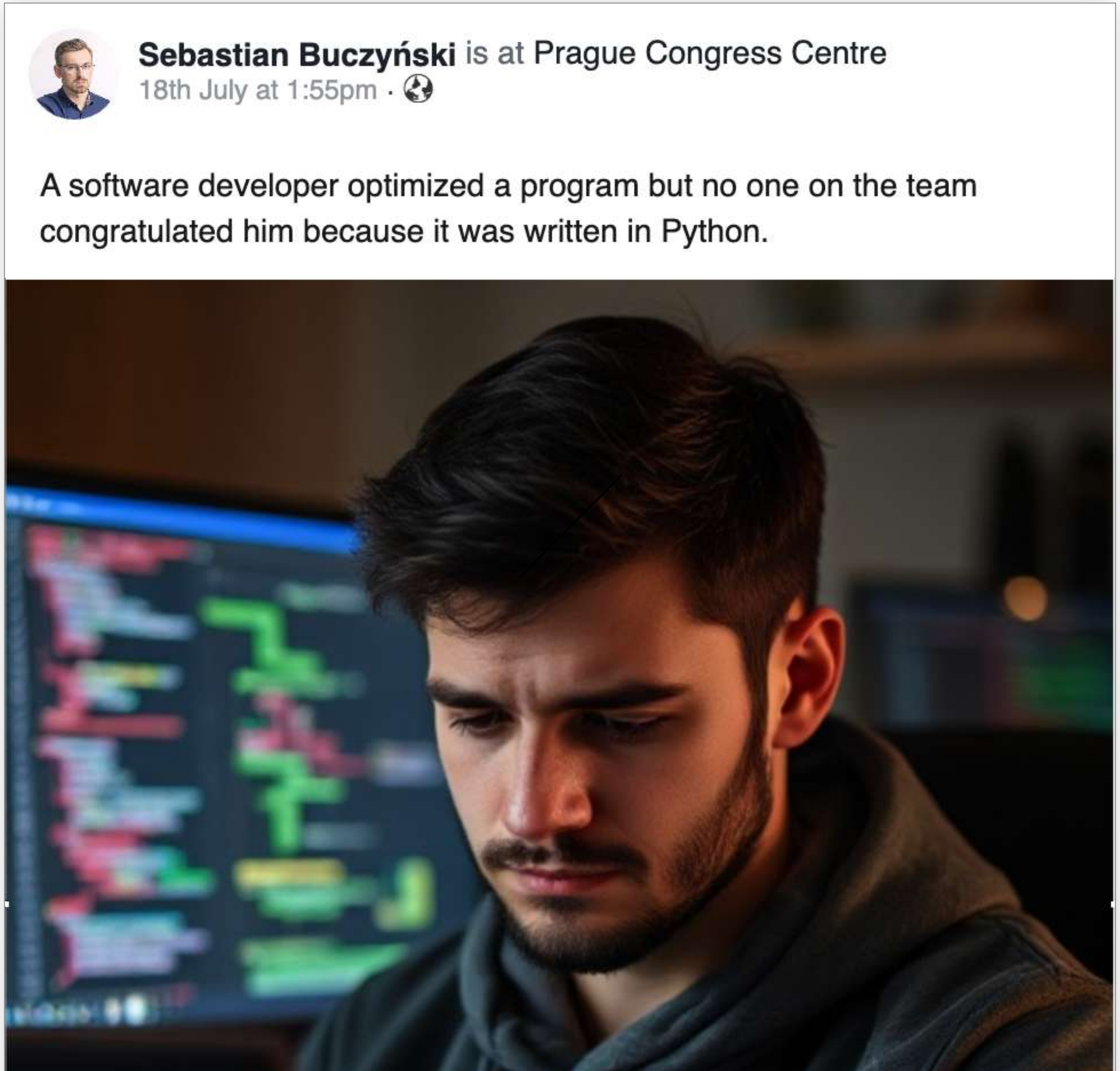
When in practice is Python performance an issue?

