Data Science Survival Skills

Version Control and Python Package Management

Exercise: Opening JPGs and opening JPGs

Decoding: Trade accuracy for speed...

IMAGEIO

OPENCV

Looks the same, but not px by px!!

Exercises **UPDATE**

- You have 1 full week (Mo after lecture to Mo before lecture)
- If you're too late… you're too late. No exceptions. You had enough time.
- You only get the bonus point, if you tackled and tried with effort **all** tasks.
- Friday we will have Q&As, further examples, tipps&tricks etc.
- The solution will be provided as soon as possible after the deadline.

Bonus points are indicated in **StudOn** using the "Bestanden/Passed" option for each exercise.

Agenda

- The dilemma of code versions
- Version control concepts
- Git
- Github/Gitlab
- Pypi
- A Python package
- Documentation (!)

“Sure, I just need 10 lines of code...”

Here you go!

Thing for guy - coffee.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[6]
```
import numpy as np
```

[7]
```
x = np.random.randint(0,255, (256, 256, 3))
```

[8]
```
def conv(x):
  if len(x.shape)==2:
    return None
  return x @ (0.2126, 0.7152, 0.0722)
```

[10]
```
xp = conv(x).astype(np.uint8)
print(xp.shape, xp.dtype)
```

```
(256, 256) uint8
```

Here you go!!

Thing for guy - coffee 2.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

[6]
```
import numpy as np
```

[7]
```
x = np.random.randint(0,255, (256, 256, 3))
```

```
def conv(x):
  if len(x.shape)==2:
    return x
  return x @ (0.2126, 0.7152, 0.0722)
```

[10]
```
xp = conv(x).astype(np.uint8)
print(xp.shape, xp.dtype)
```

```
(256, 256) uint8
```

Versioning

- Thesis.docx
- Thesis_1.docx
- Thesis_1_anki.docx
- Thesis_2.docx
- Thesis_2_anki.docx
- Thesis_2_AB.docx
- .....
- Thesis_final.docx
- Thesis_final_anki.docx
- Thesis_final2.docx
- Thesis_final2_fix.docx

?!?

Software versioning

Semantic versioning

Example: Python versioning, e.g. 2.7 and 3.10 → major change may indicate incompatibilities and breaking changes!

More on SemVar

Comparison of development stage indicators

Stage	Semver	Num. Status	Num 90+
Alpha	1.2.0-a.1	1.2.0.1	1.1.90
Beta	1.2.0-b.2	1.2.1.2	1.1.93
Release candidate	1.2.0-rc.3	1.2.2.3	1.1.97
Release	1.2.0	1.2.3.0	1.2.0
Post-release fixes	1.2.5	1.2.3.5	1.2.5

Alpha, Beta, Release Candidate

Alpha

- Internally tested
- Not feature complete
- Serious performance issues
- (un)known bugs
- Software may crash often

Beta

- Feature complete
- Contains significant less bugs
- Still performance, speed issues
- Can also crash and data loss may occur
- Interaction with users
  ➔ usability testing

Release Candidate (SILVER)

- Final beta product with acceptable bugs
- Minimal interaction with source code and documentation

Release (GOLD)

- Release to Manufacturing (RTM)
- Digitally signed
  ➔ knowing product state

Glottis Analysis Tools

How can I track more meaningful file versions?

