



ENTERPRISE TECHNOLOGY MANAGEMENT

A Tale of two Kitchens

Hypermodernizing your Codebase

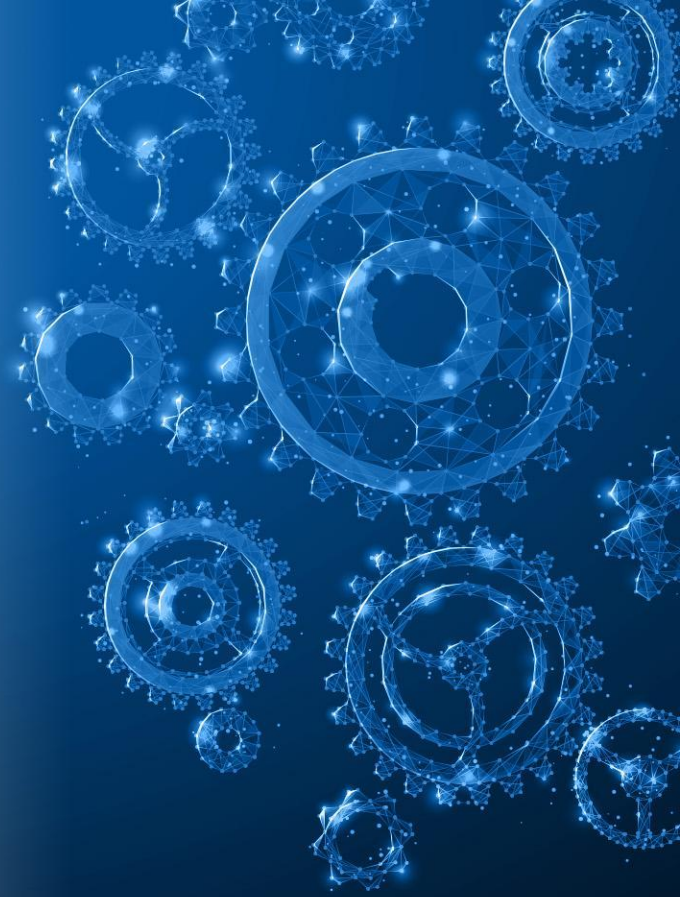
Christian Ledermann

EuroPython 14/07/2022

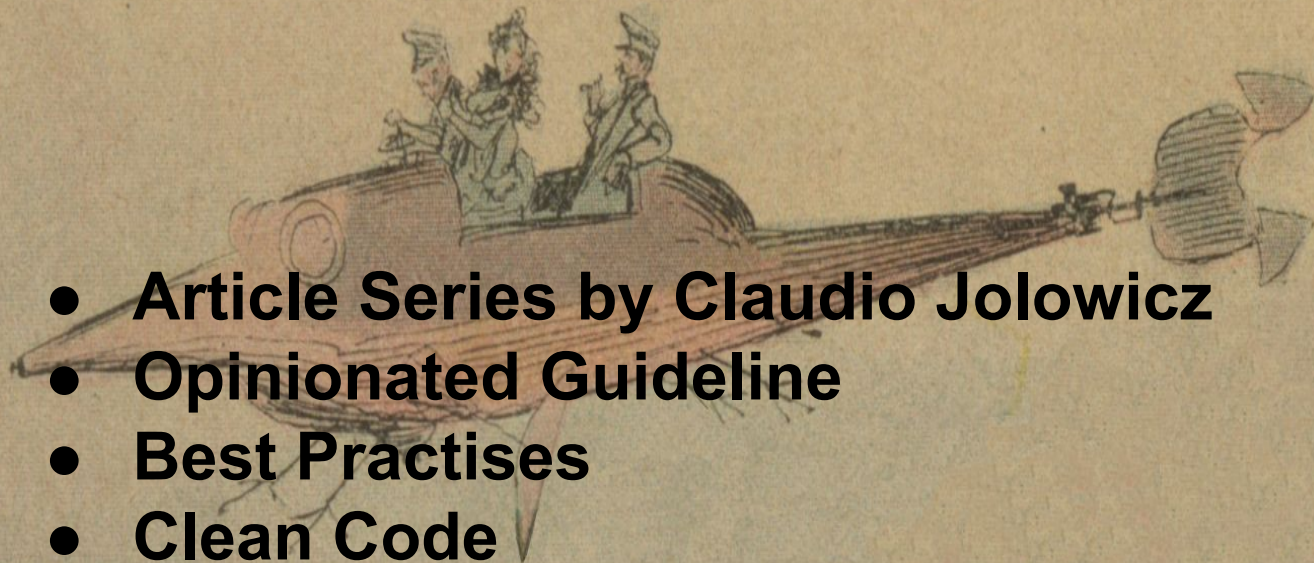


Overview

- What is Hypermodern Python
- Why Hypermodern Python
- How to hypermodernize your legacy Code



Hypermodern Python



- **Article Series by Claudio Jolowicz**
- **Opinionated Guideline**
- **Best Practises**
- **Clean Code**

Hypermodern Python

- Packaging and dependency management with [Poetry](#)
- Test automation with [Nox](#)
- Linting with [pre-commit](#) and [Flake8](#)
- Continuous integration with [GitHub Actions](#)
- Documentation with [Sphinx](#), [MyST](#), and [Read the Docs](#) using the [furo](#) theme
- Automated uploads to [PyPI](#) and [TestPyPI](#)
- Automated release notes with [Release Drafter](#)
- Automated dependency updates with [Dependabot](#)
