Scientific Software Center

IWR Interdisciplinary Center for Scientific Computing

UNIVERSITÄT HEIDELBERG ZUKUNFT SEIT 1386

Open Research Software

Best practices for reliable, maintainable software

Liam Keegan, SSC

Scientific
Software
Center

Scientific Software Center

- Team of Research Software Engineers (currently 6)
- Offer researchers at Heidelberg University
  - Large scale software development
  - Small scale software development
  - Consultation / advice
  - Teaching / training
- Our website / github page also offers
  - Coding guidelines
  - Template repositories

ssc.iwr.uni-heidelberg.de

github.com/ssciwr

Research Software

- Is an increasingly vital part of scientific research
- Is an intrinsic part of reproducible science
- Is not only code written by “real programmers”
    - Your Python data analysis script is also research software!

For people to trust your research, they need to trust your software

- Needs to be **open**
- Needs to be **reliable**
- Needs to be **maintainable**

Best practices for reliable, maintainable software

- Open source development
- Version control
- Testing
- Documentation
- Continuous integration
- Community involvement

For each of these I will

- Describe what it is and what the benefits are
- Make some concrete recommendations
- Show an example of this from an open source library (pybind11)

Scientific Software Center

Open source development

Open source development

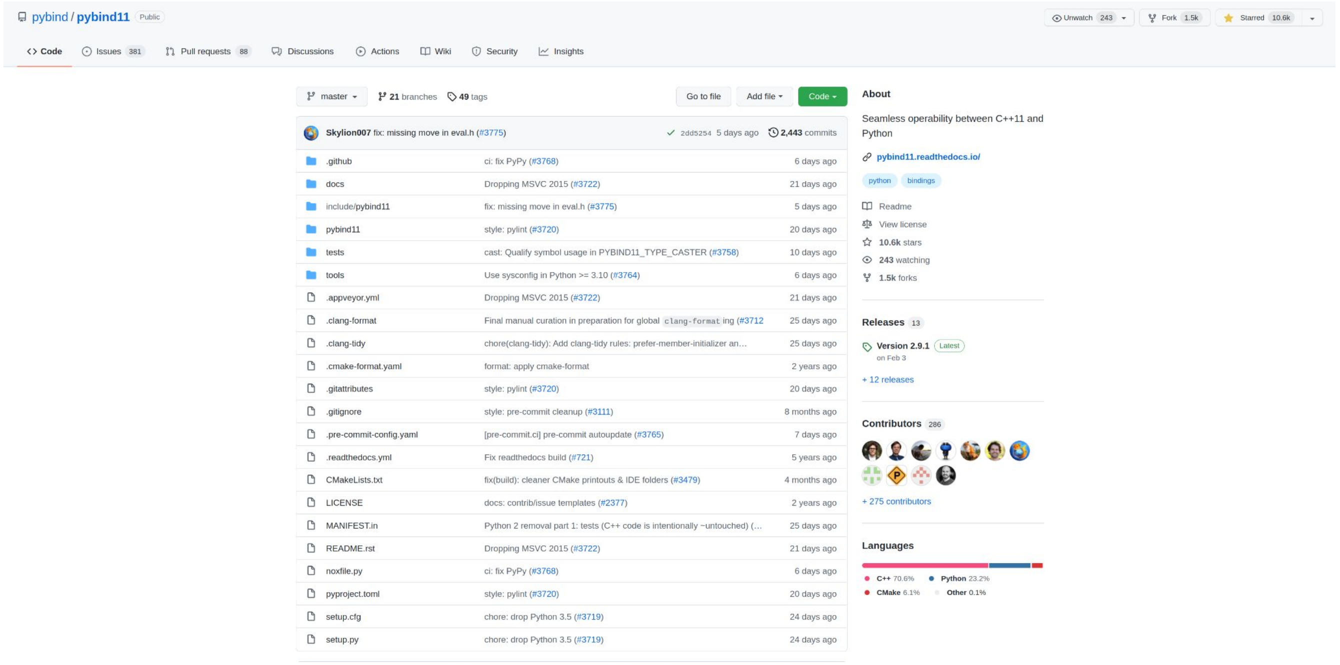
- Making your source code publicly available
    - e.g. GitHub, GitLab, Software Heritage, Zenodo
- Advantages
    - Makes it easier for people to reproduce your results
    - People can find mistakes and bugs
    - People can fix mistakes and bugs
    - People can offer suggestions, improvements
    - People can cite and use your work
    - Gives others confidence in the value of your code

Which platform to use?

