authority

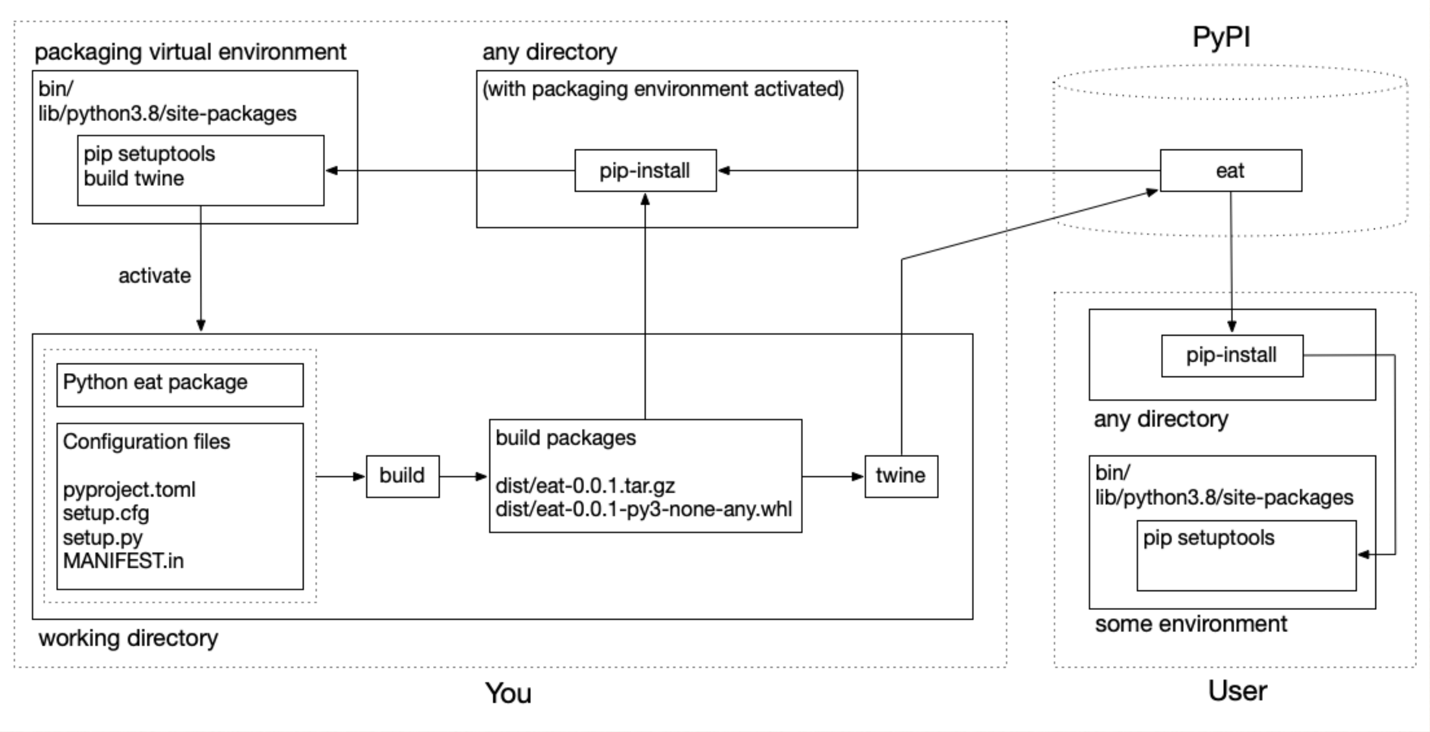
The Python Packaging Authority (PyPA) is a working group that maintains a core set of software projects used in Python packaging.

- PyPI - Python Package Index
  - a repository of software for Python

Packaging

- Simplest
  - Dump all code in an archive with some notes on how to run it
- Actual Python packages
  - Add `__init__.py` file and think about organization
- Use “`python -m build`” to create an archive from the package
- Put the archive on pypi.org
- Use pip to install local or remote package

Packaging



https://github.com/marcverhagen/packaging-tutorial-part1