- Code formatting with [Black](#) and [Prettier](#)
- Import sorting with [isort](#)
- Testing with [pytest](#)
- Code coverage with [Coverage.py](#)
- Coverage reporting with [Codecov](#)
- Command-line interface with [Click](#)
- Static type-checking with [mypy](#)
- Runtime type-checking with [Typeguard](#)
- Automated Python syntax upgrades with [pyupgrade](#)
- Security audit with [Bandit](#) and [Safety](#)
- Check documentation examples with [xdoctest](#)
- Generate API documentation with [autodoc](#) and [napoleon](#)
- Generate command-line reference with [sphinx-click](#)
- Manage project labels with [GitHub Labeler](#)

A Tale of Two Kitchens



© Steven Depolo CC BY 2.0



© Jack Monroe (@BootstrapCook)

Which Kitchen?

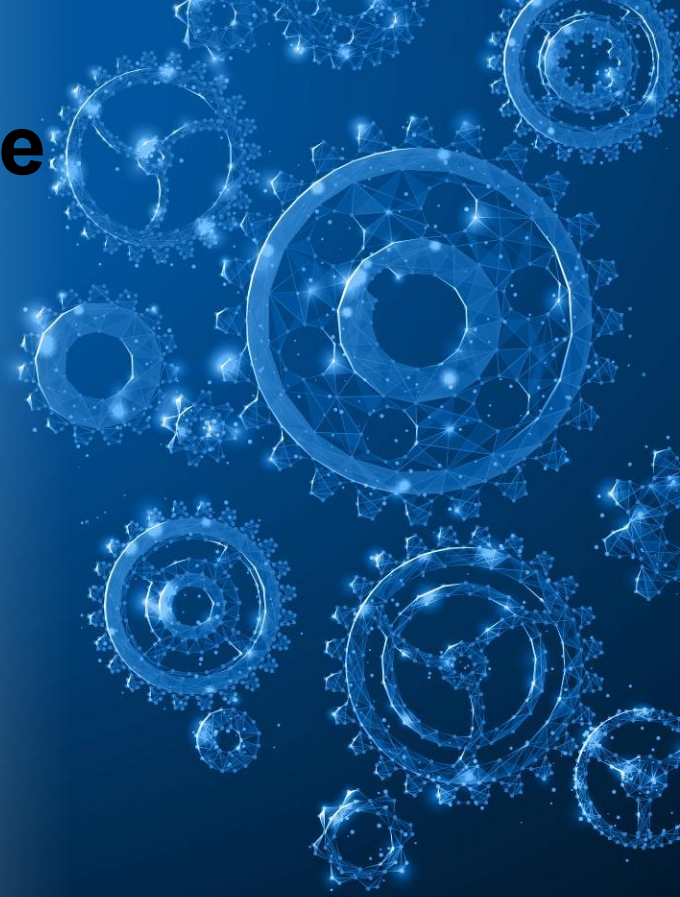
- Security
- Health and Safety
- Deliver Fast
- Deliver High Quality
- Job Satisfaction
- Professional Growth