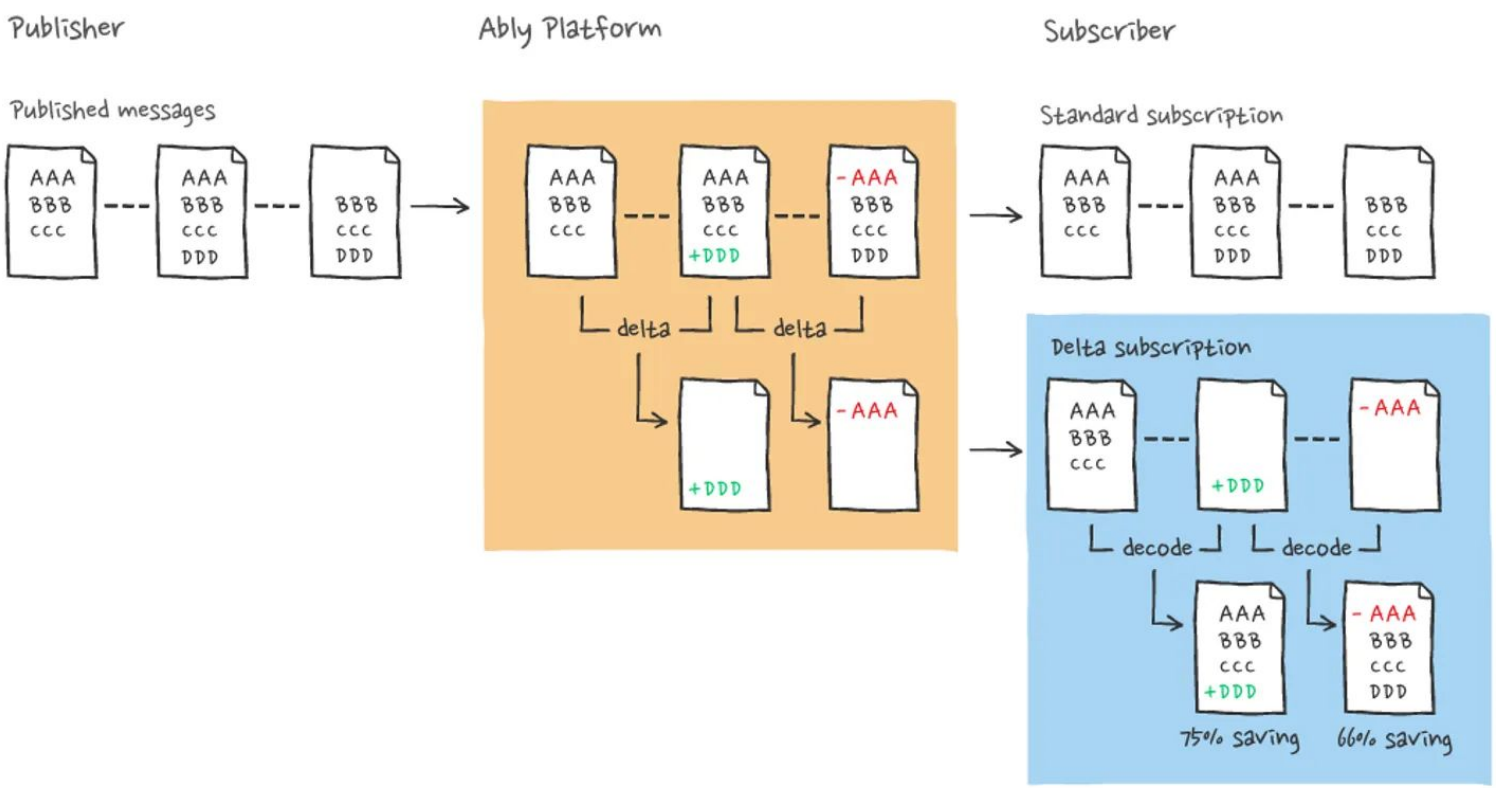
Is there some kind of “version control”?

CVS - Concurrent Versions System

Introduction | News | Documentation | Get the Software | Help and Bug Reports | Development

Already developed in the 80s.
The store changes using **delta compression!**

Delta compression



https://ably.com/blog/practical-guide-to-diff-algorithms

Apache Subversion (SVN)

- Tried to be successor to CVS
- Is used in the following projects:
  Clang, FreeBSD, GCC...
- Fixed a lot of previous bugs in CVS
  And implemented more features

Issue:

- Renaming is copy&delete that is fed back to complete file history ➔ could break things in older versions

The BitKeeper controversy (early 2000s)

BitKeeper: “You can use BitKeeper free of charge for cool freeware and open source project”*

*if you are not actively supporting any competitor to BitKeeper

Used in the Linux kernel by some people...