- GitHub.com / GitLab.com / etc
  - Commercial git hosting with a (substantial) free tier of services
- Self-hosted gitlab / forgejo / etc
  - Your institute may offer self-hosted gitlab or other code hosting services
- Software Heritage
  - Public software archive, provides a SWHID for your code
- Zenodo
  - Public data archive, provides a DOI for your code
- Recommendation
  - Some form of git hosting + Software Heritage + Zenodo

Scientific Software Center

Open source development example



pybind / **pybind11** Public

Unwatch 243 · Fork 1.5k · Starred 10.6k

<> Code · Issues 381 · Pull requests 88 · Discussions · Actions · Wiki · Security · Insights

master · **21** branches · **49** tags · Go to file · Add file · Code

Skylion007 fix: missing move in eval.h (#3775) ✓ 2dd5254 5 days ago · **2,443** commits

.github	ci: fix PyPy (#3768)	6 days ago
docs	Dropping MSVC 2015 (#3722)	21 days ago
include/pybind11	fix: missing move in eval.h (#3775)	5 days ago
pybind11	style: pylint (#3720)	20 days ago
tests	cast: Qualify symbol usage in PYBIND11_TYPE_CASTER (#3758)	10 days ago
tools	Use sysconfig in Python >= 3.10 (#3764)	6 days ago
.appveyor.yml	Dropping MSVC 2015 (#3722)	21 days ago
.clang-format	Final manual curation in preparation for global `clang-format` ing (#3712	25 days ago
.clang-tidy	chore(clang-tidy): Add clang-tidy rules: prefer-member-initializer an…	25 days ago
.cmake-format.yaml	format: apply cmake-format	2 years ago
.gitattributes	style: pylint (#3720)	20 days ago
.gitignore	style: pre-commit cleanup (#3111)	8 months ago
.pre-commit-config.yaml	[pre-commit.ci] pre-commit autoupdate (#3765)	7 days ago
.readthedocs.yml	Fix readthedocs build (#721)	5 years ago
CMakeLists.txt	fix(build): cleaner CMake printouts & IDE folders (#3479)	4 months ago
LICENSE	docs: contrib/issue templates (#2377)	2 years ago
MANIFEST.in	Python 2 removal part 1: tests (C++ code is intentionally ~untouched) (…	25 days ago
README.rst	Dropping MSVC 2015 (#3722)	21 days ago
noxfile.py	ci: fix PyPy (#3768)	6 days ago
pyproject.toml	style: pylint (#3720)	20 days ago
setup.cfg	chore: drop Python 3.5 (#3719)	24 days ago
setup.py	chore: drop Python 3.5 (#3719)	24 days ago

