



Scientific
Software
Center



UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Open Research Software

Best practices and tools for reliable, maintainable software

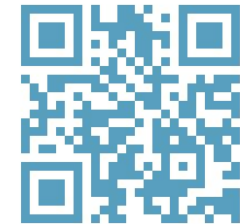
Liam Keegan, SSC

Scientific Software Center

- Team of Research Software Engineers (currently 6)
- Offer researchers at Heidelberg University
 - Software development as a service
 - Consultation and advice
 - Teaching and training
- Our website / github page also offers
 - Coding guidelines
 - Template repositories
 - Slides and code samples from our courses



ssc.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)

Tools for writing reliable, maintainable software

- Integrated Development Environments (e.g. vscode, pycharm)
- AI code completion (e.g. copilot, intellicode)
- Generative AI (e.g. chatgpt, claude, gemini)

For each of these I will

- Describe what it is and what the benefits are
- Make some concrete recommendations
- Show an example of use

Open source development

Open source development

- Making your source code publicly available
 - e.g. GitHub, GitLab, Software Heritage, Zenodo
- With a suitable open source license
- 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

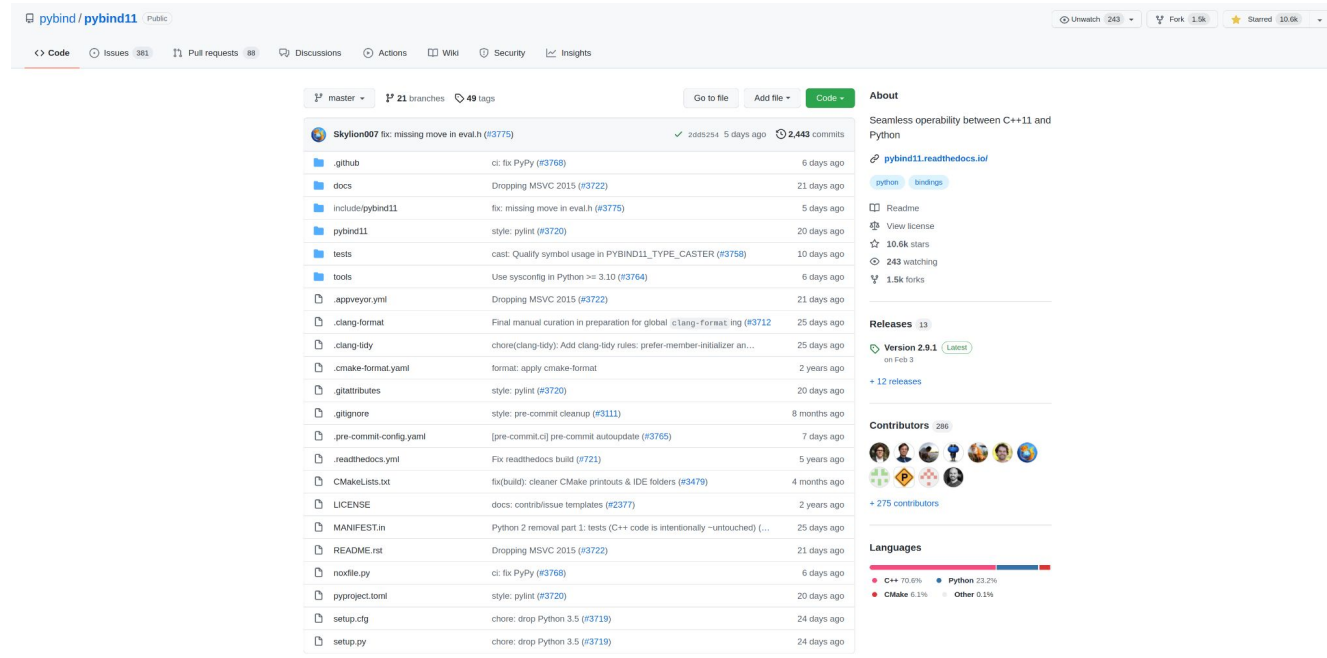
Which platform to use?