Andrew Tridgell reverse engineered BitKeeper protocol to create “SourcePuller” -> BitKeeper revoked free licenses.....

Git

https://github.blog/2020-12-17-commits-are-snapshots-not-diffs/

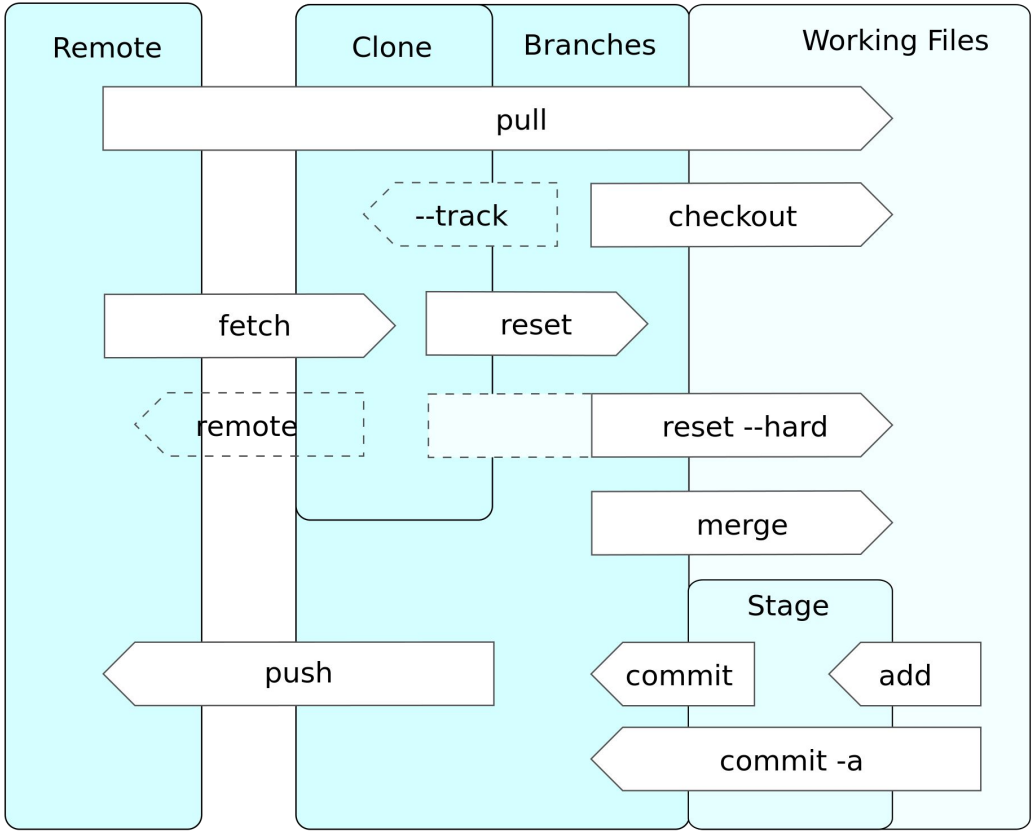
Developed in 1 Month (April 2005)
Powered the Kernel release 2.6.12 release (June 2005)