About

Seamless operability between C++11 and Python

pybind11.readthedocs.io/

python · bindings

Readme
View license
10.6k stars
243 watching
1.5k forks

Releases 13

Version 2.9.1 Latest
on Feb 3

+ 12 releases

Contributors 286

+ 275 contributors

Languages

● **C++** 70.6% ● **Python** 23.2%
● **CMake** 6.1% ● **Other** 0.1%

https://github.com/pybind/pybind11/

Version control

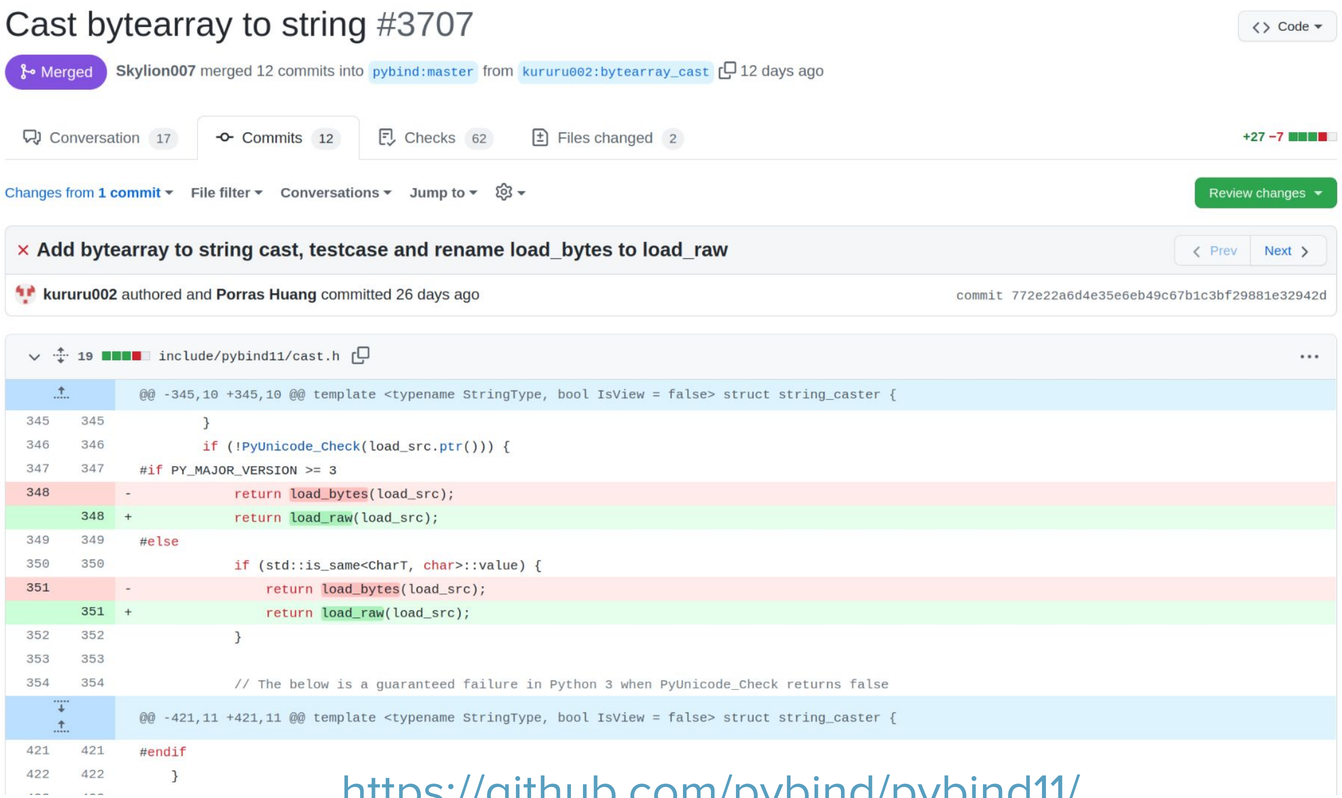
Version control

- Use a tool to track changes to your software
  - e.g. git, subversion, mercurial
- Advantages
  - Easily keep track of changes to the code
  - What changed, who changed it, when and why it changed
  - Easy to refer to specific commit, tag or version for reproducibility
  - Easy to undo or revert changes
  - Easy for multiple people to collaborate on the same code
  - Gives others confidence in the history of your code

Which version control system to use?

- Git
  - The de-facto standard, now used by the vast majority of open source projects
- Workflows
  - There are many ways to use git, known as workflows
  - Centralized workflow, Feature branching, Forking workflow
  - git-flow, gitlab-flow, github-flow, ...
- Recommendation
  - Git with a main branch
  - New code is developed on a new branch and then merged into main

Version control example



Cast bytearray to string #3707

Merged Skylion007 merged 12 commits into `pybind:master` from `kururu002:bytearray_cast` 12 days ago

Conversation 17 | Commits 12 | Checks 62 | Files changed 2 | +27 −7

Changes from 1 commit | File filter | Conversations | Jump to | Review changes

Add bytearray to string cast, testcase and rename load_bytes to load_raw

kururu002 authored and Porras Huang committed 26 days ago — commit 772e22a6d4e35e6eb49c67b1c3bf29881e32942d

19 include/pybind11/cast.h

```
@@ -345,10 +345,10 @@ template <typename StringType, bool IsView = false> struct string_caster {
         }
         if (!PyUnicode_Check(load_src.ptr())) {
#if PY_MAJOR_VERSION >= 3
-            return load_bytes(load_src);
+            return load_raw(load_src);
#else
            if (std::is_same<CharT, char>::value) {
-                return load_bytes(load_src);
+                return load_raw(load_src);
            }

            // The below is a guaranteed failure in Python 3 when PyUnicode_Check returns false
@@ -421,11 +421,11 @@ template <typename StringType, bool IsView = false> struct string_caster {
#endif
    }
```