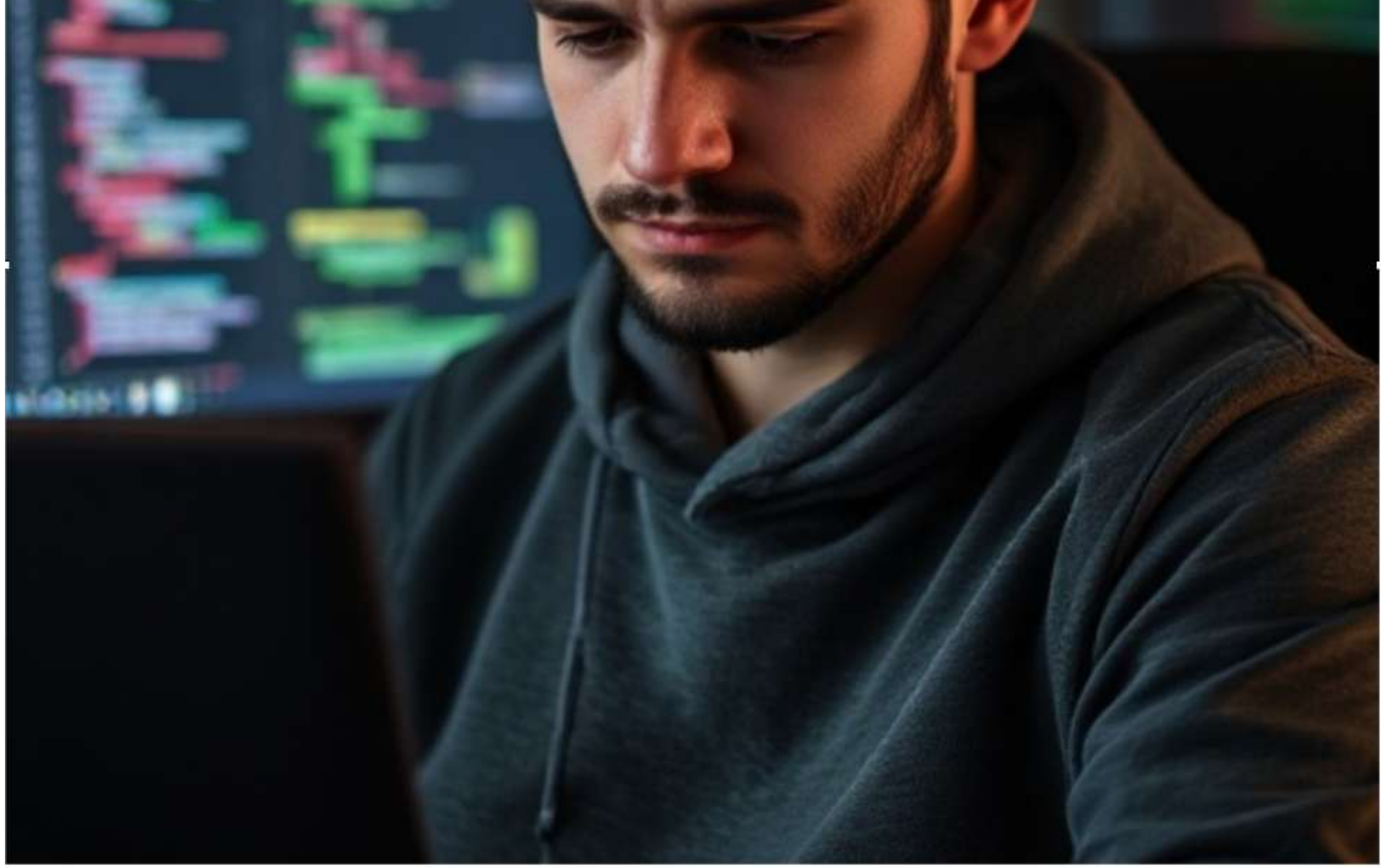
Facts and myths

Sebastian Buczyński @ EP2025



Sebastian Buczyński is at Prague Congress Centre
18th July at 1:55pm ·

A software developer optimized a program but no one on the team congratulated him because it was written in Python.



