Two-Day MDS-Rely Workshop on Code Packaging and Release

Jungtaek Kim
jungtaek.kim@pitt.edu

University of Pittsburgh
https://jungtaekkim.github.io

February 6, 2024 / February 13, 2024

University of Pittsburgh

Table of Contents


How do you code so well?
Practice.
It must be an innate gift... A gift from God...
It's practice.
I'll never understand how some people are so talented... A mystery...
Practice.
©Sarah Andersen

University of Pittsburgh

Overview

Overview

- These sessions are all example-based.
- We will talk about code management, code refactoring, code packaging, and code release.
- Many alternatives are available, so you can choose any other frameworks.
- There is no right answer for code management and packaging.
- You can make your way to manage your project.

- If you are willing to review the code of your project, please let us know.

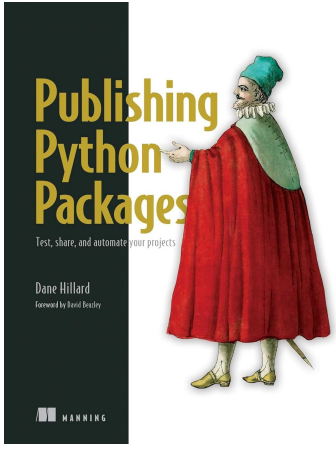
Overview

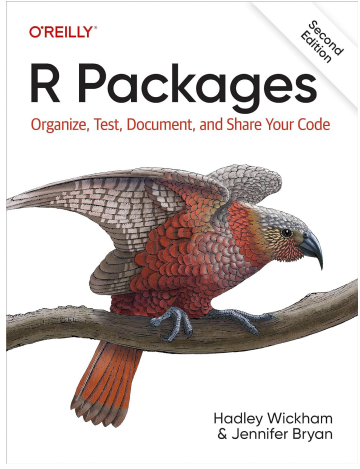
- We will look into several popular Python projects:
  - NumPy,
  - SciPy,
  - and scikit-learn.
- My Python project, BayesO will be explored.
- Your projects?

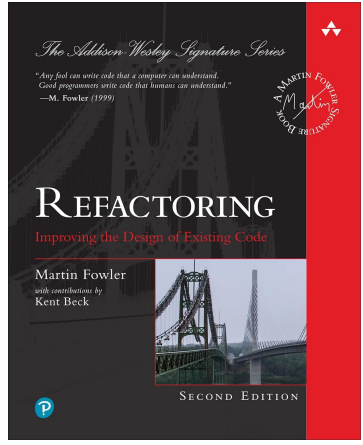
Overview

- We will cover many components for code packaging and release:
  - GitHub, Git, open-source licenses, code refactoring, code formatter, lint, unit testing, continuous integration, GitHub Actions, documentation, and Python Package Index.
- Journal of Open Source Software will be introduced.
- The goals of most of the code management an deployment tools are to
  - improve readability,
  - automate processes,
  - increase productivity,
  - speed up software development,
  - and ensure open access.

Useful Materials





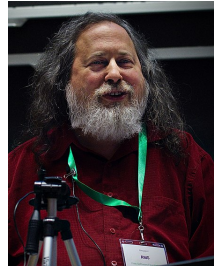


Key People in Free and Open Source Software

Bruce Perens

Eric S. Raymond

Mark Shuttleworth



Richard Stallman

Linus Torvalds

Figure 1: Alphabetically ordered.

You need to do practice!

Codebase

Codebase

- Codebase typically contains
  - main source files,
  - README,
  - LICENSE,
  - CONTRIBUTING,
  - doc,
  - unit tests,
  - lint configuration files,
  - continuous integration files,
  - and others.

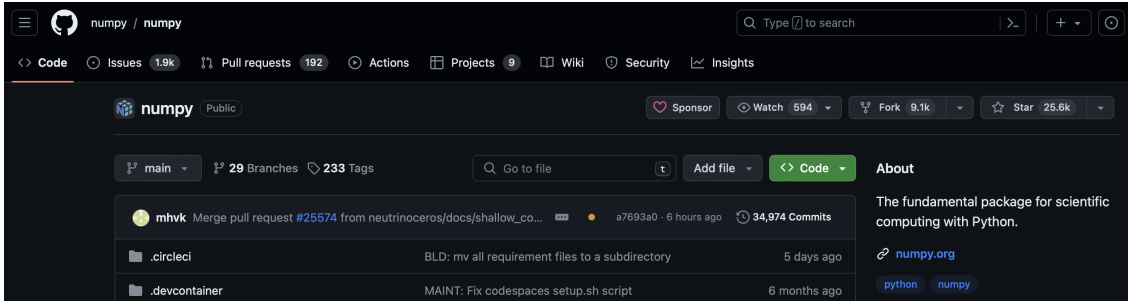
GitHub Repositories

NumPy GitHub Repository

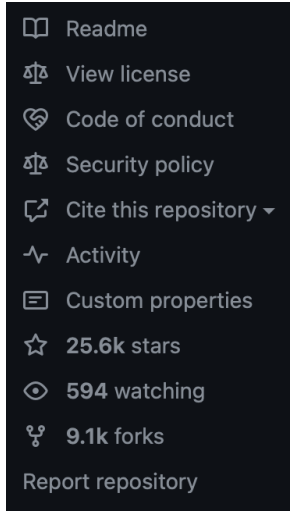
- NumPy is a Python package for scientific computing.

```
A_M = np.log(2) / 2
B_M = np.log(2250) - A_M * 1971
Moores_law = lambda year: np.exp(B_M) * np.exp(A_M * year)
```