https://github.com/pybind/pybind11/

Scientific Software Center

Testing

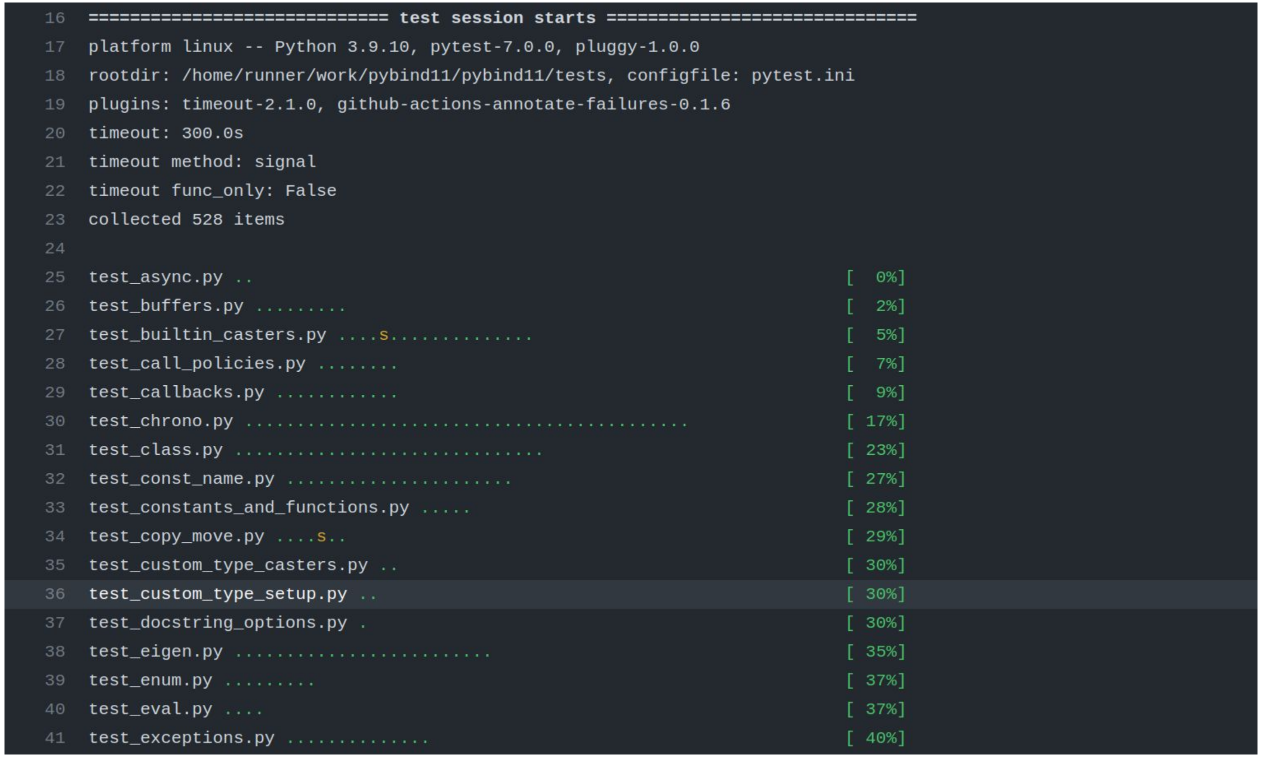
Testing

- Write automated tests that check the software is working correctly
- Advantages
  - Ensure correctness of your code
  - Maintain correctness of your code
  - Find bugs earlier and more easily
  - Make changes or refactor code without fear
  - Easier for new contributors to make positive changes
  - Complement the documentation as examples of use
  - Gives others confidence in the correctness of your code

Types of tests

- Unit tests
  - Test a small, isolated part of code
- Integration / system tests
  - Test larger, connected parts of code
- Regression tests
  - Test for a bug that was fixed to ensure it doesn't come back
- Approval tests
  - Retro-fitting tests before making changes to legacy code
- Recommendation
  - Write unit tests for new projects or new code in legacy projects
  - Write approval tests for legacy code which doesn't have any tests

Testing example



```
============================= test session starts ==============================
platform linux -- Python 3.9.10, pytest-7.0.0, pluggy-1.0.0
rootdir: /home/runner/work/pybind11/pybind11/tests, configfile: pytest.ini
plugins: timeout-2.1.0, github-actions-annotate-failures-0.1.6
timeout: 300.0s
timeout method: signal
timeout func_only: False
collected 528 items

test_async.py ..                                                         [  0%]
test_buffers.py .........                                                [  2%]
test_builtin_casters.py ....s.............                               [  5%]
test_call_policies.py ........                                           [  7%]
test_callbacks.py ............                                           [  9%]
test_chrono.py ............................................              [ 17%]
test_class.py ..............................                             [ 23%]
test_const_name.py ......................                                [ 27%]
test_constants_and_functions.py .....                                    [ 28%]
test_copy_move.py ....s..                                                [ 29%]
test_custom_type_casters.py ..                                           [ 30%]
test_custom_type_setup.py ..                                             [ 30%]
test_docstring_options.py .                                              [ 30%]
test_eigen.py .........................                                  [ 35%]
test_enum.py .........                                                   [ 37%]
test_eval.py ....                                                        [ 37%]
test_exceptions.py ..............                                        [ 40%]
```

