Working Legacy Code with modern Python tooling

codecentric

Index

Meeting Reality

- Aim of Code
- Good Code
- Legacy Code

Connascence and PAIN

- Metrics
- Decision helpers

Tools

- Focus on value, business
- Automate boring stuff

What is the aim of working code?

Working software over comprehensive documentation

Responding to change over following a plan

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software

What is NOT the aim of working code?

Having the most sophisticated technology

“PEP 8 everyone”

Show everybody that you are really smart

Why work legacy code

People leave

Things change

Wanted: New features

What is legacy code?

Good code

- Passes the tests
- Reveals intention
- No duplication
- Fewest element

```
# Hey baby, givin' it your all...
def oink_oink_oink_IlIlll (ribbit_ααααα)->int :
   oinks =0
   # Bada bing, bada boom
   for ribbit in ribbit ααααα :
       oinks +=ribbit # there's nothing like Miami's heat
   return oinks
```


https://blog.pragmaticengineer.com/bad-code/

What is legacy code?

Good code

- Passes the tests
- Reveals intention
- No duplication
- Fewest elements

```
# Hey baby, givin' it your all...
def oink_oink_oink_IlIlll (ribbit_ααααα)->int :
    oinks =0
    # Bada bing, bada boom
    for ribbit in ribbit ααααα :
        oinks +=ribbit # there's nothing like Miami's heat
    return oinks

def sum_of(numbers: List[int]) -> int:
    return sum(numbers)
```

What is legacy code?

Good code

- Passes the tests
- Reveals intention
- No duplication
- Fewest elements

```
class Modulator(str, Enum):
    LINEAR = "linear"
    SQUARE = "square"
    CUBE = "cube"

    def __call__(self, v: float):
        if self is self.LINEAR:
            return v
        elif self is self.SQUARE:
            return math.sqrt(v)
        elif self is self.CUBE:
            return math.pow(v, 1.0 / 3)
        return v

def score(modulator: Modulator):
    if not modulator:
        modulator = Modulator.LINEAR

    # ...
    scores = 0.0
    stats = calc_stats(...)
    for v in stats:
        scores += modulator(v / stats["total"])
    # ...
```

What is legacy code?

Good code

- Passes the tests
- Reveals intention
- No duplication
- Fewest elements

```
def score() -> float:
    # ...
    sum of(statistics(...))
    # ...
```

What is legacy code?

Good code

- Passes the tests
- Reveals intention
- No duplication
- Fewest elements

Legacy code

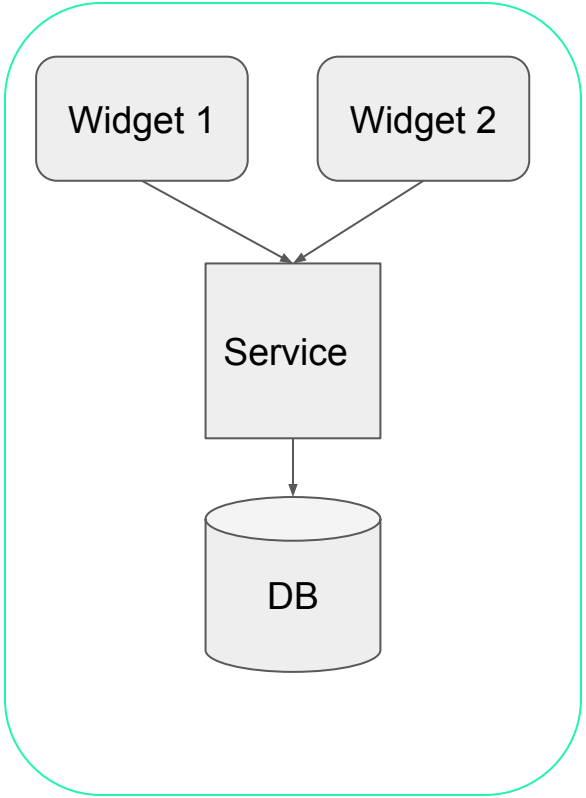
- Untested, hides intention
- Legacy != bad code