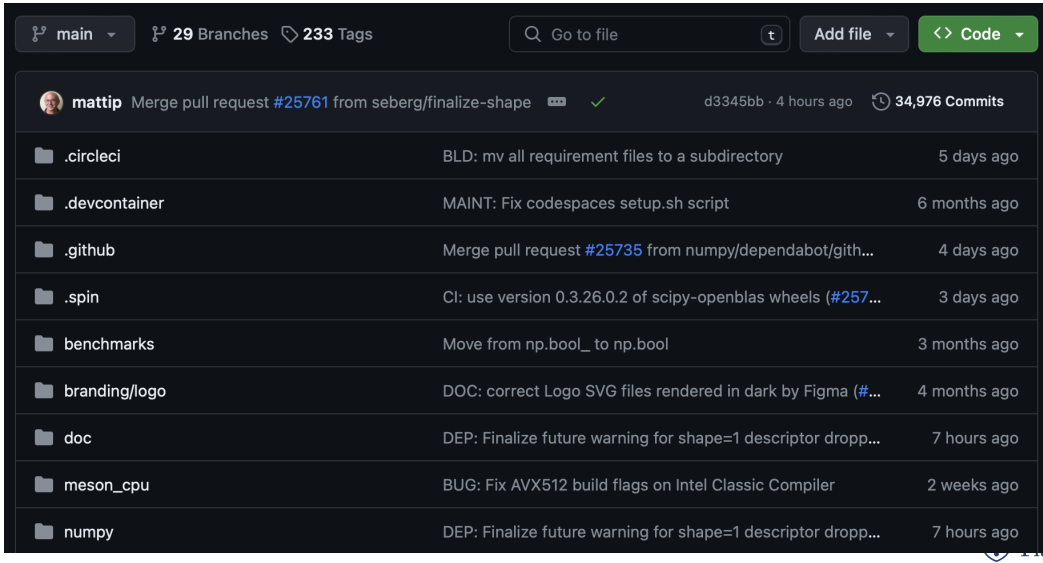
NumPy GitHub Repository



NumPy GitHub Repository



NumPy GitHub Repository



NumPy GitHub Repository

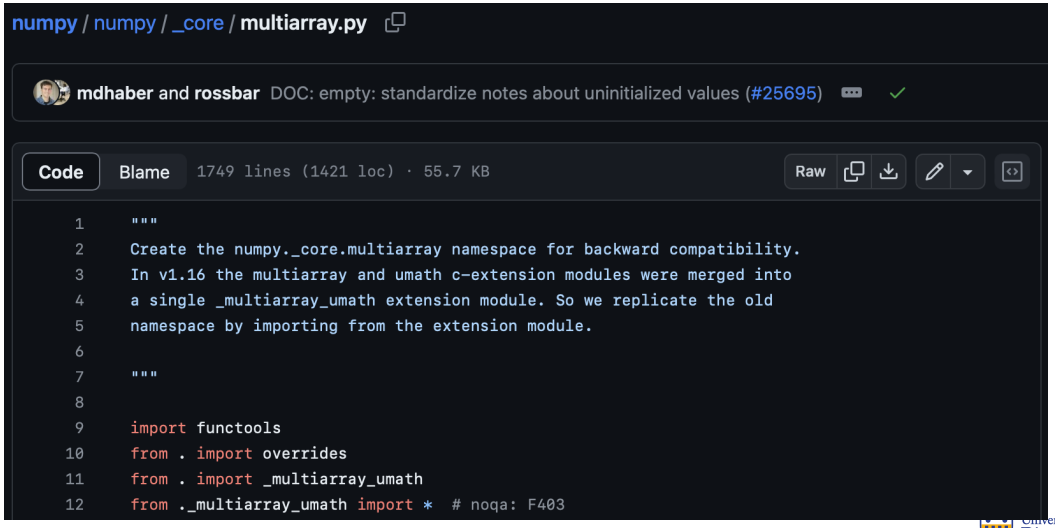
README | Code of conduct | License | Security

NumPy is the fundamental package for scientific computing with Python.

- **Website:** https://www.numpy.org
- **Documentation:** https://numpy.org/doc
- **Mailing list:** https://mail.python.org/mailman/listinfo/numpy-discussion
- **Source code:** https://github.com/numpy/numpy
- **Contributing:** https://www.numpy.org/devdocs/dev/index.html
- **Bug reports:** https://github.com/numpy/numpy/issues
- **Report a security vulnerability:** https://tidelift.com/docs/security

