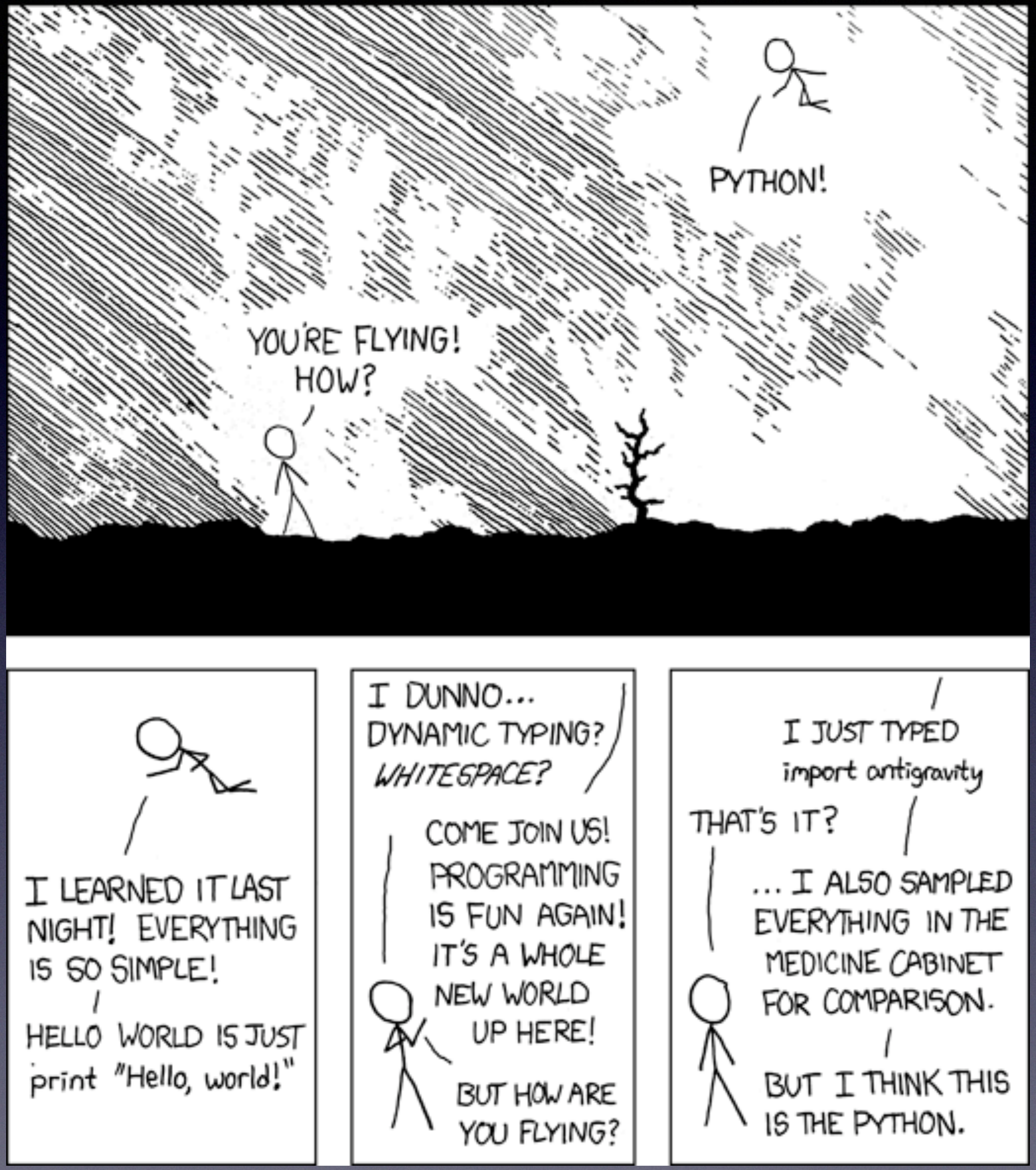
Publishing a Python Package

Brendan Herger, https://www.hergertarian.com/

Slides: https://goo.gl/c5YCRE


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

https://xkcd.com/353/

Intro

Goals

- Template for efficiently publishing a package
- Understanding of why some packages are … wonky
- Appreciation for the underbelly of Python

Goals

- Template for efficiently publishing a package
- Understanding of why some packages are … wonky
- Appreciation for the underbelly of Python

The Python source distribution has long maintained the philosophy of "batteries included”: … This gives the Python language a head start in many projects.