Culture

- Legacy as indicator

https://blog.pragmaticengineer.com/bad-code/

Our example



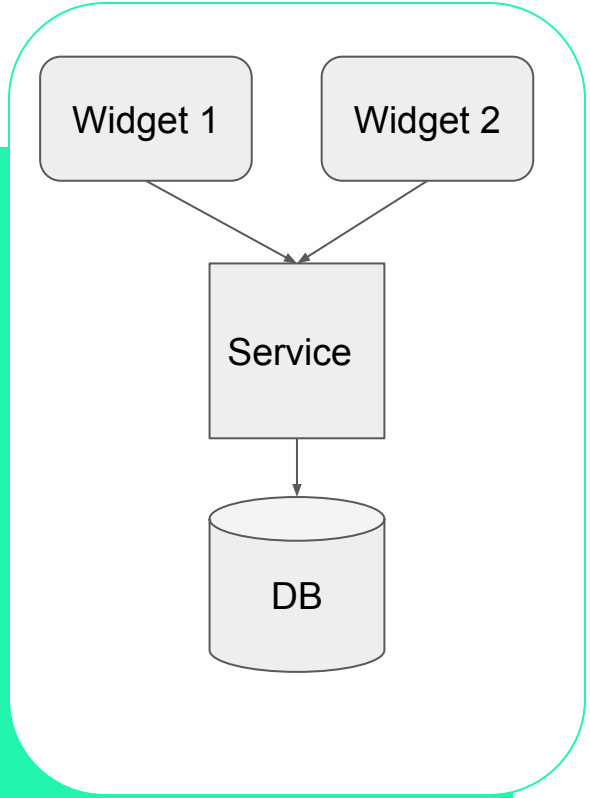
Our example

Open source

Six repositories
0 tests
0 documentation
manual deployment

New Features Wanted

Dev's refused



What to do?

PEP 8 it!

Recreate it from scratch in XZY

Find metrics

- Methods
  - SOLID
  - Coupling
- Tools

Connascence

Element A and B are connascent, if there is a **change in A, that requires a change in B**

- generalization of coupling and cohesion
- many degrees of different severity

Connascence

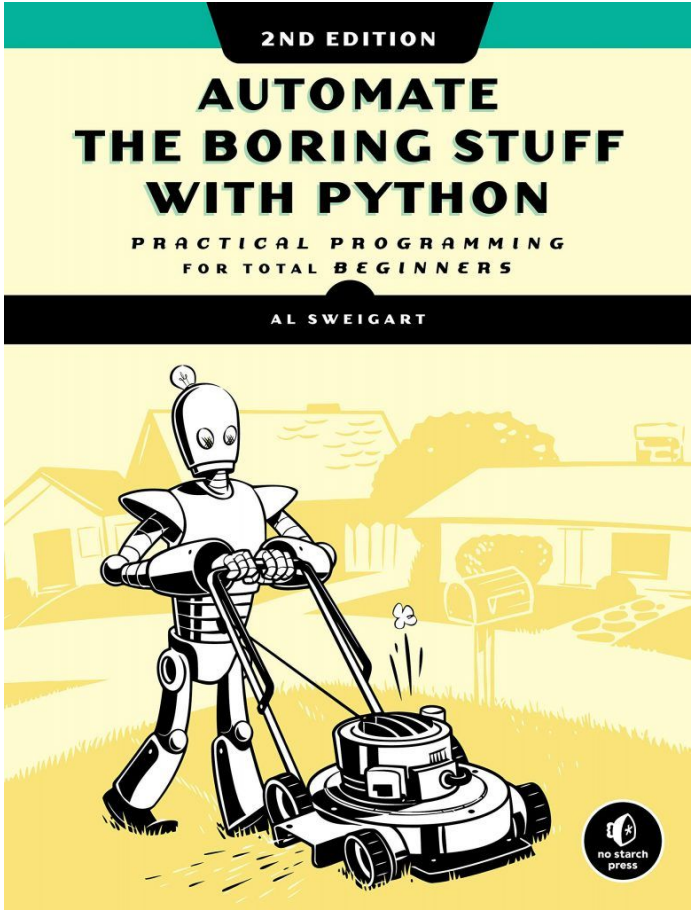
Element A and B are connascent, if there is a **change in A, that requires a change in B**

- Three Rules:
  - **Strength**
  - **Distance**
  - **Degree**

- **PAIN**: Strength x Distance x Degree

Tooling

- Automate
- Mentor-like
- Focus on business logic



https://www.e-booky.com

Connascence 1st

Good

- Name
- Type

```
#   init  .py
from .collection import WIP as WIPCollection
from .material import WIP as WIPMaterial

# app.py
def GET_APP_STATE_DB (input) :
   request=input[ 0 ]
   return request.app.state._db
```

Connascence 1st

Good

- Name
- Type

```python
# Renamed WIP in packages accordingly, no import ... as

def database(request: Request) -> Database:
    return request.app.state._db
```