The Git principle

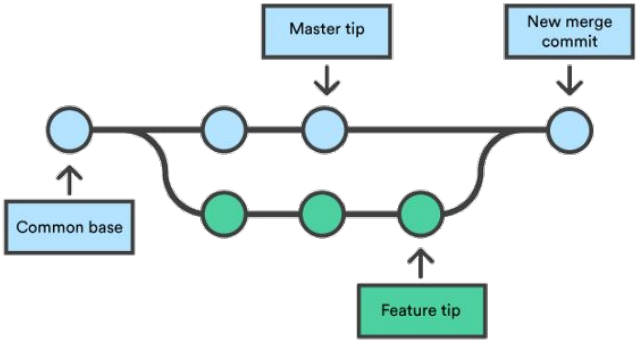
git --everything-is-local

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

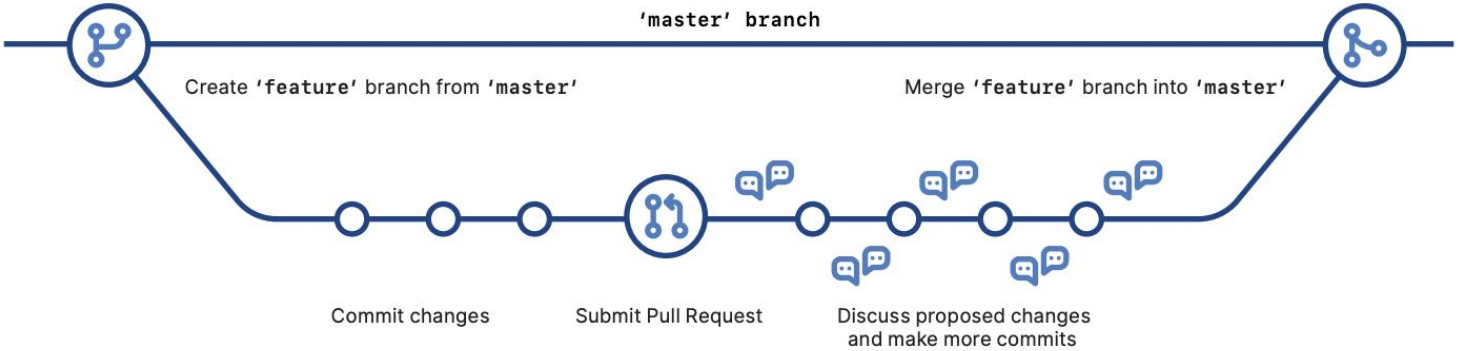
Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.



The branching principle



GitHub Flow



Where to store these “repositories”?

GitLab vs. GitHub Comparison Challenge

	GitLab	GitHub	Missing in GitHub
Manage	3/6	1/6	DevOps Score, Audit Management Across DevOps Lifecycle
Plan	5/7	2/7	Time Tracking, Agile Portfolio Management, Service Desk
Create	6/7	4/7	Web IDE, Design Management
Verify	3/7	2/7	Performance Testing
Package	3/4	3/4	Dependency Proxy
Secure	6/8	3/8	DAST, Container Scanning, License Compliance, Security Dashboard
Release	6/8	3/8	Review Apps, Feature Flags, Incremental Roll Outs, Native K8s Integration
Configure	5/8	0/8	AutoDevOps, ChatOps, Runbook Config
Monitor	6/8	0/8	Logging, Tracing, Cluster Monitoring, Alerting
Defend	1/5	0/5	Web App Firewall

Which product is best for you? Share your review on twitter with **#GitChallenge**

Comparing GitLab Terminology

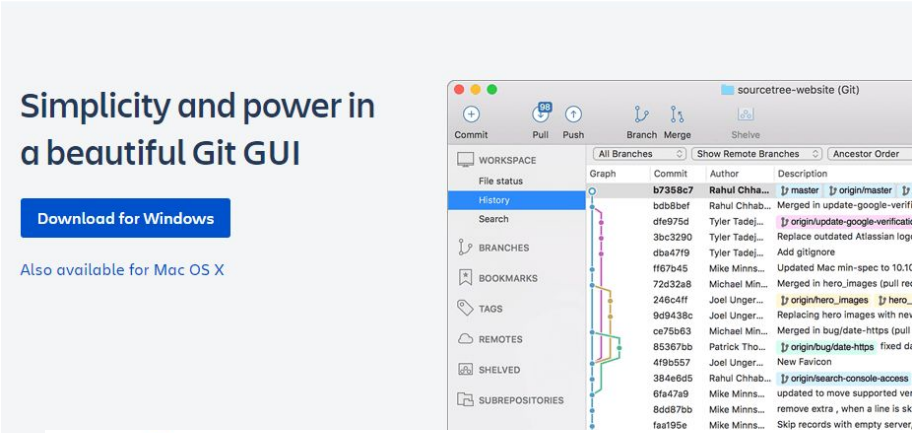
Bitbucket	GitHub	GitLab	So, what does it mean?
Pull Request	Pull Request	Merge Request	In GitLab a request to merge a feature branch into the official master is called a Merge Request.
Snippet	Gist	Snippet	Share snippets of code. Can be public, internal or private.
Repository	Repository	Project	In GitLab a Project is a container including the Git repository, discussions, attachments, project-specific settings, etc.
Teams	Organizations	Groups	In GitLab, you add projects to groups to allow for group-level management. Users can be added to groups and can manage group-wide notifications.