https://github.com/pybind/pybind11/

Documentation

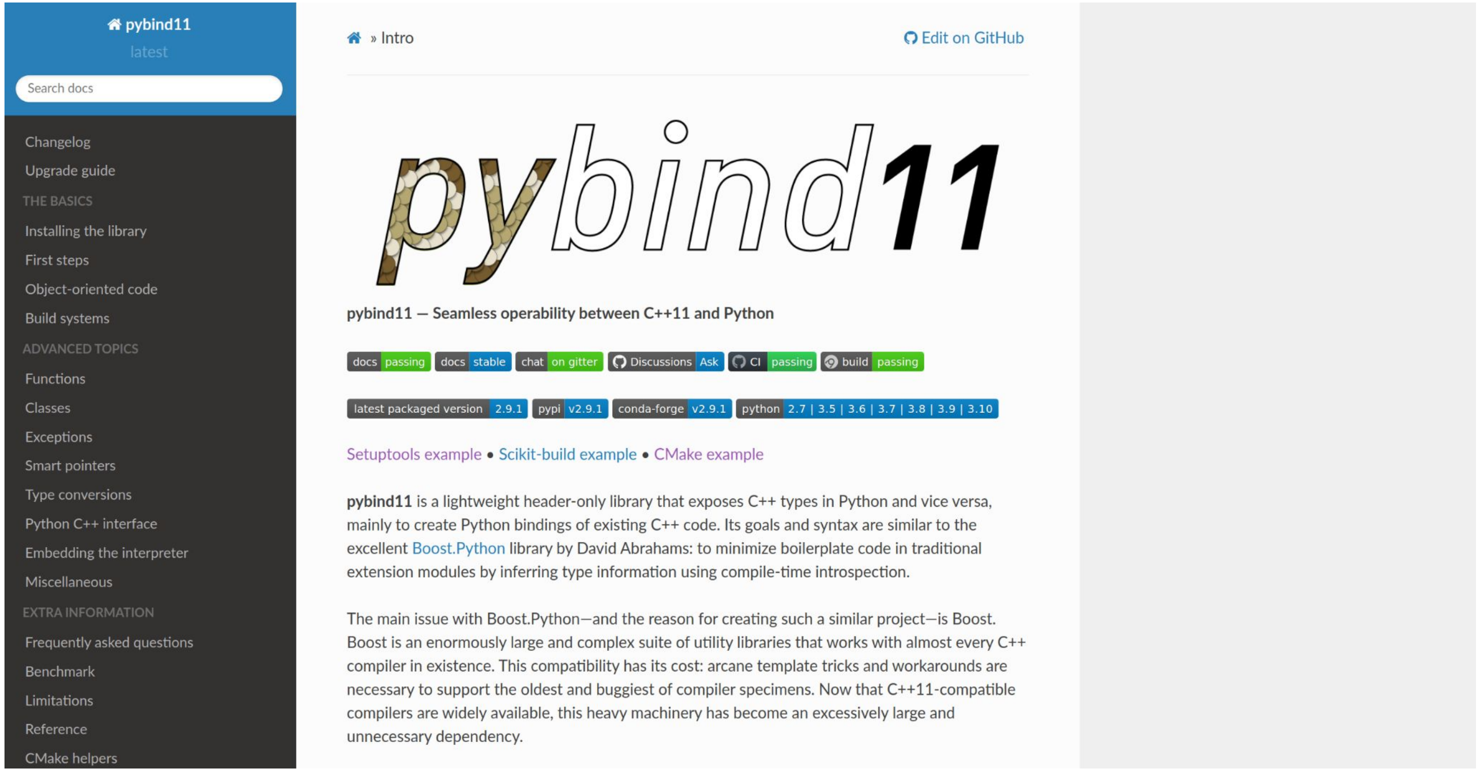
Documentation

- Document how your code works and how to use it
- Advantages
    - Helps users understand how to use the code
    - Helps developers understand how to modify the code
    - Encourages people to learn about your code
    - Gives others confidence in the usability of your code
    - By writing it you can identify hard-to-use code that could be improved