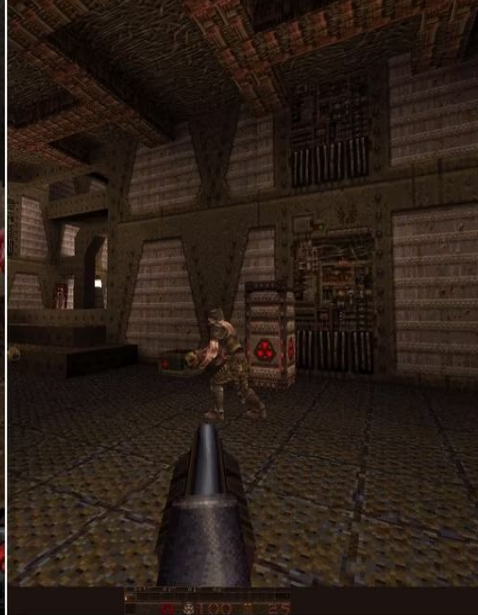




Why Clean up your Codebase

- Minimize Context switches
- LEAN: Eliminate Waste
- Broken Windows
- Boy Scout Rule
- Improve quality





The most important thing I have done as a programmer in recent years is to aggressively pursue static code analysis. Even more valuable than the hundreds of serious bugs I have prevented with it is the change in mindset about the way I view software reliability and code quality.

-- John Carmack

A woman wearing a purple t-shirt, blue jeans, and a purple face mask is working at a brick stove. She is holding a metal lid over a pot. In the foreground, there is a wooden tray containing several glass jars and some gold-colored coins. The background is a weathered, light blue wall with visible cracks.

Start Small Start Simple Start Now



Getting Started - Format Matters

- pre-commit
- isort, absolufy-imports, unimport, remove-star
- black
- ruff
- flake8 + Plugins



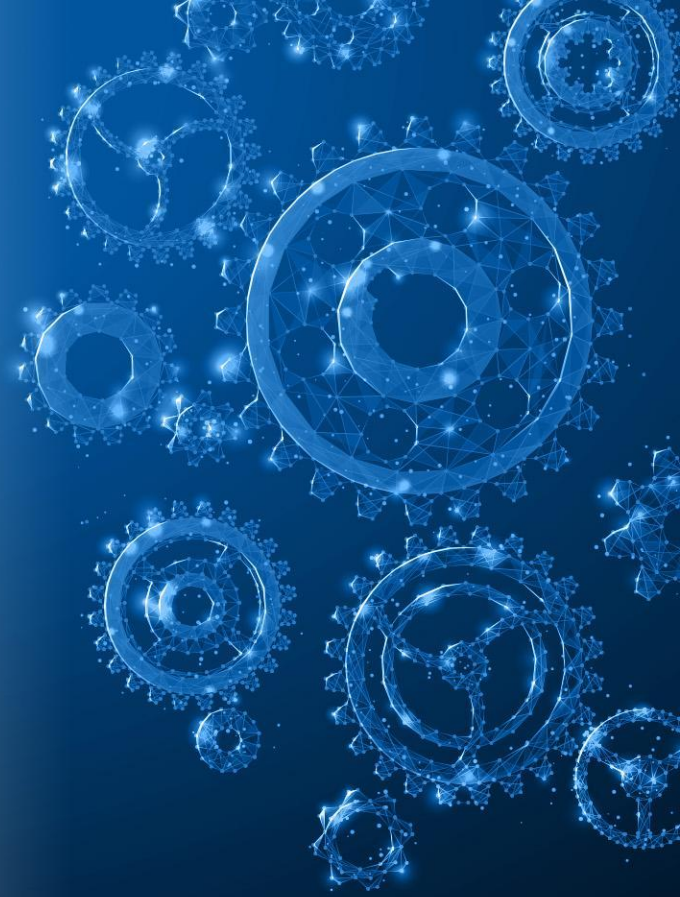
Security

- BugBear
- Bandit
- Safety
- Dependabot
- Pysa



You improve what you measure

- Tests
 - Coverage
 - diff-cover
- Complexity
 - McCabe, Radon, Xenon
 - Lizard
 - Cognitive Complexity