Example repository

https://github.com/anki-xyz/pipra

Git software

Sourcetree



TortoiseGit
Windows Shell Interface to Git

About Download Support Contribute

Storing Python code

navdeep-G Update README.rst d469f2f on 20 Jul 2019 29 commits

docs	basics	10 years ago
sample	Lets allow the helpers to be helpfull	5 years ago
tests	Lets allow the helpers to be helpfull	5 years ago
.gitignore	add a Python gitignore	5 years ago
LICENSE	Update LICENSE	5 years ago
MANIFEST.in	need to include the LICENSE file, otherwise pypi installs are broke...	5 years ago
Makefile	Update Makefile	6 years ago
README.rst	Update README.rst	2 years ago
requirements.txt	basics	10 years ago
setup.py	Update setup.py	4 years ago

Sample Repository

tl;dr: This is what Kenneth Reitz recommended in 2013.

This repository is available on GitHub.

```
README.rst
LICENSE
setup.py
requirements.txt
sample/__init__.py
sample/core.py
sample/helpers.py
docs/conf.py
docs/index.rst
tests/test_basic.py
tests/test_advanced.py
```

https://docs.python-guide.org/writing/structure/#structure-of-code-is-key

Content of repo

navdeep-G Update README.rst d469f2f on 20 Jul 2019 29 commits

docs	basics	10 years ago
sample	Lets allow the helpers to be helpfull	5 years ago
tests	Lets allow the helpers to be helpfull	5 years ago
.gitignore	add a Python gitignore	5 years ago
LICENSE	Update LICENSE	5 years ago
MANIFEST.in	need to include the LICENSE file, otherwise pypi installs are broke...	5 years ago
Makefile	Update Makefile	6 years ago
README.rst	Update README.rst	2 years ago
requirements.txt	basics	10 years ago
setup.py	Update setup.py	4 years ago

Documentation

Location	`./docs/`
Purpose	Package reference documentation.

The Actual Module

Location	`./sample/` or `./sample.py`
Purpose	The code of interest

Test Suite

For advice on writing your tests, see Testing Your Code.

Location	`./test_sample.py` or `./tests`
Purpose	Package integration and unit tests.

License

Location	`./LICENSE`
Purpose	Lawyering up.

Setup.py

Location	`./setup.py`
Purpose	Package and distribution management.

Requirements File

Location	`./requirements.txt`
Purpose	Development dependencies.