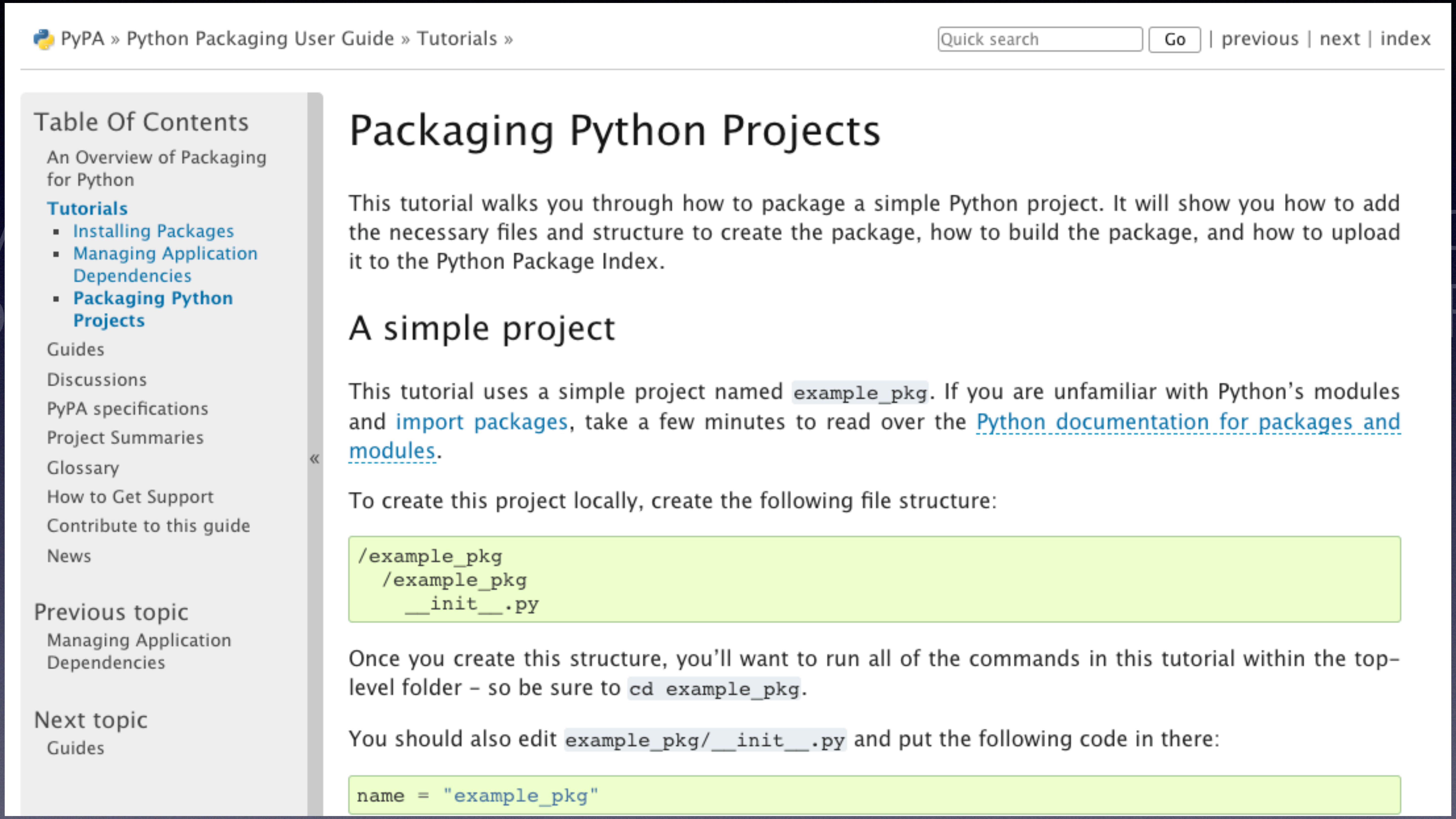
–PEP 206

The Python source distribution has long maintained the philosophy of "batteries included”: … This gives the Python language a head start in many projects.