Typing

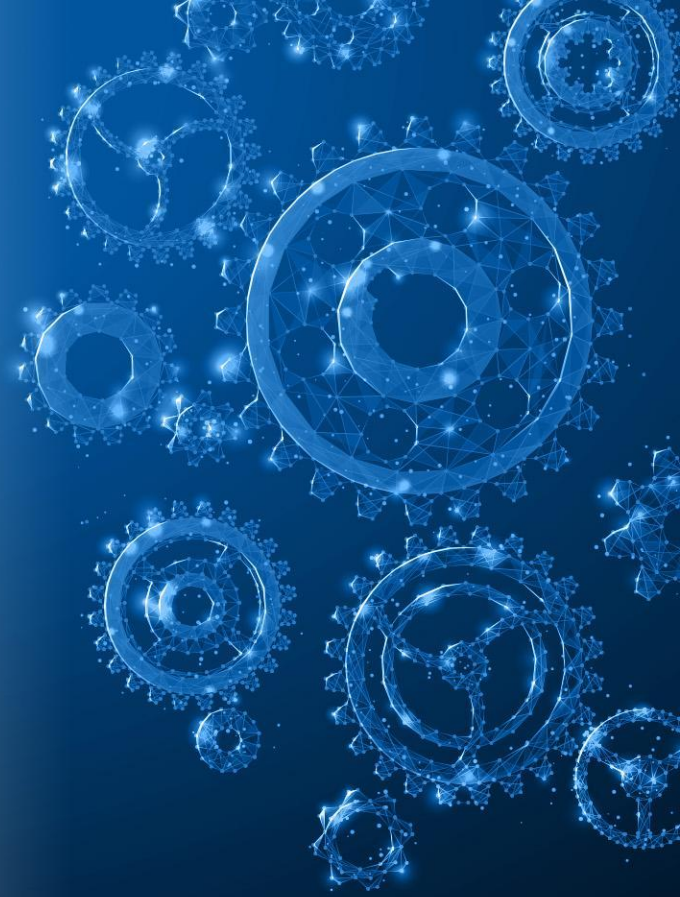
- MyPy
- MonkeyType
- pyre infer, pytype, pyright
- typeguard (for tests)

Tests need Love Too

- Teyit for unit tests
- pytestify
- unittest2pytest

Testing (continued)

- Hypothesis and Ghostwriter
- Schemathesis for web APIs



Testing (continued)

Quis custodiet ipsos custodes?

Who is watching those watchmen?