Licenses

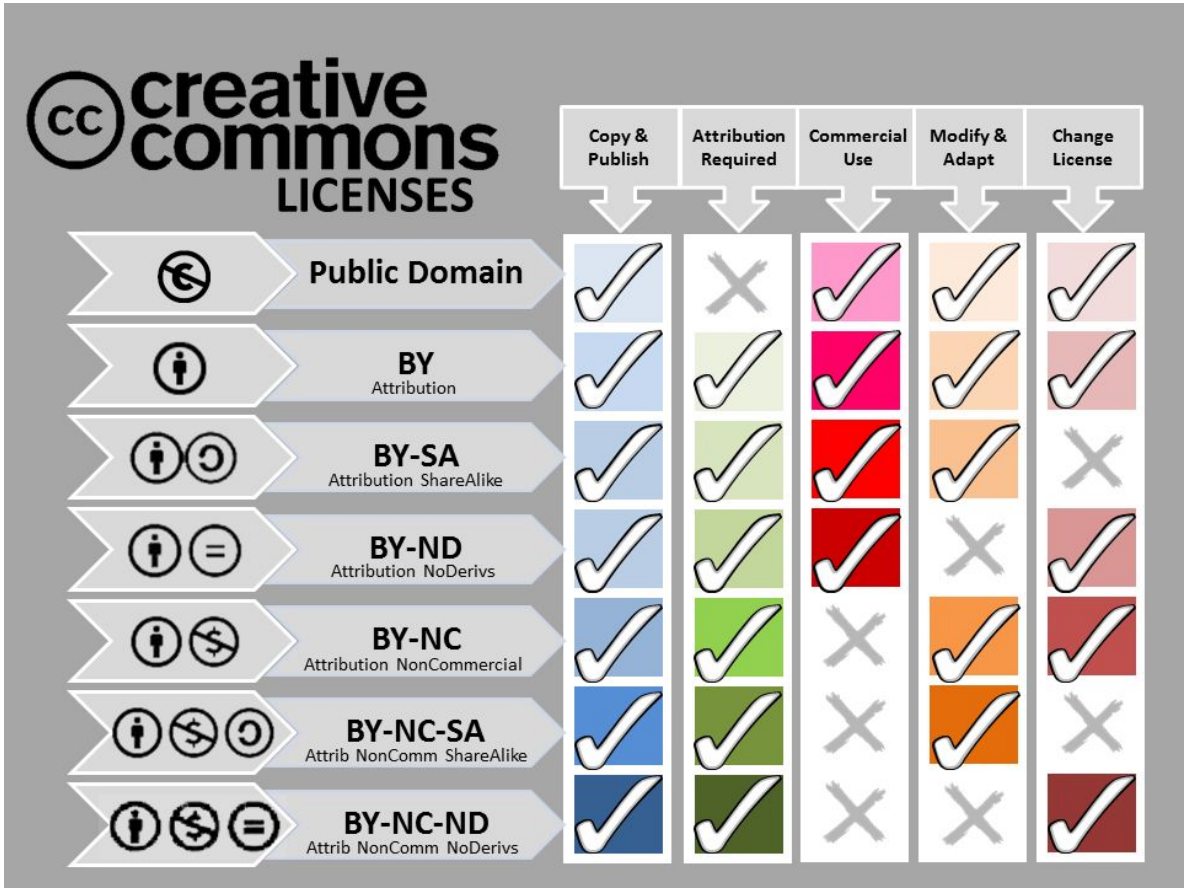


Table 2. Ranking of FOSS licenses' degree of *Openness* based on CC elements.

	Share Alike	No Derives	Noncommercial	Attribution	Ranking of *Openness*
GPL	Yes	No	No	Yes	1
LGPL	Yes	No	No	Yes	1
MPL	Yes	No	No	Yes	1
QPL	No	No	No	Contingent[20]	2
CPL	No	No	No	Contingent	2
Artistic	No	No	No	Contingent[21]	2
Apache v.2.0	No	No	No	Yes	3
zlib	No	No	No	Yes	3
Apache v.1.1	No	No	No	Yes	3
BSD	No	No	No	Yes	3
MIT	No	No	No	Yes	3

https://choosealicense.com/

Licenses on Github

master pipra / LICENSE

Go to file ···

anki-xyz/pipra is licensed under the

GNU General Public License v3.0

Permissions of this strong copyleft license are conditioned on making available complete source code of licensed works and modifications, which include larger works using a licensed work, under the same license. Copyright and license notices must be preserved. Contributors provide an express grant of patent rights.