Types of Documentation

- Source code
    - Target audience is other humans, not the computer!
- Comments
    - For you and other developers
- API Documentation
    - Technical documentation for developers / power users
- User documentation
    - Documentation written for users
- Examples
    - Very helpful
- Recommendation
    - Include your documentation in your git repository and update it alongside code changes

Documentation example



pybind11 — Seamless operability between C++11 and Python

Setuptools example • Scikit-build example • CMake example

pybind11 is a lightweight header-only library that exposes C++ types in Python and vice versa, mainly to create Python bindings of existing C++ code. Its goals and syntax are similar to the excellent Boost.Python library by David Abrahams: to minimize boilerplate code in traditional extension modules by inferring type information using compile-time introspection.

The main issue with Boost.Python—and the reason for creating such a similar project—is Boost. Boost is an enormously large and complex suite of utility libraries that works with almost every C++ compiler in existence. This compatibility has its cost: arcane template tricks and workarounds are necessary to support the oldest and buggiest of compiler specimens. Now that C++11-compatible compilers are widely available, this heavy machinery has become an excessively large and unnecessary dependency.

https://github.com/pybind/pybind11/

Scientific Software Center

Continuous integration

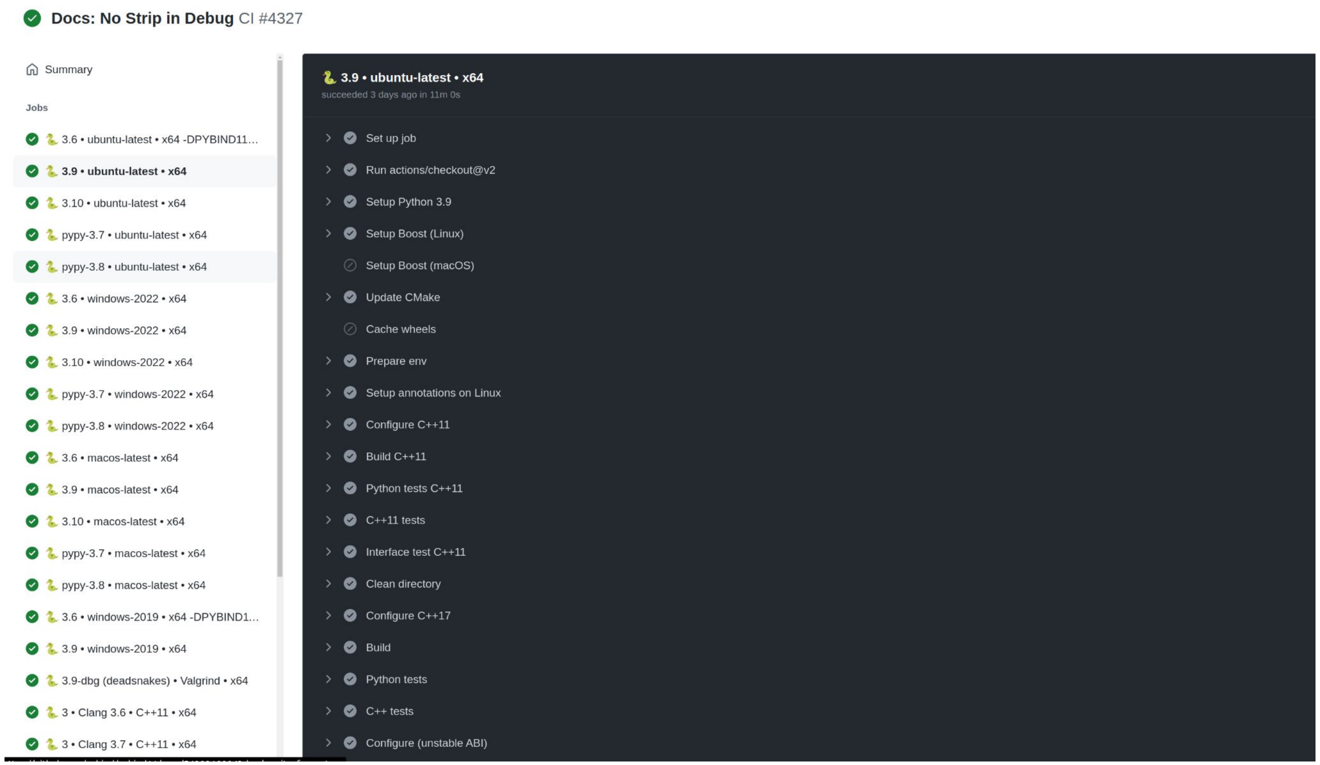
Continuous integration

- Automatic checks before code changes are accepted
- Advantages
  - Ensure all tests pass before code is changed
  - Can automatically apply uniform formatting of the code
  - Can automatically do static analysis to identify code smells or bugs