😢😮😡 2k 21 Comments 37 Shares

😢 Sad Comment Share

Bob Rustacean It's not even that fast after the optimization

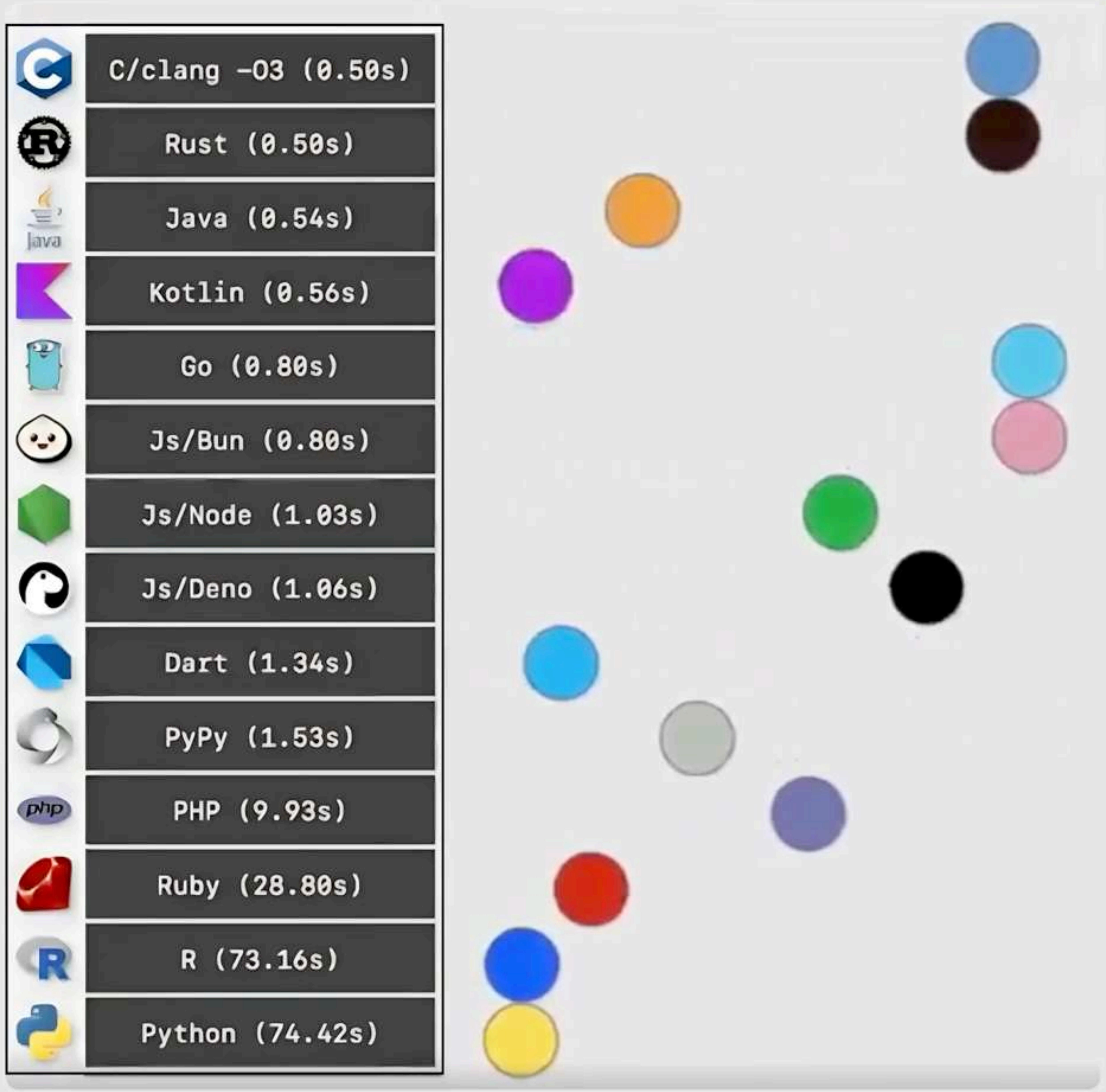
Like · Reply · 1h

1 Billion Nested Loop Iterations

In

Different Languages

Here's the shocking truth ☞



C/clang -O3 (0.50s)
Rust (0.50s)
Java (0.54s)
Kotlin (0.56s)
Go (0.80s)
Js/Bun (0.80s)
Js/Node (1.03s)
Js/Deno (1.06s)
Dart (1.34s)
PyPy (1.53s)
PHP (9.93s)
Ruby (28.80s)
R (73.16s)
Python (74.42s)