University of Pittsburgh

NumPy GitHub Repository



Other GitHub Repositories

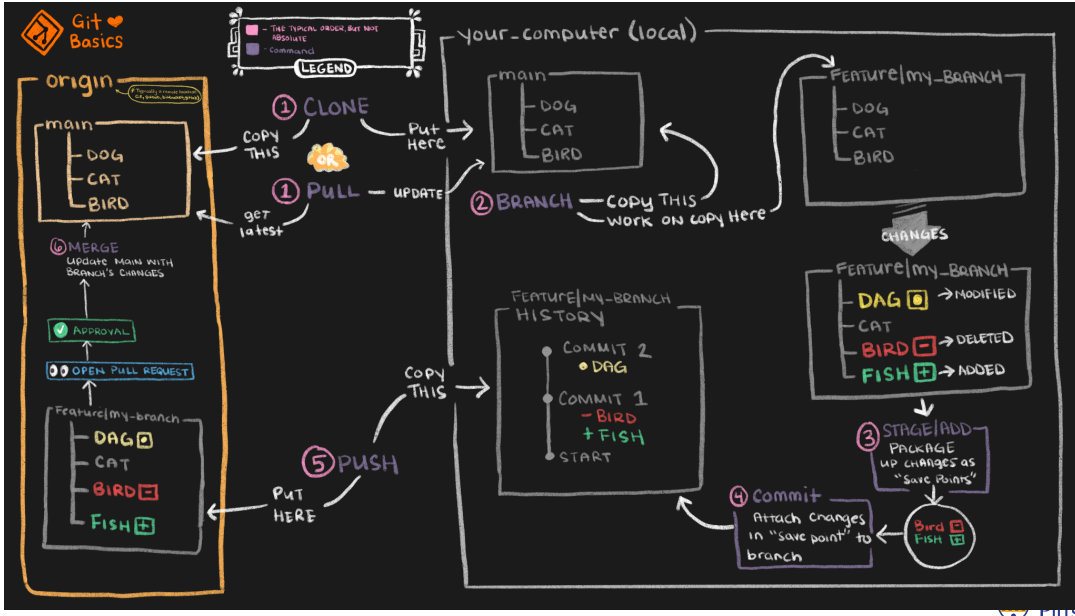
- SciPy
- scikit-learn
- BayesO

Git

Git

- Git is a distributed version control system.
- It tracks changes in any set of computer files, usually used for coordinating work among programmers.
- Git was originally authored by Linus Torvalds in 2005 for development of the Linux kernel.
- Git has become the most popular distributed version control system, with nearly 95% of developers reporting it as their primary version control system as of 2022.
- Git repositories as a service are offered by many service providers such as GitHub and BitBucket.

Taken from Wikipedia.


Git ❤ Basics
- THE TYPICAL ORDER, BUT NOT ABSOLUTE
- Command
LEGEND
origin
(Typically a remote location, e.g. GitHub, BitBucket, GitLab)
main
DOG
CAT
BIRD
⑥ MERGE
Update main with branch's changes
✓ APPROVAL
OPEN PULL REQUEST
Feature/my-branch
DAG
CAT
BIRD
FISH
① CLONE
COPY THIS
Put Here
OR
① PULL — UPDATE
get latest
Your-computer (local)
main
DOG
CAT
BIRD
② BRANCH — COPY THIS
Work on copy Here
Feature/my-branch
DOG
CAT
BIRD
CHANGES
Feature/my-branch
DAG → MODIFIED
CAT
BIRD → DELETED
FISH → ADDED
③ STAGE/ADD
Package up changes as "save points"
Bird
Fish
④ Commit
Attach changes in "save point" to branch
Feature/my-branch HISTORY
COMMIT 2
• DAG
COMMIT 1
- BIRD
+ FISH
START
COPY THIS
⑤ PUSH
Put HERE

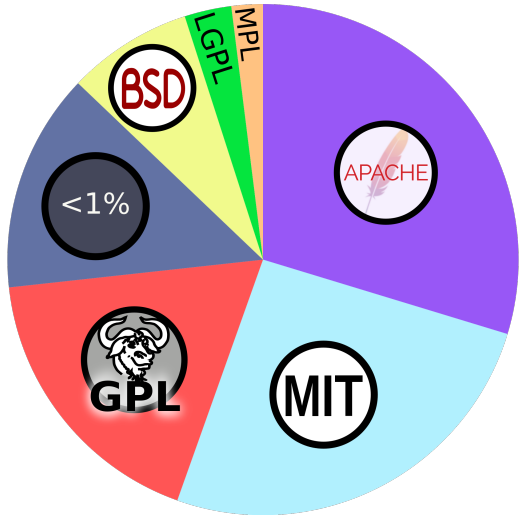
Open-Source Licenses

Open-Source Licenses

- Open-source licenses are software licenses that allow content to be used, modified, and shared.
- Free and open-source licenses use these existing legal structures for an inverse purpose.
- They grant the recipient the rights to use the software, examine the source code, modify it, and distribute the modifications.
- The two main categories of open-source licenses are permissive and copyleft.
- We often use the Open Source Initiative Approved Licenses such as the Apache 2.0 Licnese, the GNU General Public License v3, and the MIT License.

Taken from Wikipedia.

Open-Source Licenses



Open-Source Licenses

GNU AGPLv3

Permissions of this strongest 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. When a modified version is used to provide a service over a network, the complete source code of the modified version must be made available.

Permissions

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

Conditions

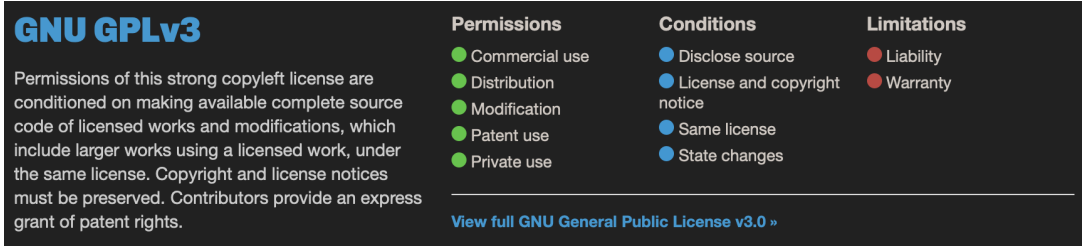
- Disclose source
- License and copyright notice
- Network use is distribution
- Same license
- State changes

Limitations

- Liability
- Warranty

View full GNU Affero General Public License v3.0 »

Open-Source Licenses



Open-Source Licenses

GNU LGPLv3

Permissions of this copyleft license are conditioned on making available complete source code of licensed works and modifications under the same license or the GNU GPLv3. Copyright and license notices must be preserved. Contributors provide an express grant of patent rights. However, a larger work using the licensed work through interfaces provided by the licensed work may be distributed under different terms and without source code for the larger work.