?

–PEP 206

The Py d the
philosop Python



PyPA » Python Packaging User Guide » Tutorials »

Quick search Go | previous | next | index

Packaging Python Projects

This tutorial walks you through how to package a simple Python project. It will show you how to add the necessary files and structure to create the package, how to build the package, and how to upload it to the Python Package Index.

A simple project

This tutorial uses a simple project named `example_pkg`. If you are unfamiliar with Python's modules and import packages, take a few minutes to read over the Python documentation for packages and modules.

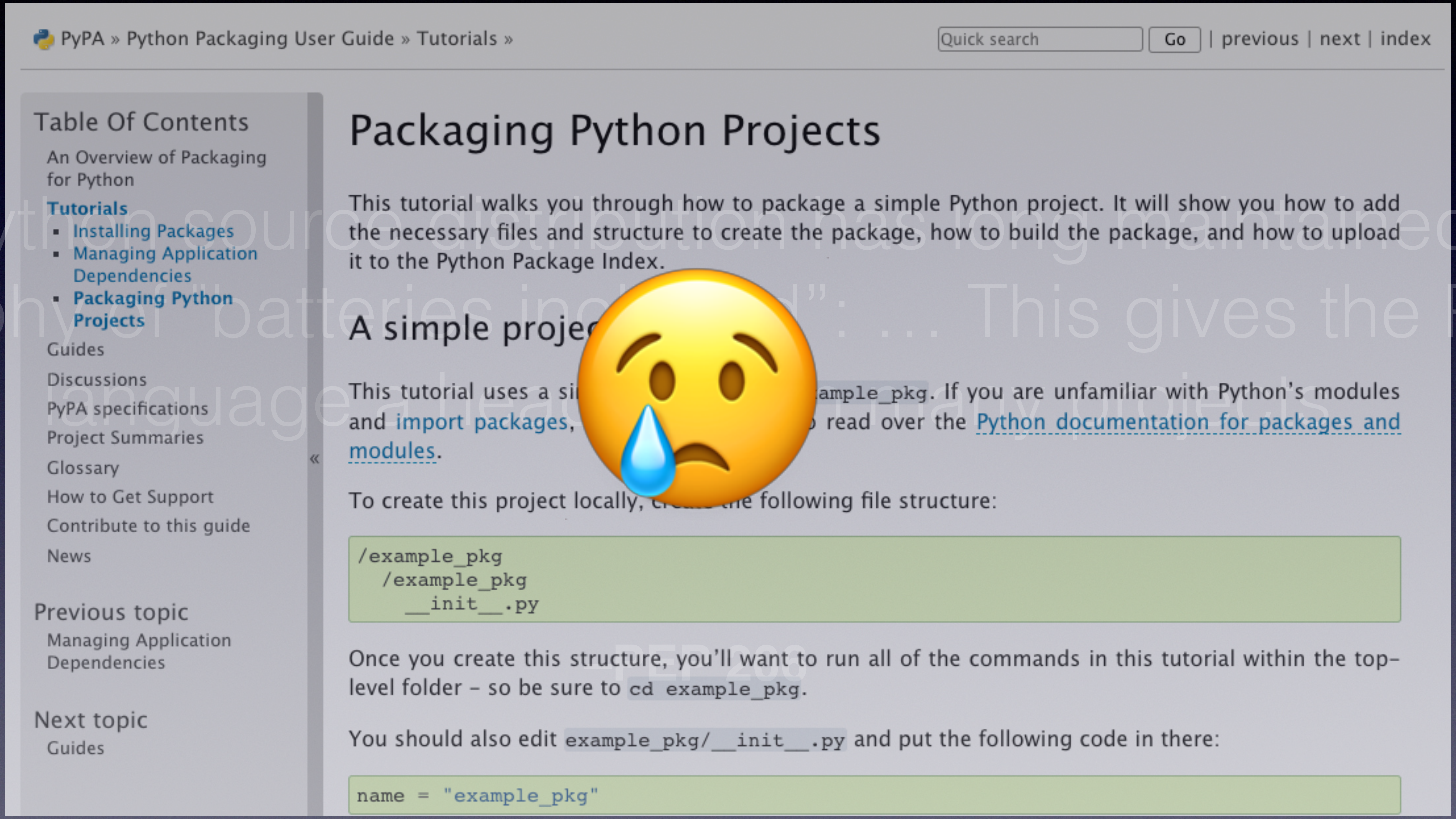
To create this project locally, create the following file structure:

```
/example_pkg
  /example_pkg
    __init__.py
```