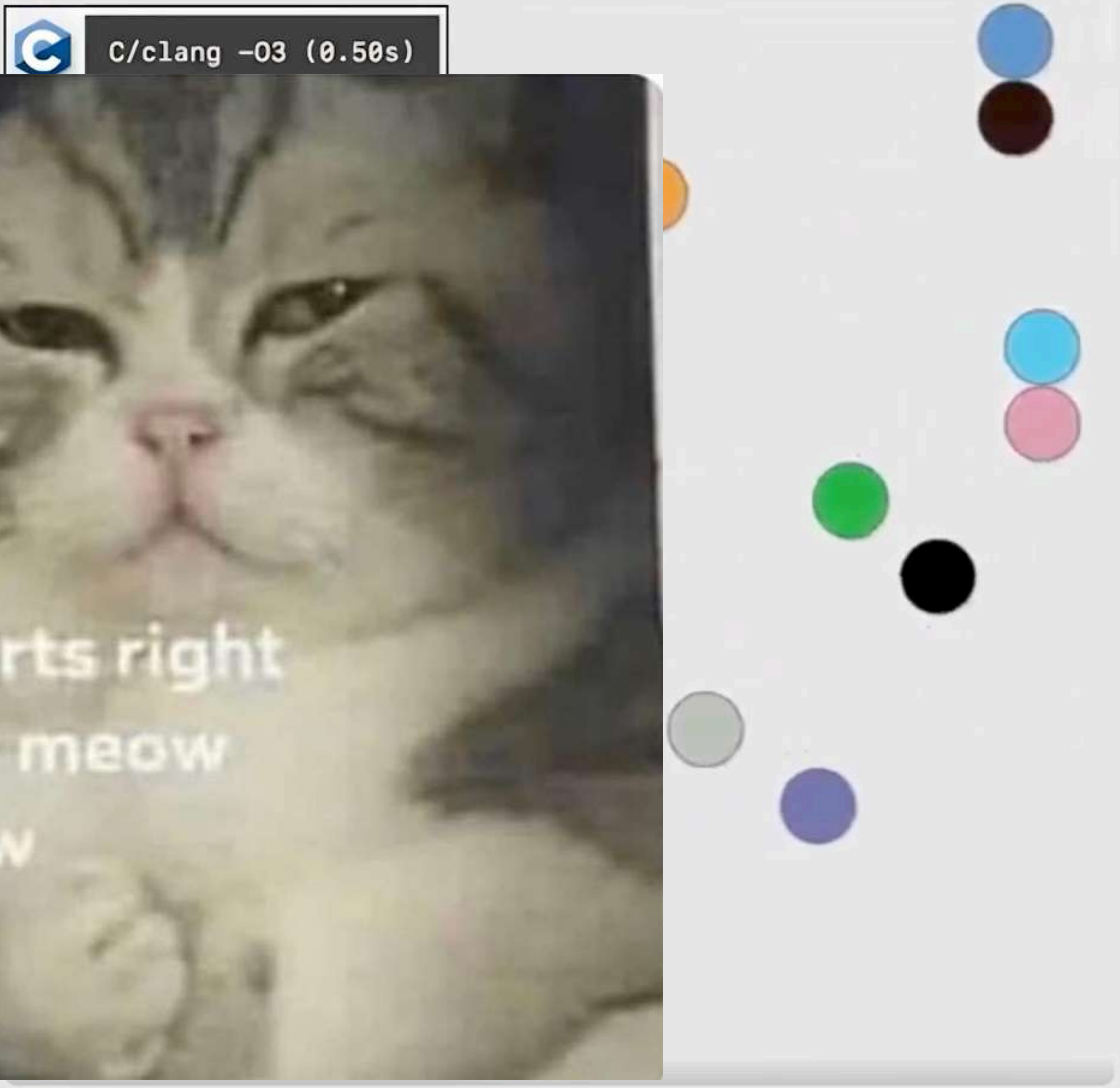
@myakcoding

1 Bi
Neste
Itera

I

Different

Here's the sho


```
C/clang -O3 (0.50s)
```

Sebastian Buczyński

Software Architect @ Sauce Labs

Trainer / Consultant

Author of Implementing the Clean Architecture book

breadcrumbscollector.tech

Learning how to ride a 🏍️

Software Engineering Educator

```python
class State(StrEnum):
    OK = "ok"
    ERROR = "error"
    FAIL = "fail"
```

```python
class State(StrEnum):
    OK = "ok"
    ERROR = "error"
    FAIL = "fail"


class Foo:
    ...

    def bar(self, state: str) -> None:
        ...
```

```python
class State(StrEnum):
    OK = "ok"
    ERROR = "error"
    FAIL = "fail"


class Foo:
    ...

    def bar(self, state: str) → None:
        is_valid_state = state in State
```

```python
class State(StrEnum):
    OK = "ok"
    ERROR = "error"
    FAIL = "fail"


class Foo:
    ...

    def bar(self, state: str) → None:
        is_valid_state = state in State
        if is_valid_state:
```

```python
class State(StrEnum):
    OK = "ok"
    ERROR = "error"
    FAIL = "fail"


class Foo:
    ...

    def bar(self, state: str) → None:
        is_valid_state = state in State
        if is_valid_state:
            now = time.time()
```

```python
class State(StrEnum):
    OK = "ok"
    ERROR = "error"
    FAIL = "fail"


class Foo:
    ...

    def bar(self, state: str) -> None:
        is_valid_state = state in State
        if is_valid_state:
            now = time.time()
            self._state_changed_at = now
            self._old_state = self._state
            self._state = state
```

```python
class State(StrEnum):
    OK = "ok"
    ERROR = "error"
    FAIL = "fail"


class Foo:
    ...

    def bar(self, state: str) → None:
        is_valid_state = state in State
        if is_valid_state:
            now = time.time()
            self._state_changed_at = now
            self._old_state = self._state
            self._state = state
```

```python
class State(StrEnum):
    OK = "ok"
    ERROR = "error"
    FAIL = "fail"


class Foo:
    ...

    def bar(self, state: str) -> None:
        is_valid_state = state in State
        if is_valid_state:
            now = time.time()
            self._state_changed_at = now
            self._old_state = self._state
            self._state = state
```

```python
class State(StrEnum):
    OK = "ok"
    ERROR = "error"
    FAIL = "fail"


class Foo:
    ...

    def bar(self, state: str) -> None:
        is_valid_state = state in State
        if is_valid_state:
            now = time.time()
            self._state_changed_at = now
            self._old_state = self._state
            self._state = state
```

```python
class State(StrEnum):
    OK = "ok"
    ERROR = "error"
    FAIL = "fail"


class Foo:
    ...

    def bar(self, state: str) → None:
        is_valid_state = state in State
        if is_valid_state:
            now = time.time()
            self._state_changed_at = now
            self._old_state = self._state
            self._state = state
```

```
class State(StrEnum):
    OK = "ok"
    ERROR = "error"
    FAIL = "fail"


class Foo:
    ...

    def bar(self, state: str) -> None:
        is_valid_state = state in State
        if is_valid_state:
            now = time.time()
            self._state_changed_at = now
            self._old_state = self._state
            self._state = state
```