  - Can require that new code is covered by tests
  - Test the code on multiple platforms (e.g. Windows, Mac, Linux)
  - Can automatically deploy new releases of software
  - Helps others improve the quality of their proposed code changes

Types of continuous integration

- Integrated into git hosting service
  - GitHub Actions, GitLab CI/CD, ...
- External services
  - Travis CI, Circle CI, ...
- Self hosted
  - Jenkins, ...
- Recommendation
  - Typically easiest to use the CI provided by your git hosting service
  - E.g. for code on GitHub use GitHub Actions

Continuous integration example



✓ **Docs: No Strip in Debug** CI #4327

Summary

Jobs

- ✓ 3.6 • ubuntu-latest • x64 -DPYBIND11...
- ✓ **3.9 • ubuntu-latest • x64**
- ✓ 3.10 • ubuntu-latest • x64
- ✓ pypy-3.7 • ubuntu-latest • x64
- ✓ pypy-3.8 • ubuntu-latest • x64
- ✓ 3.6 • windows-2022 • x64
- ✓ 3.9 • windows-2022 • x64
- ✓ 3.10 • windows-2022 • x64
- ✓ pypy-3.7 • windows-2022 • x64
- ✓ pypy-3.8 • windows-2022 • x64
- ✓ 3.6 • macos-latest • x64
- ✓ 3.9 • macos-latest • x64
- ✓ 3.10 • macos-latest • x64
- ✓ pypy-3.7 • macos-latest • x64
- ✓ pypy-3.8 • macos-latest • x64
- ✓ 3.6 • windows-2019 • x64 -DPYBIND1...
- ✓ 3.9 • windows-2019 • x64
- ✓ 3.9-dbg (deadsnakes) • Valgrind • x64
- ✓ 3 • Clang 3.6 • C++11 • x64
- ✓ 3 • Clang 3.7 • C++11 • x64

3.9 • ubuntu-latest • x64
succeeded 3 days ago in 11m 0s

- ✓ Set up job
- ✓ Run actions/checkout@v2
- ✓ Setup Python 3.9
- ✓ Setup Boost (Linux)
- Setup Boost (macOS)
- ✓ Update CMake
- Cache wheels
- ✓ Prepare env
- ✓ Setup annotations on Linux
- ✓ Configure C++11
- ✓ Build C++11
- ✓ Python tests C++11
- ✓ C++11 tests
- ✓ Interface test C++11
- ✓ Clean directory
- ✓ Configure C++17
- ✓ Build
- ✓ Python tests
- ✓ C++ tests
- ✓ Configure (unstable ABI)