Permissions

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

Conditions

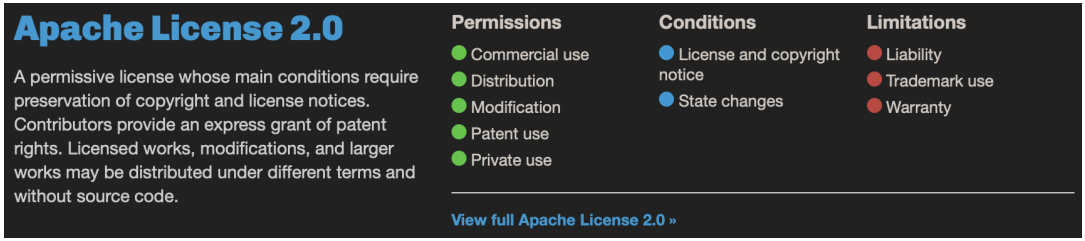
- Disclose source
- License and copyright notice
- Same license (library)
- State changes

Limitations

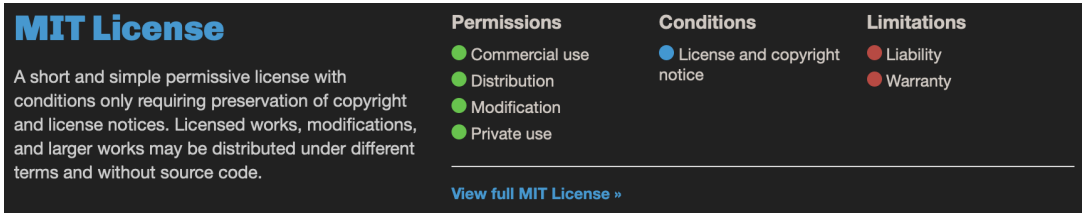
- Liability
- Warranty

View full GNU Lesser General Public License v3.0 »

Open-Source Licenses



Open-Source Licenses



NumPy License: Modified BSD License

README | Code of conduct | License | Security

```
Copyright (c) 2005-2024, NumPy Developers.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

    * Redistributions of source code must retain the above copyright
       notice, this list of conditions and the following disclaimer.

    * Redistributions in binary form must reproduce the above
       copyright notice, this list of conditions and the following
       disclaimer in the documentation and/or other materials provided
       with the distribution.

    * Neither the name of the NumPy Developers nor the names of any
       contributors may be used to endorse or promote products derived
       from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

SciPy License: BSD License

README | Code of conduct | **BSD-3-Clause license** | Security

```
Copyright (c) 2001-2002 Enthought, Inc. 2003-2024, SciPy Developers.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above
   copyright notice, this list of conditions and the following
   disclaimer in the documentation and/or other materials provided
   with the distribution.

3. Neither the name of the copyright holder nor the names of its
   contributors may be used to endorse or promote products derived
   from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

University of Pittsburgh

scikit-learn License: BSD License

README | Code of conduct | **BSD-3-Clause license** | Security

BSD 3-Clause License

Copyright (c) 2007-2023 The scikit-learn developers.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

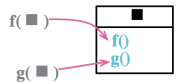
Code Refactoring

Code Refactoring

- Code refactoring is the process of restructuring existing computer code without changing its external behavior.
- It is intended to improve the design, structure, and/or implementation of the software, while preserving its functionality.
- Potential advantages of code refactoring may include improved code readability and reduced complexity.
- Another potential goal for refactoring is improved performance by writing programs that perform faster or use less memory.
- If done well, it may help developers discover and fix hidden or dormant bugs in the system by simplifying the underlying logic and eliminating unnecessary complexity levels.
- If done poorly, it may fail the requirement that external functionality not be changed, and may thus introduce new bugs.

Taken from Wikipedia.

Combine Functions into Class




```
function base(aReading) {...}
function taxableCharge(aReading) {...}
function calculateBaseCharge(aReading) {...}
```

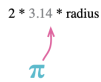
```
class Reading {
  base() {...}
  taxableCharge() {...}
  calculateBaseCharge() {...}
}
```

Rename Variable


```
let a = height * width;
```

```
let area = height * width;
```

Replace Magic Literal




```
function potentialEnergy(mass, height) {
  return mass * 9.81 * height;
}
```

```
const STANDARD_GRAVITY = 9.81;
function potentialEnergy(mass, height) {
  return mass * STANDARD_GRAVITY * height;
}
```

Inline Variable

```
let basePrice = anOrder.basePrice;
return (basePrice > 1000);
```

```
return anOrder.basePrice > 1000;
```

Introduce Assertion


```
if (this.discountRate)
  base = base - (this.discountRate * base);
```

```
assert(this.discountRate >= 0);
if (this.discountRate)
  base = base - (this.discountRate * base);
```

Code Formatter

Code Formatter

- Black is one of the most popular code formatter in Python.
- It allows us to automatically reformat code in order to satisfy code convention.
- It can save much time to format code and increase code readability.

Black

```
# in:

j = [1,
     2,
     3
]

# out:

j = [1, 2, 3]
```

Black

```
# in:

ImportantClass.important_method(exc, limit, lookup_lines, capture_locals, extra_argument)

# out:

ImportantClass.important_method(
    exc, limit, lookup_lines, capture_locals, extra_argument
)
```

Black

```python
# in:

def very_important_function(template: str, *variables, file: os.PathLike, engine: str, heade
    """Applies `variables` to the `template` and writes to `file`."""
    with open(file, 'w') as f:
        ...

# out:

def very_important_function(
    template: str,
    *variables,
    file: os.PathLike,
    engine: str,
    header: bool = True,
    debug: bool = False,
):
    """Applies `variables` to the `template` and writes to `file`."""
    with open(file, "w") as f:
        ...
```

Black Playground

Black v24.1.0 - The uncompromising Python code formatter.