- MutMut Mutation Testing

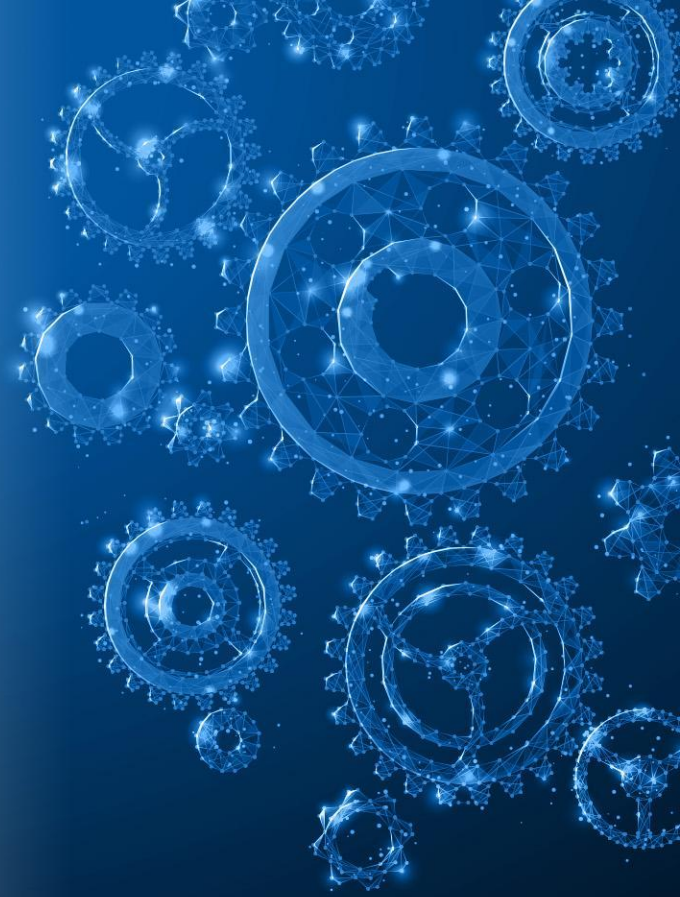
Upgrade

- Pyupgrade
- flynt
- Django-upgrade
- Django-codemod



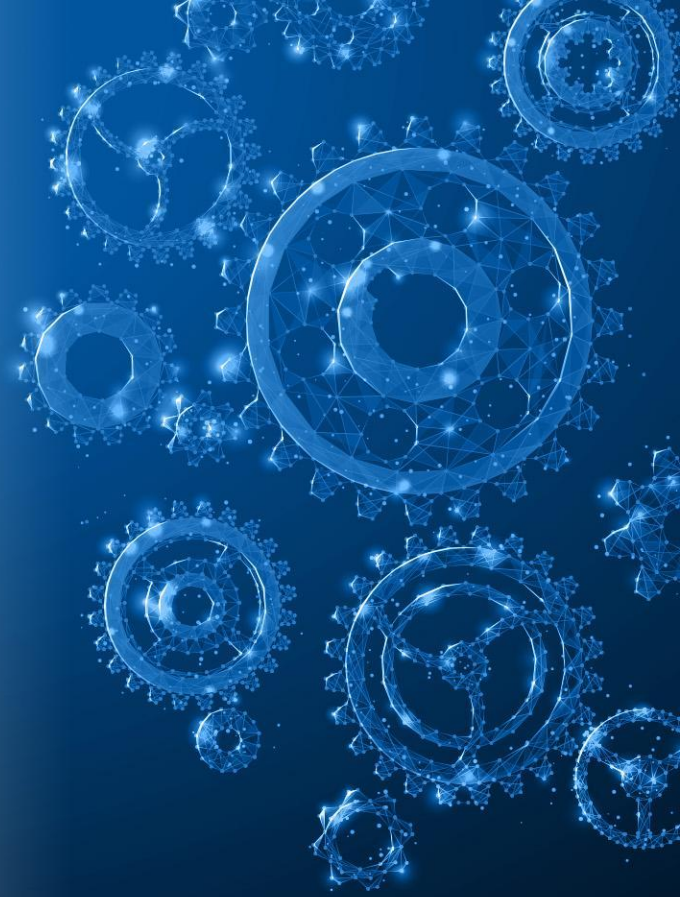
Roll our Own

- libCST
- ...