Eyeballing

Intuition

```
Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
    21                                           def bar(self, state: str) → None:
    22      1000        721.0      0.7     28.3      is_valid_state = state in State
    23      1000        270.0      0.3     10.6      if is_valid_state:
    24      1000        371.0      0.4     14.5          now = time.time()
    25      1000        255.0      0.3     10.0          self._state_changed_at = now
    26      1000        304.0      0.3     11.9          self._old_state = self._state
    27      1000        277.0      0.3     10.9          self._state = state
```

imaginary knowledge

=

imaginary improvements

```python
class Foo:
    ...

    def bar(self, state: str) -> None:
        is_valid_state = state in State
        if is_valid_state:
            now = time.time()
            self._state_changed_at = now
            self._old_state = self._state
            self._state = state
```

```python
from line_profiler import profile


class Foo:
    ...

    def bar(self, state: str) -> None:
        is_valid_state = state in State
        if is_valid_state:
            now = time.time()
            self._state_changed_at = now
            self._old_state = self._state
            self._state = state
```

```python
from line_profiler import profile


class Foo:
    ...

    @profile
    def bar(self, state: str) -> None:
        is_valid_state = state in State
        if is_valid_state:
            now = time.time()
            self._state_changed_at = now
            self._old_state = self._state
            self._state = state
```

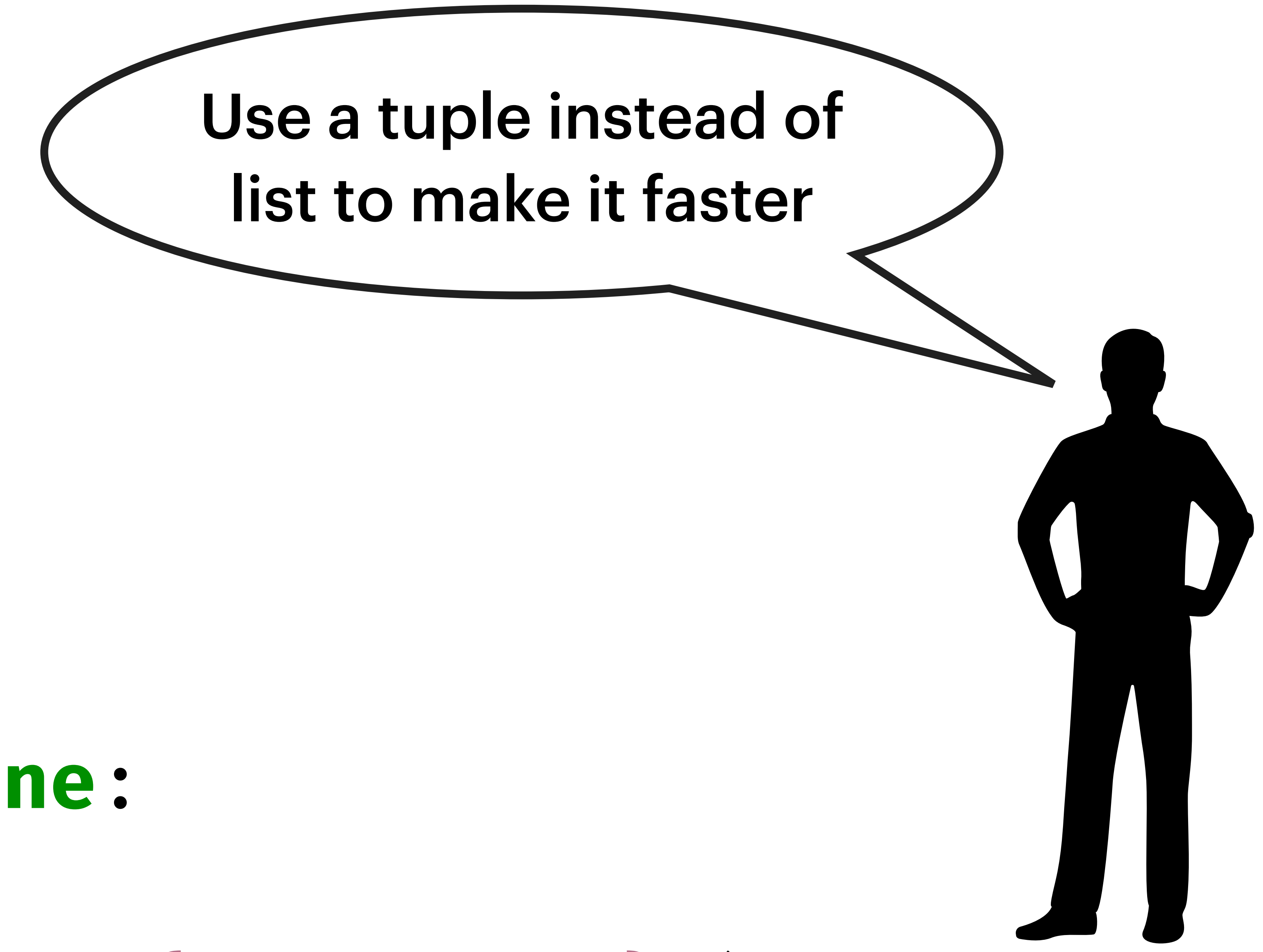
```
ve313 ~/Projects/private/python-perf-talk (0.291s)
env LINE_PROFILE=1 python 00_tricky.py
Timer unit: 1e-09 s

  0.00 seconds - /Users/spb/Projects/private/python-perf-talk/00_tricky.py:19 - bar
Wrote profile results to profile_output.txt
Wrote profile results to profile_output_2025-01-29T165617.txt
Wrote profile results to profile_output.lprof
To view details run:
python -m line_profiler -rtmz profile_output.lprof

ve313 ~/Projects/private/python-perf-talk
```