Styling

Black

- Opinionated
- Pretty much standard

```
def function(
        name,
        default=None,
        *args,
        variable="1123",
        a,
        b,
        c,
        employee,
        office,
        d,
        e,
        f,
        **kwargs
):
    """This is function is created to demonstrate black"""


string = "GeeksforGeeks"
```

Styling

Black

- Opinionated
- Pretty much standard

```
def function(name, default=None, *args, variable="1123", a, b, c, employee, offi
    """This is function is created to demonstrate black"""


string = 'GeeksforGeeks'

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

Connascence 2nd

Bad

- Position
- Value
- Meaning
- Algorithm
- Execution order

```
# Position interchangable
def _spellcheck(text, lang="de-DE")

 spellcheck("de-DE", "This is a text")   # better not
_spellcheck(text="This is a text", lang="de-DE")   # yes
```

```
# Implicit code
MissingField = Field("MissingField",
    [
        (f.name, (f.value, f.field type))
        for f in [ …,Attribute.NODE_ID,...
        ]
    ],
)
```

Linting

```
#   init  .py
from .collection import WIP as WIPCollection
from .material import WIP as WIPMaterial
```

```
  init  .py:17:89: E501 Line too long (112 > 88 characters)
  init  .py:5:19: F401 [*] `WIPCollection` imported but unused
ruff.py:70:17: F541 [*] f-string without any placeholders
```

Ruff

- not as big as Flake8
- Customizable
- Written in Rust

https://github.com/astral-sh/ruff

Typing

```
MissingField = Field("MissingField",
    [
        (f.name, (f.value, f.field type))
        for f in [ …,Attribute.NODE_ID,...
        ]
    ],
)
```

Mypy

- static type checker

```
mypy test.py:42: error: "Field" has no attribute "NODE ID"  [attr-defined]
mypy test.py:71: error: "ValueWeights" has no attribute "weights"  [attr-defined
mypy test.py:36: error: Variable "Base" is not valid as a type  [valid-type]
mypy_test.py:45: error: Invalid base class "Base"  [misc]
...
```

Pre-Commit hooks


```yaml
default_language_version:
   python: python3.9

repos:
 - repo: https://github.com/ambv/black
   rev: 22.3.0
   hooks:
     - id: black
       language_version: python3.9
 - repo: https://github.com/PyCQA/flake8
   rev: 4.0.1
   hooks:
     - id: flake8
       args: [ "--max-line-length",
"140","--per-file-ignores" ]
 - repo: https://github.com/jendrikseipp/vulture
   rev: 'v2.3'
   hooks:
     - id: vulture
       args: [ "app", "--min-confidence", "61" ]
```

https://pre-commit.com/

Connascence 3rd

Worst

- Timing
- Identity
- Manual Execution

Six repositories

0 tests and little documentation

Deployed manually

Testing

Lock your code first

- API Testing
- Core domain

Test good and bad cases

Ask questions to the code

```
client = TestClient(api())

def test_404():
   response = client.get("/scores")
   assert response.status_code == 404
   assert response.json() == {"errors": ["Not Found"]}

def test_get_quality():
   with mock.patch("app.api.source") as mocked_source:
       with mock.patch("app.api.collection"):
           mocked_source.return_value = Score(data=[], total={})

           response = client.get("/score")
           assert response.status_code == 422

           with pytest.raises(ValueError):
               client.get("/score", params={"node_id": ""})
```

More recommendations

Learn from others:

- Katas, Advent of Code
- Pair + Ensemble Programming
- GPT, CoPilot, AI …
- Use templates for project structure

Think about the person after you

Use an OpenAPI compliant framework/library: FastAPI

FastAPI

Summary

Legacy code

- Not necessarily bad code
- Often about circumstances

How big is the PAIN

Tools

- Automate
- Remove noise to talk business

codecentric

codecentric AG
Am Mittelhafen 14
48155 Münster

Robert Meißner
Product Owner
robert.meissner@codecentric.de
www.codecentric.de

Telefon: +49 (0) 1732816649

Legacy code

- Not necessarily bad code
- Often about circumstances

Connascence

- Enables metric driven development
- Pinpoints what to change about the code

Tools

- Black - no more “PEP 8 them!”
- Ruff
- MyPy
- Pre-Commit
- FastAPI

- Focus on business

Robert Meißner
Product Owner
robert.meissner@codecentric.de
www.codecentric.de