https://github.com/pybind/pybind11/

Scientific Software Center

Community involvement

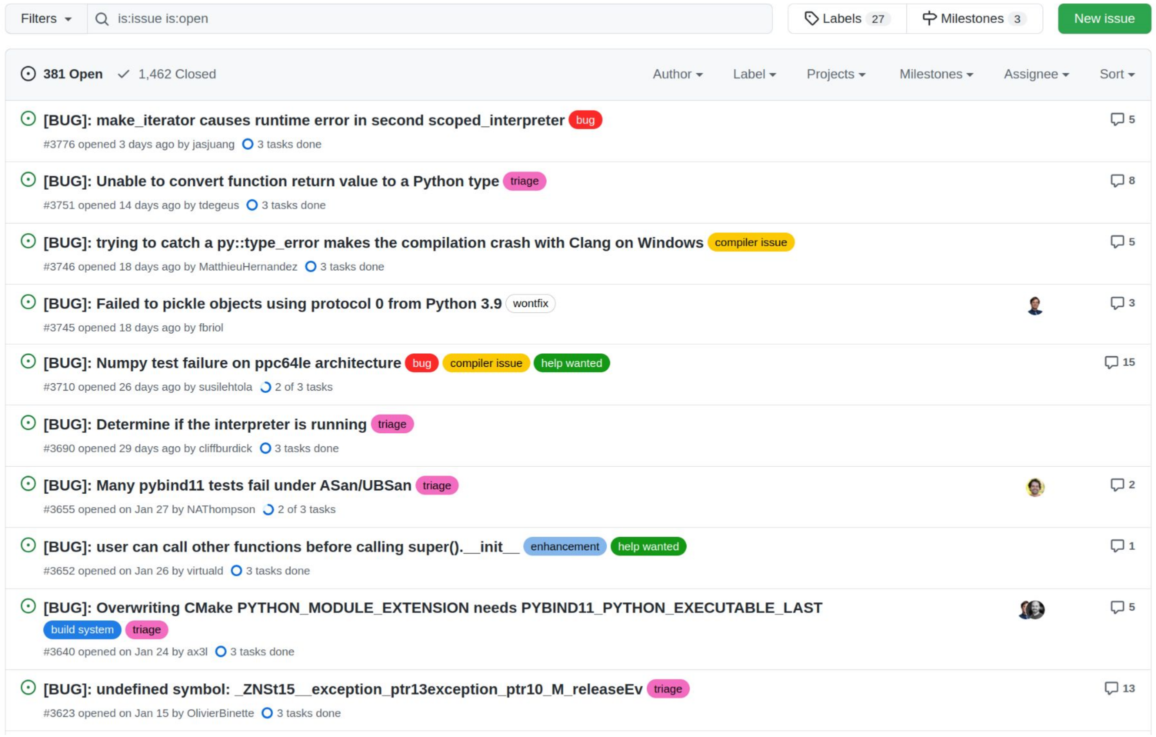
Community involvement

- Enable people to contribute bug reports, feature requests and code
- Advantages
    - People can find mistakes and bugs
    - People can fix mistakes and bugs
    - People can improve the documentation
    - People can offer suggestions, improvements
    - People can help each other to use your code
    - More contributors can make a project more sustainable
    - Helps others to use and contribute to your work

Communication channels

- Issue trackers on git hosting service
- Mailing list
- Contact email for support / questions
- Wiki pages
- Recommendation
    - Use public issue trackers for all feedback / discussions / bugs / features

Community involvement example



Filters is:issue is:open Labels 27 Milestones 3 New issue

381 Open ✓ 1,462 Closed Author Label Projects Milestones Assignee Sort

[BUG]: make_iterator causes runtime error in second scoped_interpreter bug 5
#3776 opened 3 days ago by jasjuang 3 tasks done

[BUG]: Unable to convert function return value to a Python type triage 8
#3751 opened 14 days ago by tdegeus 3 tasks done

[BUG]: trying to catch a py::type_error makes the compilation crash with Clang on Windows compiler issue 5
#3746 opened 18 days ago by MatthieuHernandez 3 tasks done

[BUG]: Failed to pickle objects using protocol 0 from Python 3.9 wontfix 3
#3745 opened 18 days ago by fbriol

[BUG]: Numpy test failure on ppc64le architecture bug compiler issue help wanted 15
#3710 opened 26 days ago by susilehtola 2 of 3 tasks

[BUG]: Determine if the interpreter is running triage
#3690 opened 29 days ago by cliffburdick 3 tasks done

[BUG]: Many pybind11 tests fail under ASan/UBSan triage 2
#3655 opened on Jan 27 by NAThompson 2 of 3 tasks

[BUG]: user can call other functions before calling super().__init__ enhancement help wanted 1
#3652 opened on Jan 26 by virtuald 3 tasks done

[BUG]: Overwriting CMake PYTHON_MODULE_EXTENSION needs PYBIND11_PYTHON_EXECUTABLE_LAST build system triage 5
#3640 opened on Jan 24 by ax3l 3 tasks done

[BUG]: undefined symbol: _ZNSt15__exception_ptr13exception_ptr10_M_releaseEv triage 13
#3623 opened on Jan 15 by OlivierBinette 3 tasks done

https://github.com/pybind/pybind11/

Summary

Scientific Software Center

Best practices for reliable, maintainable software

- Open source development
- Version control
- Testing
- Documentation
- Continuous integration
- Community involvement

Getting started

- Start from a template repository
- Basic project ready to go
    - Open source development
    - Version control
    - Testing
    - Documentation
    - Continuous integration
    - Community involvement

Basic C++ Project Template

github.com/ssciwr/cpp-project-template

Basic Python Project Template

github.com/ssciwr/python-project-template

Advanced C++ Project Template

github.com/ssciwr/cookiecutter-cpp-project