- 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

Open source development example



The screenshot shows the GitHub repository page for `pybind/pybind11`. The repository is public and has 243 unwatched users, 1.5k forks, and 10.6k stars. The main content is a file browser showing the repository structure and a list of recent commits.

Repository Structure:

- `.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: Quality 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
- `.gignore`: 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.ctg`: chore: drop Python 3.5 (#3719) - 24 days ago
- `setup.py`: chore: drop Python 3.5 (#3719) - 24 days ago

Right-hand Side (About, Releases, Contributors, Languages):

- About:** Seamless operability between C++11 and Python. Link to `pybind11.readthedocs.io/`. Includes Readme, View license, 10.6k stars, 243 watching, and 1.5k forks.
- Releases:** 13 releases. Latest: **Version 2.9.1** (Latest) on Feb 3. + 12 releases.
- Contributors:** 286 contributors. + 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

[Code](#)

Merged Skylon007 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`

Prev Next

kururu002 authored and Porras Huang committed 26 days ago

commit 772e22a6d4e35e6eb49c67b1c3bf29881e32942d

```

include/pybind11/cast.h
@@ -345,10 +345,10 @@ template <typename StringType, bool IsView = false> struct string_caster {
345 345     }
346 346     if (!PyUnicode_Check(load_src.ptr())) {
347 347     #if PY_MAJOR_VERSION >= 3
348 -         return load_bytes(load_src);
348 +         return load_raw(load_src);
349 349     #else
350 350     if (std::is_same<charT, char>::value) {
351 -         return load_bytes(load_src);
351 +         return load_raw(load_src);
352 352     }
353 353
354 354     // 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 {
421 421     #endif
422 422     }
---
```

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

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

```
16 ===== test session starts =====
17 platform linux -- Python 3.9.10, pytest-7.0.0, pluggy-1.0.0
18 rootdir: /home/runner/work/pybind11/pybind11/tests, configfile: pytest.ini
19 plugins: timeout-2.1.0, github-actions-annotate-failures-0.1.6
20 timeout: 300.0s
21 timeout method: signal
22 timeout func_only: False
23 collected 528 items
24
25 test_async.py .. [ 0%]
26 test_buffers.py ..... [ 2%]
27 test_builtin_casters.py .....S..... [ 5%]
28 test_call_policies.py ..... [ 7%]
29 test_callbacks.py ..... [ 9%]
30 test_chrono.py ..... [ 17%]
31 test_class.py ..... [ 23%]
32 test_const_name.py ..... [ 27%]
33 test_constants_and_functions.py ..... [ 28%]
34 test_copy_move.py .....S.. [ 29%]
35 test_custom_type_casters.py .. [ 30%]
36 test_custom_type_setup.py .. [ 30%]
37 test_docstring_options.py . [ 30%]
38 test_eigen.py ..... [ 35%]
39 test_enum.py ..... [ 37%]
40 test_eval.py .... [ 37%]
41 test_exceptions.py ..... [ 40%]
```

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



The screenshot shows the pybind11 documentation website. On the left is a dark sidebar with a blue header containing the pybind11 logo and 'latest' text. Below the header is a search bar and a list of navigation links including 'Changelog', 'Upgrade guide', 'THE BASICS', 'Installing the library', 'First steps', 'Object-oriented code', 'Build systems', 'ADVANCED TOPICS', 'Functions', 'Classes', 'Exceptions', 'Smart pointers', 'Type conversions', 'Python C++ interface', 'Embedding the interpreter', 'Miscellaneous', 'EXTRA INFORMATION', 'Frequently asked questions', 'Benchmark', 'Limitations', 'Reference', and 'CMake helpers'. The main content area has a light blue header with 'Intro' and 'Edit on GitHub' links. The title 'pybind11' is displayed in a large, stylized font. Below the title is the tagline 'pybind11 – Seamless operability between C++11 and Python'. A row of status badges includes 'docs passing', 'docs stable', 'chat on gitter', 'Discussions Ask', 'CI passing', and 'build passing'. A second row of badges shows 'latest packaged version 2.9.1', 'pypi v2.9.1', 'conda-forge v2.9.1', and 'python 2.7 | 3.5 | 3.6 | 3.7 | 3.8 | 3.9 | 3.10'. Below these are links for 'Setuptools example', 'Scikit-build example', and 'CMake example'. The main text describes pybind11 as a lightweight header-only library for exposing C++ types in Python and vice versa, similar to Boost.Python. It mentions the main issue with Boost.Python and the reason for creating a similar project, highlighting the complexity and size of the Boost.Python machinery.