Playground built by José Padilla

```python
from seven_dwwarfs import Grumpy, Happy, Sleepy, Bashful, Sneezy, Dopey, Doc
x = {  'a':37,'b':42,

'c':927}

x = 123456789.123456789E123456789

if very_long_variable_name is not None and \
 very_long_variable_name.field > 0 or \
 very_long_variable_name.is_debug:
 z = 'hello '+'world'
else:
 world = 'world'
 a = 'hello {}'.format(world)
 f = rf'hello {world}'
if (this
and that): y = 'hello ''world'#FIXME: https://github.com/psf/black/issues/26
class Foo  (     object  ):
  def f    (self    ):
    return       37*-2
  def g(self, x,y=42):
      return y
def f  (   a: List[ int ]) :
  return       37-a[42-u :  y**3]
def very_important_function(template: str,*variables,file: os.PathLike,debug:bool=False,):
    """Applies `variables` to the `template` and writes to `file`."""
    with open(file, "w") as f:
      ...
# fmt: off
custom_formatting = [
    0,  1,  2,
    3,  4,  5,
    6,  7,  8,
]
# fmt: on
regular_formatting = [
    0,  1,  2,
    3,  4,  5,
    6,  7,  8,
]
```

```python
from seven_dwwarfs import Grumpy, Happy, Sleepy, Bashful, Sneezy, Dopey, Doc

x = {"a": 37, "b": 42, "c": 927}

x = 123456789.123456789e123456789

if (
    very_long_variable_name is not None
    and very_long_variable_name.field > 0
    or very_long_variable_name.is_debug
):
    z = "hello " + "world"
else:
    world = "world"
    a = "hello {}".format(world)
    f = rf"hello {world}"
if this and that:
    y = "hello " "world"  # FIXME: https://github.com/psf/black/issues/26


class Foo(object):
    def f(self):
        return 37 * -2

    def g(self, x, y=42):
        return y


def f(a: List[int]):
    return 37 - a[42 - u : y**3]


def very_important_function(
    template: str,
    *variables,
    file: os.PathLike,
    debug: bool = False,
):
    """Applies `variables` to the `template` and writes to `file`."""
    with open(file, "w") as f:
        ...
```

Lint

Lint

- Lint is a static code analysis tool used to flag programming errors, bugs, stylistic errors and suspicious constructs.
- The term originates from a Unix utility that examined C language source code.
- The analysis performed by lint-like tools can also be performed by an optimizing compiler, which aims to generate faster code.
- Lint-like tools are especially useful for dynamically typed languages like JavaScript and Python.
- Because the compilers of such languages typically do not enforce as many and as strict rules prior to execution, linter tools can also be used as simple debuggers for finding common errors (e.g. syntactic discrepancies) as well as hard-to-find errors such as heisenbugs (drawing attention to suspicious code as "possible errors").

Taken from Wikipedia.

Pylint

--argument-naming-style

Naming style matching correct argument names.

Default: `snake_case`

Pylint

--bad-names

Bad variable names which should always be refused, separated by a comma.

Default: `('foo', 'bar', 'baz', 'toto', 'tutu', 'tata')`

Pylint

--class-const-naming-style

Naming style matching correct class constant names.

Default: `UPPER_CASE`

Pylint

--max-attributes

Maximum number of attributes for a class (see R0902).

Default: `7`

Pylint

--max-returns

Maximum number of return / yield for function / method body.

Default: `6`

Pylint

--max-line-length

Maximum number of characters on a single line.

Default: `100`

Pylint

--max-nested-blocks

Maximum number of nested blocks for function / method body

Default: `5`

Go to the example of Pylint.

Go to the code example.

Unit Testing

Unit Testing

- Unit testing is a software testing method by which individual units of source code are tested to determine whether they are fit for use.
- It is a standard step in development and implementation approaches such as Agile.
- Unit would be the smallest component that can be isolated within the complex structure of an app.
- It could be a function, a subroutine, a method or property.
- Unit tests are automated tests or manual tests, written and run by software developers to ensure that a section of an application meets its design and behaves as intended.

Taken from Wikipedia.

Advantages of Unit Testing

- Early detection of problems in the development cycle
- Reduced cost
- Test-driven development
- More frequent releases
- Allows for code refactoring
- Detects changes which may break a design contract
- Reduce uncertainty
- Documentation of system behavior

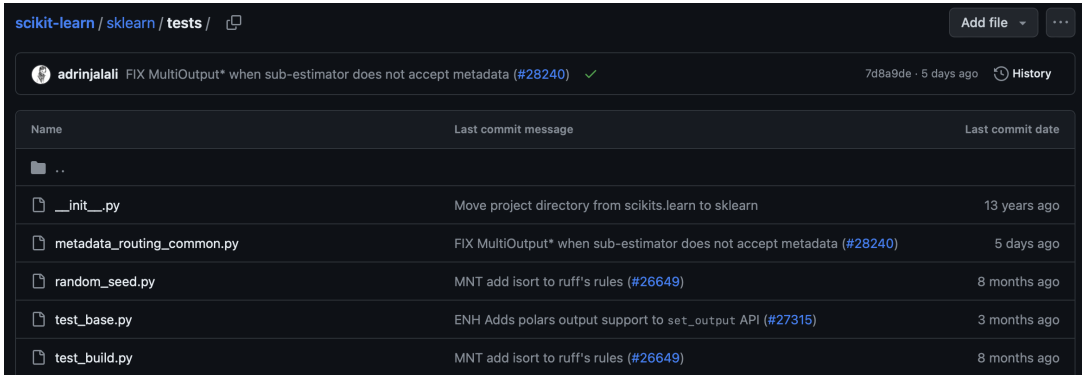
Taken from Wikipedia.