Once you create this structure, you'll want to run all of the commands in this tutorial within the top-level folder – so be sure to `cd example_pkg`.

You should also edit `example_pkg/__init__.py` and put the following code in there:

```
name = "example_pkg"
```



PyPA » Python Packaging User Guide » Tutorials »

Packaging Python Projects

This tutorial walks you through how to package a simple Python project. It will show you how to add the necessary files and structure to create the package, how to build the package, and how to upload it to the Python Package Index.

A simple projec

This tutorial uses a si ample_pkg. If you are unfamiliar with Python's modules and import packages, read over the Python documentation for packages and modules.

To create this project locally, e following file structure:

```
/example_pkg
  /example_pkg
    __init__.py
```

Once you create this structure, you'll want to run all of the commands in this tutorial within the top-level folder – so be sure to `cd example_pkg`.

You should also edit `example_pkg/__init__.py` and put the following code in there:

```
name = "example_pkg"
```

‘Normal’ Workflow

‘Normal’ Workflow

- **Design:** What are we building? What do the files and functions look like?
- **Write:** The actual hard work
- **Structure:** PyPI requires some extra stuff
- **Release:** Share with the world

Cheat sheet

Cheat sheet

- **Design:**
  - Whiteboard out files, classes and functions
  - Explore a user journey
  - What would the ‘Getting started’ guide look like?

Cheat sheet

- **Write:**
  - **Documentation:** Choose a documentation format, such as Sphinx's rST (as suggested in PEP 287). Write a README, for people who don't work in the next desk over
  - **KISS:** Keep it simple, stupid
  - **Backlog:** Do all of the need to have's. Keep track of the nice to haves, future work
  - **Testing:** Write unittests (bonus points!)

Cheat sheet

- **Structure:**
  - **setup.py:** The setup.py file is the core of describing how your code becomes a package. Less is more
  - **Dry-run and dog-food:** Run unittests, try installing your package locally. Then try using your package
  - **Continuous Integration:** Setup CI (such as Travis or CircleCI) to regularly run your unittests