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

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 -DPYBIND11...
- ✔ 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/>

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
- Public chat room / discussion board
- Mailing list
- Contact email for support / questions
- Wiki pages

Recommendation:

- Use public issue trackers for all feedback / discussions / bugs / features

Community involvement example

Filters ▾ Q 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 3
 #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 av3l 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/issues>

Community involvement example

Q is:open Sort by: Latest activity Label Filter: Open

Categories

- 🗨️ View all discussions
- 📢 Announcements
- ⋮ General
- 💡 Ideas
- 📊 Polls
- 👤 Q&A
- 👉 Show and tell

Most helpful Last 30 days

- 👤 naquad ✓ 1
- 👤 Svalorzen ✓ 1

[Community guidelines](#)

[pybind11.readthedocs.io](#)

Discussions

- ↑ 1 ⋮ **pybind11 add function args sequentially**
Nodmgatal started 18 hours ago in [General](#) 👤 🗨️ 0
- ↑ 1 👤 **classes exported from different modules, built with different versions of pybind11 are not interoperable (anymore)?**
rhaschke asked 2 days ago in [Q&A](#) · Unanswered 👤 🗨️ 0
- ↑ 1 👤 **Dependency shared library not loaded during importing submodule**
kerim371 asked 3 days ago in [Q&A](#) · Unanswered 👤 🗨️ 0
- ↑ 1 👤 **numpy array of arrays?**
cary-ilm asked 4 days ago in [Q&A](#) · Unanswered 👤 🗨️ 0
- ↑ 8 👤 **Removing old Python support**
henryiii started 2 weeks ago in [Polls](#) 👤 🗨️ 7
- ↑ 1 👤 **Overhead of method access**
juanjosegarciaripoll asked 3 weeks ago in [Q&A](#) · Unanswered 👤 🗨️ 2
- ↑ 1 👤 **how to bind C style array of struct ?**
Lovtifa asked on Jul 26, 2023 in [Q&A](#) · Unanswered 👤 🗨️ 1
- ↑ 2 👤 **Interoperability between a pybind11_type and its C++ equivalent**
GabrieljMS asked last month in [Q&A](#) · Unanswered 👤 🗨️ 0
- ↑ 1 👤 **std::stringstream unexpected behavior when use pybind11 module in jupyter cell which the module was statically linked with stdc++ (linux only)**
buchengsyh asked on Apr 23 in [Q&A](#) · Unanswered 👤 🗨️ 0
- ↑ 2 ⋮ **Imported target "pybind11::headers" includes non-existent path**
hakonhagland started on Apr 4 in [General](#) 👤 🗨️ 2

<https://github.com/pybind/pybind11/discussions>

Integrated Development Environment

Integrated Development Environment (IDE)

The IDE is where you type your code, and can provide

- Syntax highlighting
- Code formatting
- Code completion
- Code refactoring
- Compiling, running and debugging
- Warnings about errors and issues
- Suggested improvements based on best practices

Which IDE to use?