Limitations and Disadvantages of Unit Testing

- Difficulty in setting up realistic and useful tests
- Requires discipline throughout the development process
- Requires version control
- Requires regular reviews
- Limitations for embedded system software
- Limitations for testing integration with external systems

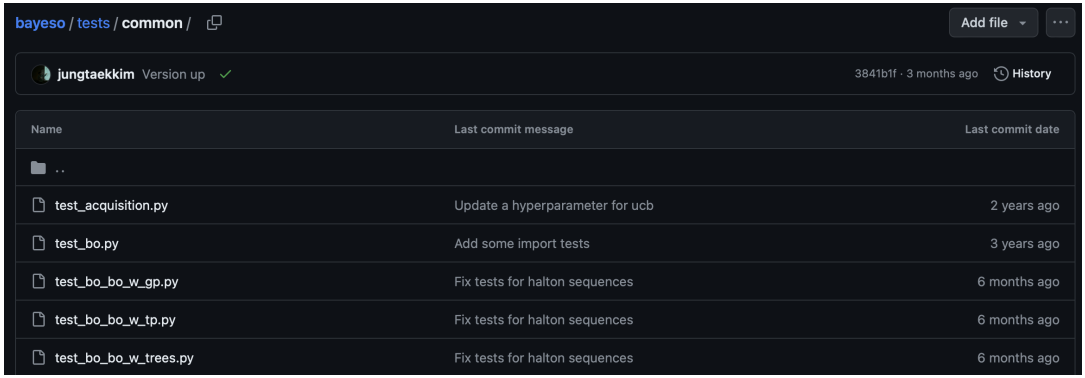
Taken from Wikipedia.

Examples of Unit Testing

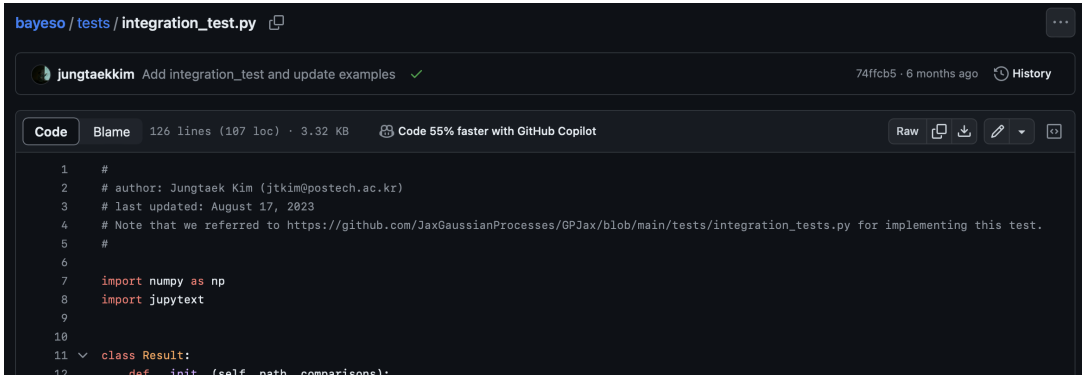


University of Pittsburgh

Examples of Unit Testing



Examples of Unit Testing



Go to the code example.

Continuous Integration

Continuous Integration

- Continuous integration is the practice of merging all developers' working copies to a shared mainline several times a day.
- It is typically implemented in such a way that it triggers an automated build with testing.
- Common practices suggested by various authors:
  - Automate the build,
  - Make the build self-testing,
  - Everyone commits to the baseline every day,
  - Every commit should be built,
  - Every bug-fix commit should come with a test case,
  - Keep the build fast,
  - Test in a clone of the production environment,
  - Make it easy to get the latest deliverables,
  - Everyone can see the results of the latest build,
  - Automate deployment.

Taken from Wikipedia.