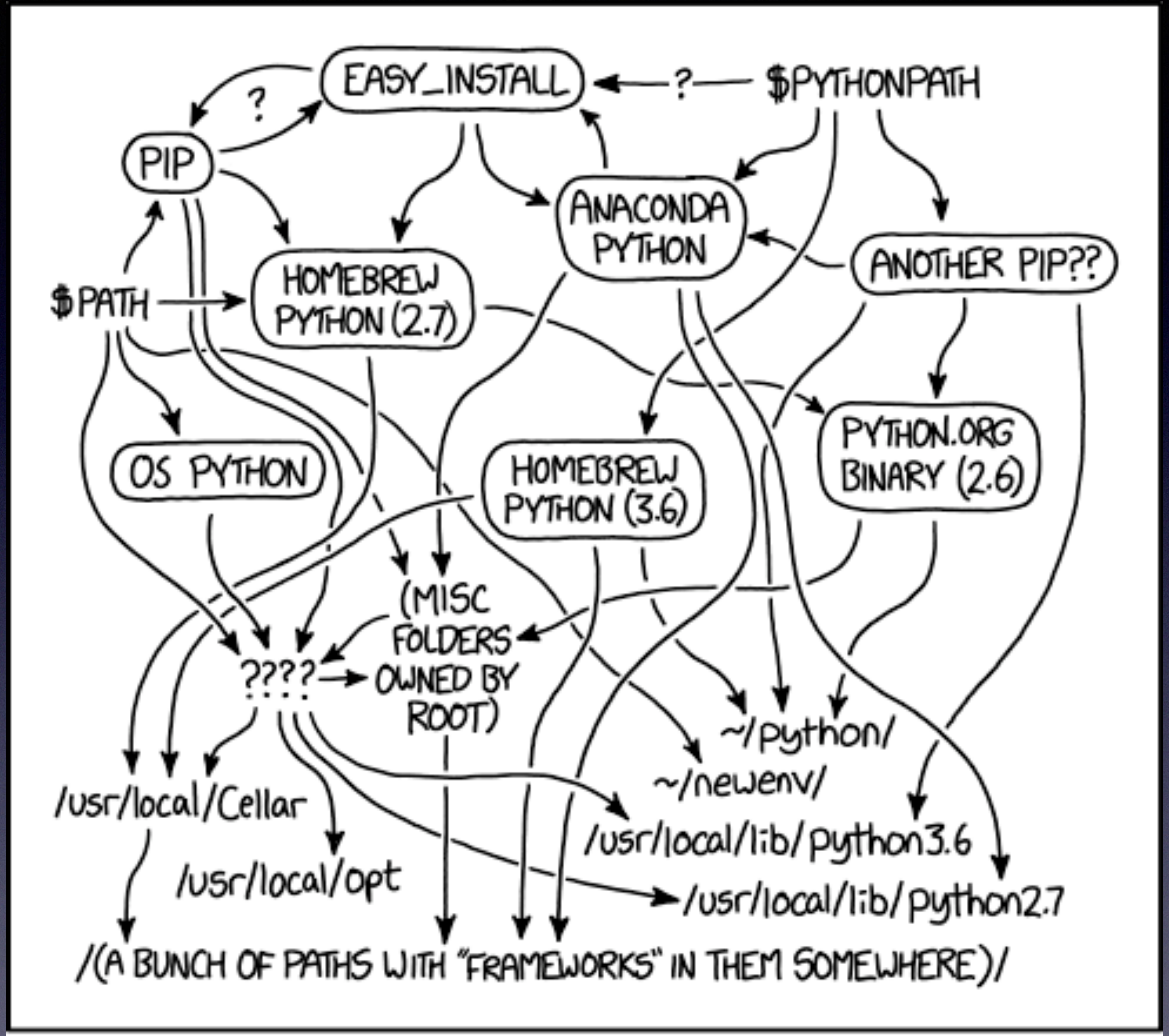
Cheat sheet

- **Release:**
  - **Package:** You’ll likely use twine to create wheels, releasable files
  - **Upload:** Create a PyPI account, use that account to upload your files
  - **Celebrate!**

Cheat sheet

- **Release:**
  - **Package:** You’ll likely use tw[illegible]create wheels, releasable files
  - **Upload:** Create a PyPI account, use that account to upload your files
  - **Celebrate!**

General Ramblings



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

https://xkcd.com/1987/

General Ramblings

- **Batteries included:** Is for users only. The underbelly of the beast is a dangerous place
- **Imitation is the sincerest form of flattery**
- **Documentation formats:** rST, Numpy, Google, or build your own!
- **Documentation generation:** (bonus points) Sphinx is the standard for creating nice documentation pages. readthedocs is the standard for hosting them
-

Recap

Recap

- Python is a batteries included language, except for when you're the one building the package
- The process isn't difficult, but also isn't well documented
- Imitation is the sincerest form of flattery

Thanks!

Brendan Herger, https://www.hergertarian.com/
Slides: https://goo.gl/c5YCRE