- JetBrains (PyCharm / CLion / etc)
 - Commercial, but students can get a free license
 - There is also a free community edition of PyCharm for Python coding
- Visual Studio Code
 - Free, open-source, widely used
- Many others available
 - Eclipse, Visual Studio (commercial), SublimeText, notepad++, vim

Recommendation:

- VS Code or JetBrains

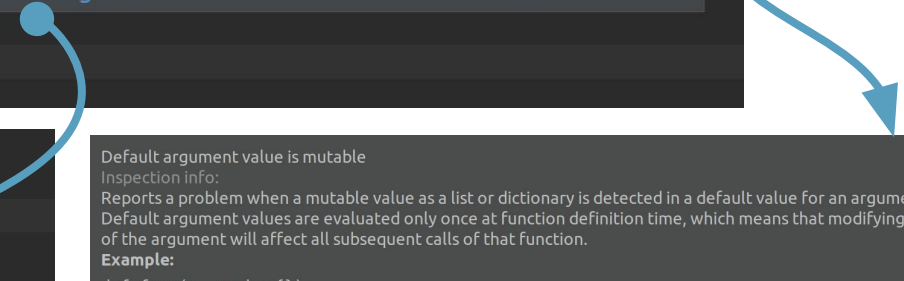
IDE example (Python code in PyCharm)

```

1  import numpy as np
2
3  class MyClass:
4      def __init__(self, x=[1.0, 0.5]):
5          self.x = x
6          self.rotation = np.cos(x)
7          self.translation = np.sin(x)
8          self.result = self.rotation + self.translation
9
  
```


Default argument value is mutable

Replace mutable default argument Alt+Shift+Enter More actions... Alt+Enter



```

3  class MyClass:
4      def __init__(self, x=None):
5          if x is None:
6              x = [1.0, 0.5]
7          self.x = x
8          self.rotation = np.cos(x)
9          self.translation = np.sin(x)
10         self.result = self.rotation + self.translation
  
```



Default argument value is mutable

Inspection info:

Reports a problem when a mutable value as a list or dictionary is detected in a default value for an argument. Default argument values are evaluated only once at function definition time, which means that modifying the default value of the argument will affect all subsequent calls of that function.

Example:

```
def func(s, cache={}):
    cache[s] = None
```

When the quick-fix is applied, the code changes to:

```
def func(s, cache=None):
    if cache is None:
        cache = {}
    cache[s] = None
```

AI code completion

AI code completion

- Autocomplete (on steroids) for your IDE
- As you type it makes pretty good suggestions with surprisingly little input
- You can start writing a function and it will suggest the rest
- It can also generate documentation for a function
- It can write tests for a function
- It can also simplify or explain code
- Works especially well with commonly used or “boilerplate” code

Which AI code completion to use?

- IntelliCode
 - Free, included in VS Code
- GitHub Copilot
 - Commercial
 - But students can get a free GitHub PRO account which includes copilot
- Many others available
 - But the two above are probably the most widely used

Recommendation:

- IntelliCode or Copilot

AI code completion examples (copilot/pycharm)

I started writing a function by typing “def rotate_vector”:

```
def rotate_vector(vector, angle):  
    """Rotate a vector by a given angle in radians."""  
    x, y = vector  
    cos_angle = math.cos(angle)  
    sin_angle = math.sin(angle)  
    return (x * cos_angle - y * sin_angle, x * sin_angle + y * cos_angle)
```

AI code completion examples (copilot/pycharm)

Then edited my text to “def rotate_3d_vector”:

```
def rotate_3d_vector(v, angle, axis):  
    """Rotate a 3D vector by a given angle around a given axis."""  
    # Normalize the axis vector  
    axis = axis / np.linalg.norm(axis)  
    # Compute the rotation matrix  
    R = rotation_matrix(angle, axis)  
    # Rotate the vector  
    return np.dot(R, v)
```

Generative AI

Generative AI

- You can simply ask a chatbot to write your code for you
- The results are not perfect - but often pretty good!
- You can also iterate, asking for changes and improvements
- The kind of code you get depends on the kind of prompt you use
 - E.g. “Please write high quality code that follows best practices in software development.”
- Can be a good way to brainstorm alternative solutions
- Also very helpful when you are not an expert with the language you are using
- Your role here changes from writing code to reviewing code

Possible issues with generative AI

- Copyright issues
 - Was the training data copyrighted? Is your data copyrighted?
- Spaghetti code
 - Easy to quickly generate a lot of not-great code: makes your life harder later
- Subtly wrong code
 - They sometimes “hallucinate”, which is a polite word for lying convincingly to you

Recommendation:

- Use with care!