Permissions

- ✓ Commercial use
- ✓ Modification
- ✓ Distribution
- ✓ Patent use
- ✓ Private use

Limitations

- ✕ Liability
- ✕ Warranty

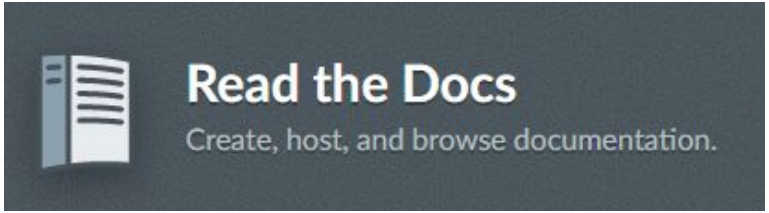
Conditions

- ⓘ License and copyright notice
- ⓘ State changes
- ⓘ Disclose source
- ⓘ Same license

This is not legal advice. Learn more about repository licenses.

Documentation

- Sphinx
- Read The Docs



https://www.writethedocs.org/guide/writing/beginners-guide-to-docs/

- You will be using your code in 6 months
- You want people to use your code
- You want people to help out
- You want your code to be better
- You want to be a better writer

How to document a function

Formatting Type	Description	Supported by Sphynx	Formal Specification
Google docstrings	Google's recommended form of documentation	Yes	No
reStructured Text	Official Python documentation standard; Not beginner friendly but feature rich	Yes	Yes
NumPy/SciPy docstrings	NumPy's combination of reStructured and Google Docstrings	Yes	Yes
Epytext	A Python adaptation of Epydoc; Great for Java developers	Not officially	Yes

Tipp: Autodocs

Every IDE has an autodoc format,
I'll show you!

Also use pylint and similar tools,
Especially ones that correct your documentation:

https://github.com/psf/black

Use cookie cutter for your projects

Caveat: I have never used this on my own though...

For Python: https://github.com/audreyfeldroy/cookiecutter-pypackage

A new python package with CookieCutter

Features

- Testing setup with `unittest` and `python setup.py test` or `pytest`
- Travis-CI: Ready for Travis Continuous Integration testing
- Tox testing: Setup to easily test for Python 3.6, 3.7, 3.8
- Sphinx docs: Documentation ready for generation with, for example, Read the Docs
- bump2version: Pre-configured version bumping with a single command
- Auto-release to PyPI when you push a new tag to master (optional)
- Command line interface using Click (optional)

PyPI - your package pip installable

The Python Package Index (PyPI) is a repository of software for the Python programming language.

PyPI helps you find and install software developed and shared by the Python community. Learn about installing packages.

Package authors use PyPI to distribute their software. Learn how to package your Python code for PyPI.

```
pip install myfancypackage
```

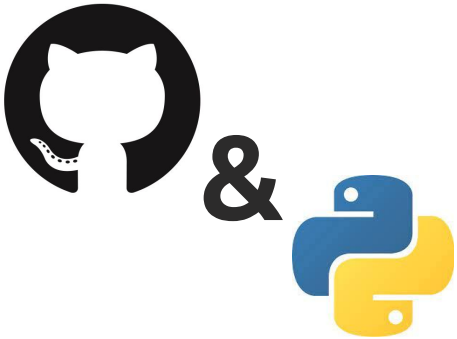
The last slide

- Version control keeps track of your code
  - Versions can be stored as delta or snapshot
  - Branching is important for adding features
  - Merging allows to combine features to a common master branch
- Your packages should have a consistent version scheme (1.2.123)
- Licenses specify how others can use your code
- Documentation is an integral part of your code
  - Distinct from commenting
  - Docs for code/functions/classes
  - Docs for tutorials, setting up, getting started, ...
- A good organized python package should be pip installable

Exercise

Description of the exercise

For this exercise, we will deal with Version Control using GitHub and package management using pip. Furthermore, we are going to learn how to create and install our packages using Git and Pip.



Creating a repo and installing it with pip

The first step is to create an account on GitHub. Then, your task is to set a new public repository!

This repo will contain the necessary structure to be installed using pip. You will create a package containing a function that we will install and check directly in our virtual environment.

Description of the exercise

- Create a Github account if you don’t have one
- You get a script idea from us with the instructions to follow.
- Create a public repository for the script and make it a pip installable.
- Ensure that you have a proper README, a license file, etc.