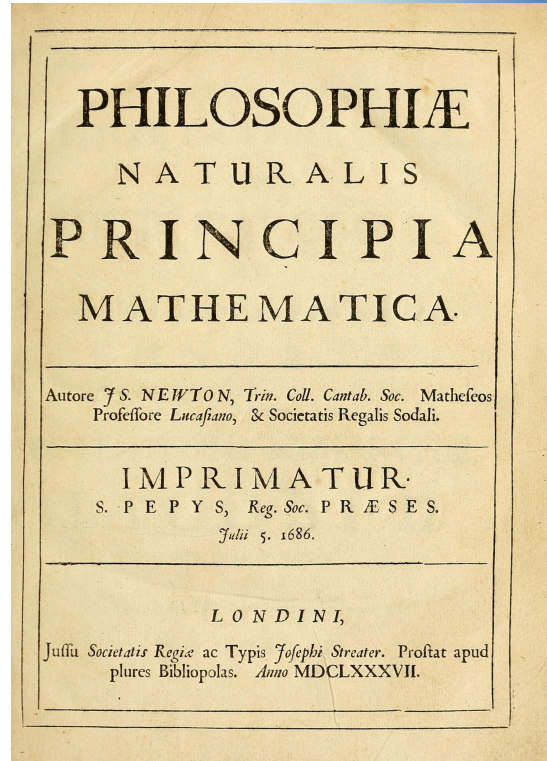
As A Service

- Souncery
- SonarCloud
- Metabob
- Coverage.io



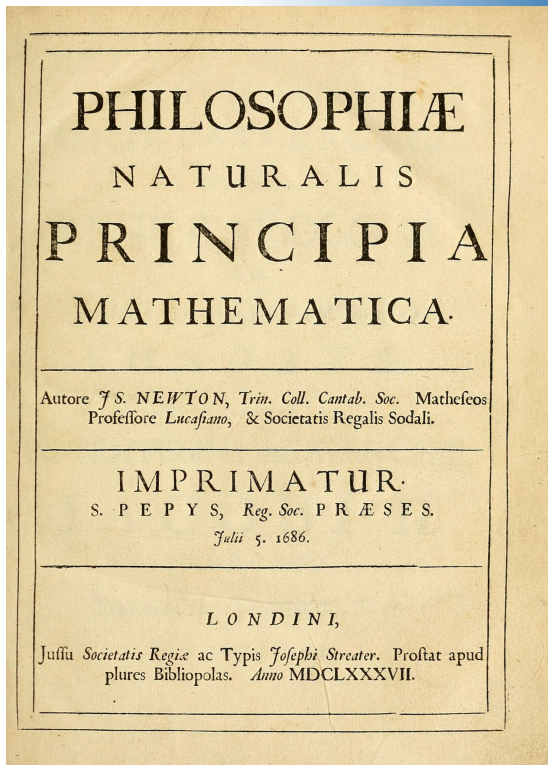
Where To Go From Here?

- DRY
- WET
- SOLID
- CUPID



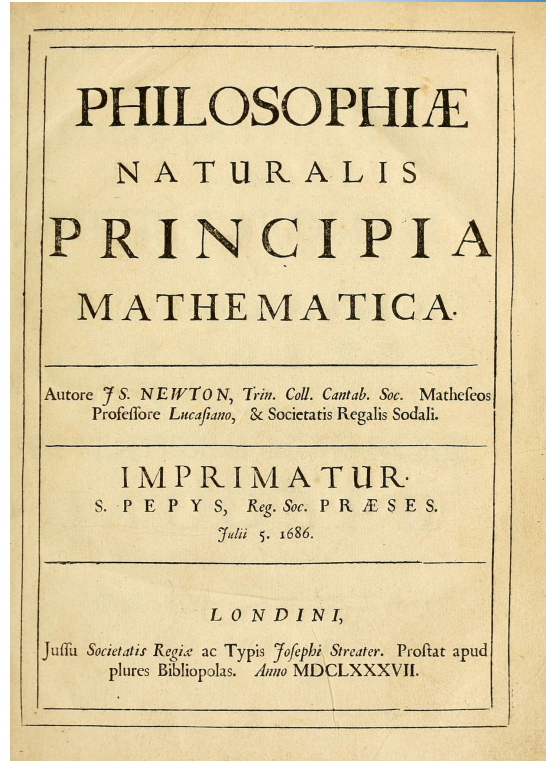
SOLID

- Single Responsibility
- Open-Closed
- Liskov Substitution
- Interface Segregation
- Dependency Inversion