Generative AI examples



You

write a python function that rotates a 3d vector



ChatGPT

To write a Python function that rotates a 3D vector, we need to use some linear algebra. Specifically, we'll use rotation matrices. A common way to represent 3D rotations is using Euler angles or quaternions, but for simplicity, we'll use rotation matrices for rotations around the principal axes (x, y, and z).

Here's the code to rotate a 3D vector using rotation matrices:

```
python Copy code
import numpy as np

def rotate_vector(vector, angle, axis):
    """
    Rotate a 3D vector around a given axis by a specified angle.

    Parameters:
    vector (list or np.ndarray): The 3D vector to rotate.
    angle (float): The rotation angle in radians.
    axis (str): The axis to rotate around ('x', 'y', or 'z').

    Returns:
    np.ndarray: The rotated 3D vector.
    """
```

```
if axis not in ['x', 'y', 'z']:
    raise ValueError("Axis must be 'x', 'y' or 'z'")

if axis == 'x':
    rotation_matrix = np.array([
        [1, 0, 0],
        [0, np.cos(angle), -np.sin(angle)],
        [0, np.sin(angle), np.cos(angle)]
    ])
elif axis == 'y':
    rotation_matrix = np.array([
        [np.cos(angle), 0, np.sin(angle)],
        [0, 1, 0],
        [-np.sin(angle), 0, np.cos(angle)]
    ])
elif axis == 'z':
    rotation_matrix = np.array([
        [np.cos(angle), -np.sin(angle), 0],
        [np.sin(angle), np.cos(angle), 0],
        [0, 0, 1]
    ])

rotated_vector = np.dot(rotation_matrix, vector)
return rotated_vector
```

Generative AI examples



You

You are an expert python developer. Write a python function that rotates a 3d vector



ChatGPT

Certainly! Below is a Python function that rotates a 3D vector around a specified axis by a given angle. The function uses the Rodrigues' rotation formula, which is a common method to rotate vectors in 3D space.

```
python Copy code

import numpy as np

def rotate_vector(vector, axis, angle):
    """
    Rotate a 3D vector around a given axis by a specified angle.

    Parameters:
    vector (numpy.array): The vector to be rotated (3D).
    axis (numpy.array): The axis to rotate around (3D).
    angle (float): The rotation angle in radians.

    Returns:
    numpy.array: The rotated vector.
    """
```

```
# Normalize the axis vector
axis = axis / np.linalg.norm(axis)

# Compute the cross-product matrix of the axis
K = np.array([
    [0, -axis[2], axis[1]],
    [axis[2], 0, -axis[0]],
    [-axis[1], axis[0], 0]
])

# Compute the rotation matrix using Rodrigues' rotation formula
I = np.eye(3)
R = I + np.sin(angle) * K + (1 - np.cos(angle)) * np.dot(K, K)

# Rotate the vector
rotated_vector = np.dot(R, vector)

return rotated_vector
```

Summary

For reliable, maintainable software

Follow best practices:

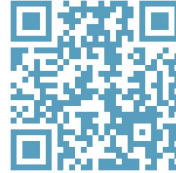
- Open source development
- Version control
- Testing
- Documentation
- Continuous integration
- Community involvement

Use good tools:

- Integrated Development Environment
- AI-powered autocomplete
- Generative AI (with care)

Getting started

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



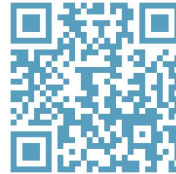
Basic C++ Project Template

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