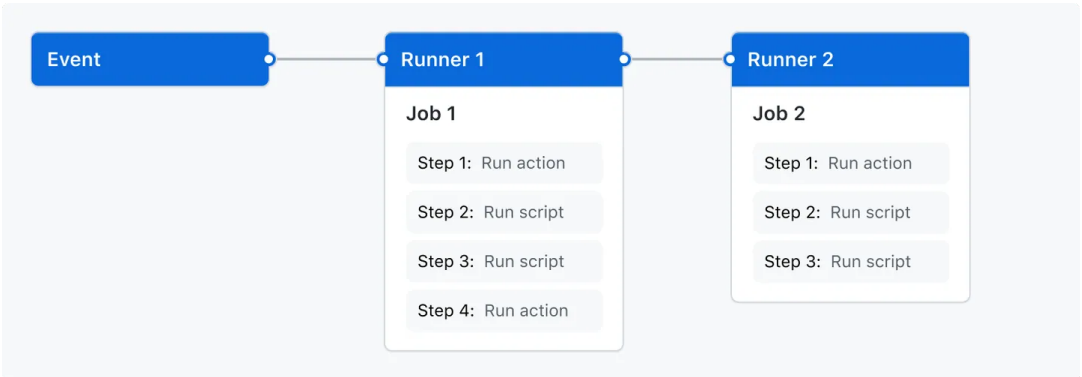
GitHub Actions

GitHub Actions

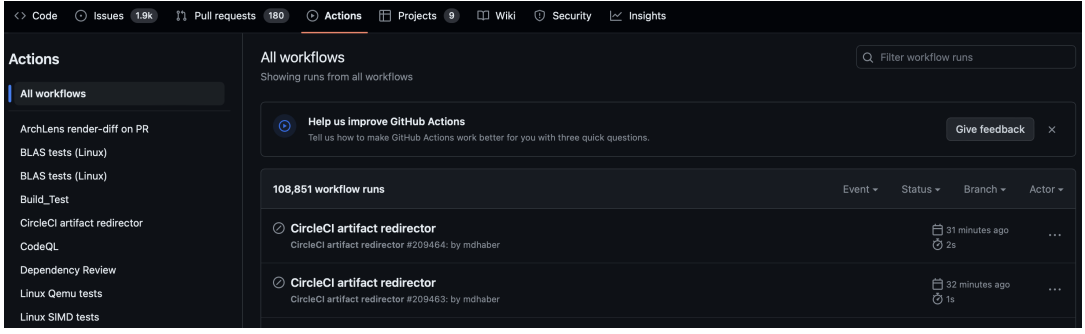
- GitHub Actions is a continuous integration and continuous delivery platform that allows you to automate your build, test, and deployment pipeline.
- Users can create workflows that build and test every pull request to their repository, or deploy merged pull requests to production.
- GitHub Actions goes beyond just DevOps and lets users run their workflows when other events happen in their repository.
- GitHub provides Linux, Windows, and macOS virtual machines to run workflows, or users can host their own self-hosted runners in their own data center or cloud infrastructure.

Taken from the official documentation.

GitHub Actions

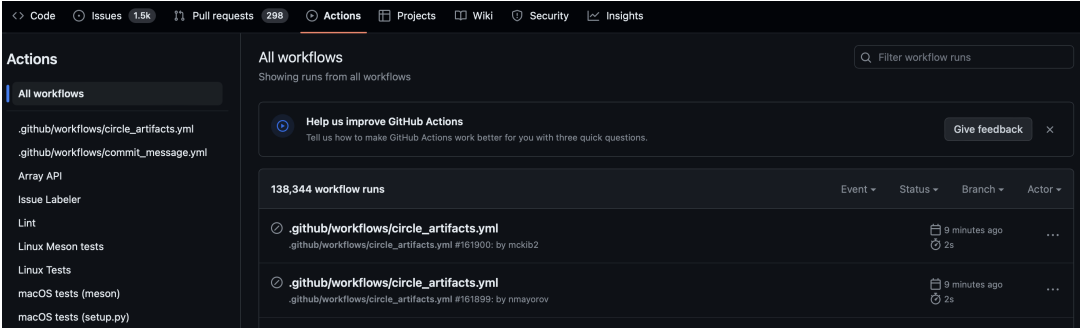


GitHub Actions in NumPy



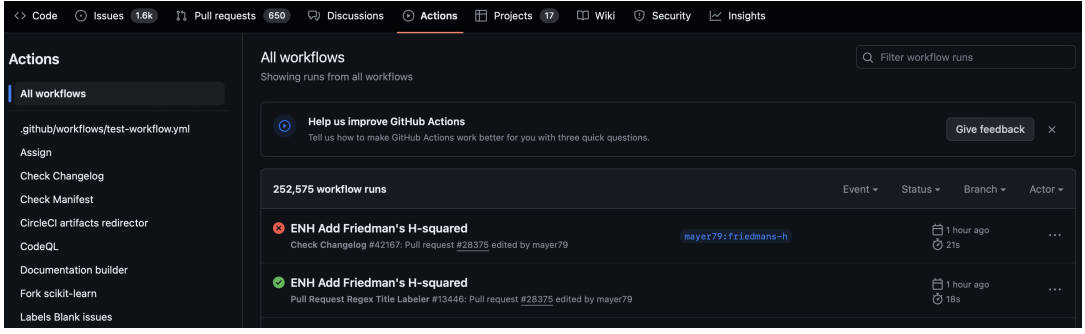
University of Pittsburgh

GitHub Actions in SciPy

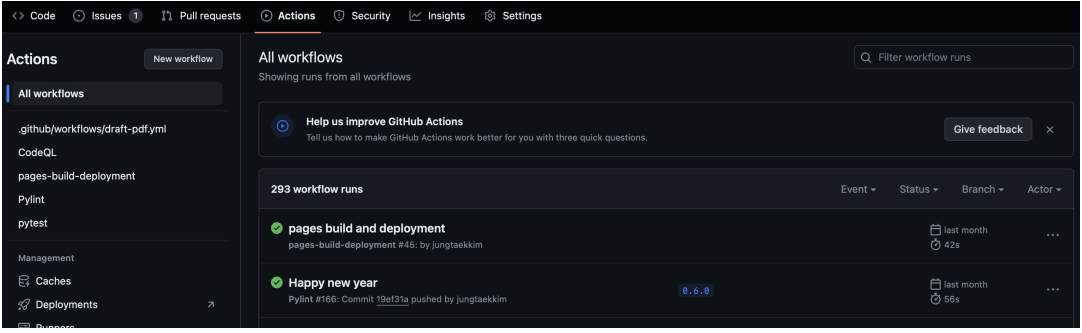


University of Pittsburgh

GitHub Actions in scikit-learn



GitHub Actions in BayesO



Go to the code example.

Documentation

Documentation

- Software documentation is written text or illustration that accompanies computer software or is embedded in the source code.
- The documentation either explains how the software operates or how to use it, and may mean different things to people in different roles.
- Documentation is an important part of software engineering.
- Notably, a docstring is a string literal specified in source code that is used, like a comment, to document a specific segment of code.
- Unlike conventional source code comments, docstrings are not stripped from the source tree when it is parsed and are retained throughout the runtime of the program.

Taken from Wikipedia and Wikipedia.