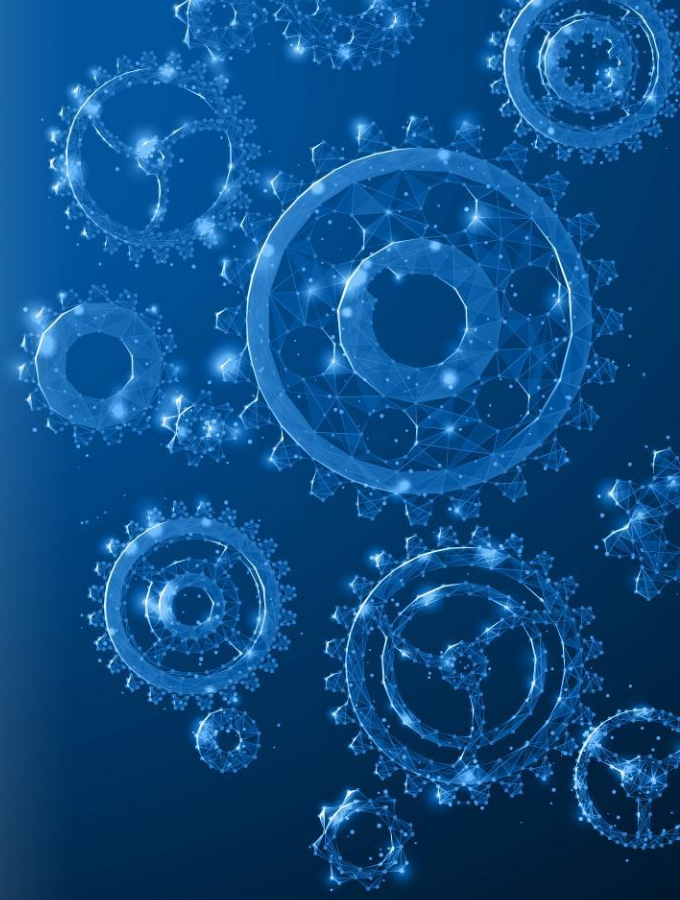
CUPID

- **Composable**: plays well with others
- **Unix philosophy**: does one thing well
- **Predictable**: does what you expect
- **Idiomatic**: feels natural
- **Domain-based**: the solution domain models the problem domain in language and structure

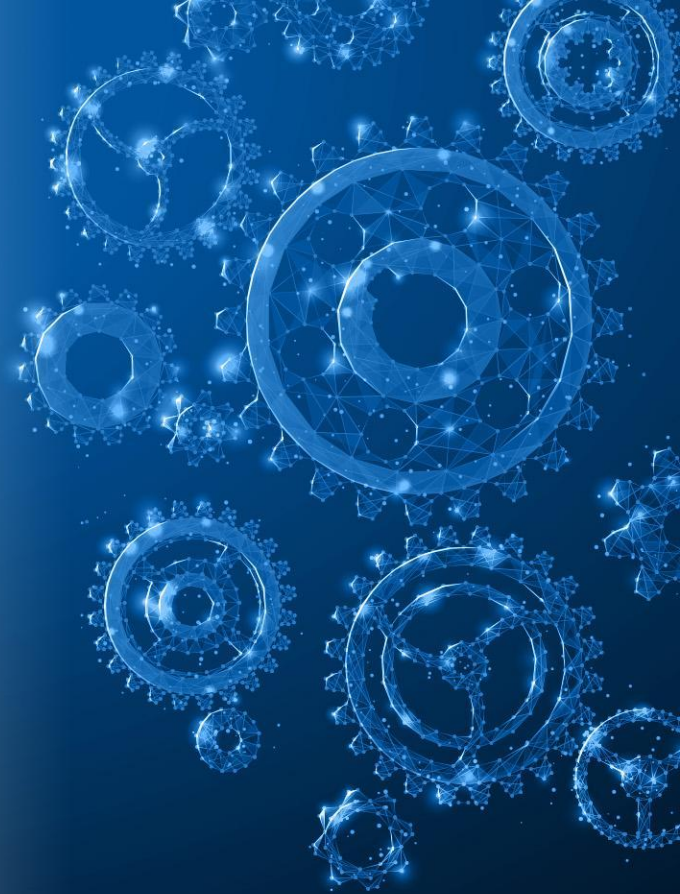


Means to Achieve an End

- Maintainability
- Extensibility
- Modularity



Perfect is the Opposite of Done



Summary

- What is Hypermodern Python
- Why Hypermodern Python
- How to hypermodernize your Legacy Code
- Next steps

Done is Better Than Perfect



THANK YOU

github.com/cleder/ep2022