Docstring

```
"""The module's docstring"""

class MyClass:
    """The class's docstring"""

    def my_method(self):
        """The method's docstring"""

def my_function():
    """The function's docstring"""
```

Examples of Documentation

User Guide API reference Development Release notes Learn

Search the docs ...

NumPy documentation

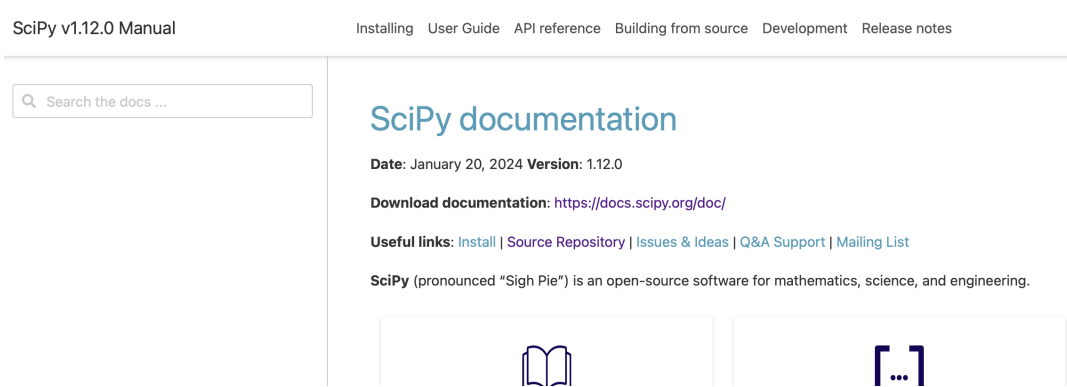
Version: 1.26

Download documentation: Historical versions of documentation

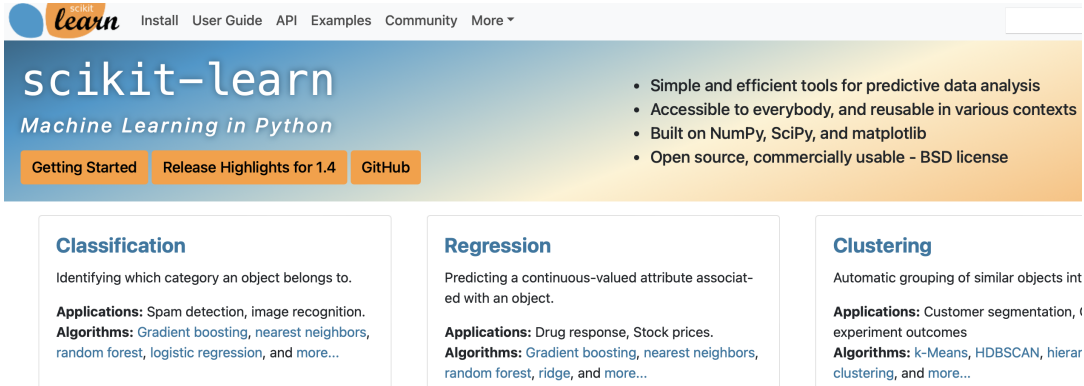
Useful links: Installation | Source Repository | Issue Tracker | Q&A Support | Mailing List

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

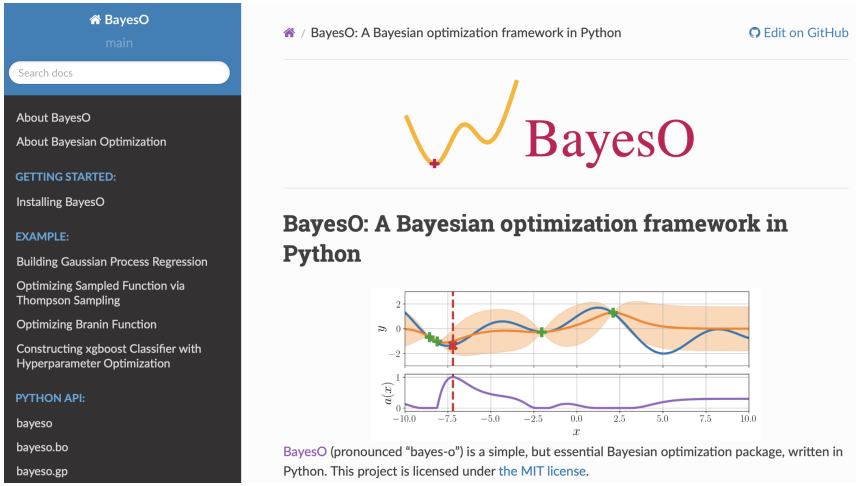
Examples of Documentation



Examples of Documentation



Examples of Documentation



Go to the code example.

Python Package Index

Python Package Index

- The Python Package Index, abbreviated as PyPI, is the official third-party software repository for Python.
- It is analogous to the CPAN repository for Perl and to the CRAN repository for R.
- PyPI is run by the Python Software Foundation, a charity.
- Some package managers, including pip, use PyPI as the default source for packages and their dependencies.
- As of 11 February 2024, more than 500,000 Python packages can be accessed through PyPI.
- It is available at `https://pypi.org`.

Taken from Wikipedia.

Journal of Open Source Software

Journal of Open Source Software

- The Journal of Open Source Software is a peer-reviewed open-access scientific journal covering open-source software from any research discipline.
- The journal is a sponsored project of NumFOCUS and an affiliate of the Open Source Initiative.
- The journal uses GitHub as publishing platform.
- It is available at `https://joss.theoj.org`.
- All reviews and the details of review processes are visible at this link.

Taken from Wikipedia.

University of Pittsburgh

Takeaways

Takeaways

- Do refactoring frequently.
- Do practice in your project.
- Open your project.
- Update your project consistently.
- Contribute to other open-source projects.

Thank you!