act on data, not intuition*

*when it comes to performance

```python
ALLOWED = ["new", "pending", "done"]


def process_state(new_state: str) -> None:
    if new_state not in ALLOWED:
        raise ValueError(f"Invalid state: {new_state}”)
    ...
```

```python
ALLOWED = ["new", "pending", "done"]


def process_state(new_state: str) -> None:
    if new_state not in ALLOWED:
        raise ValueError(f"Invalid state: {new_state}”)
    ...
```



```python
ALLOWED = ("new", "pending", "done")


def process_state(new_state: str) -> None:
    if new_state not in ALLOWED:
        raise ValueError(f"Invalid state: {new_state}”)
    ...
```

efficient lookup?

set

```
ALLOWED = {"new", "pending", "done"}
```

Measuring

Data structures

```
python -m timeit '<code>'
python -m timeit 'ALLOWED = {"new", "pending", "done"}
"done" in ALLOWED'
```

```
python -m timeit
```

```
ending", "done"}
```

```
python -m timeit '<code>'
python -m timeit -s '<setup>' '<code>'
python -m timeit
-s 'collection = ["new", "pending", "done"]'
'"done" in collection'
```

```
# list
5000000 loops, best of 5: 36 nsec per loop

# tuple
10000000 loops, best of 5: 32.2 nsec per loop (+10,5%)

# set
10000000 loops, best of 5: 20.5 nsec per loop (+43%)
```

Lookup on 3-elements long list.
Is this a problem?

solving microproblem

=

microprofit

The junior asking to rewrite the project in rust
The senior dev

not just solving the problem

choosing the right problem to solve

performance - when to bother?

performance - when to bother?

required for a solution to „work”

performance - when to bother?

required for a solution to „work”

required for User Experience

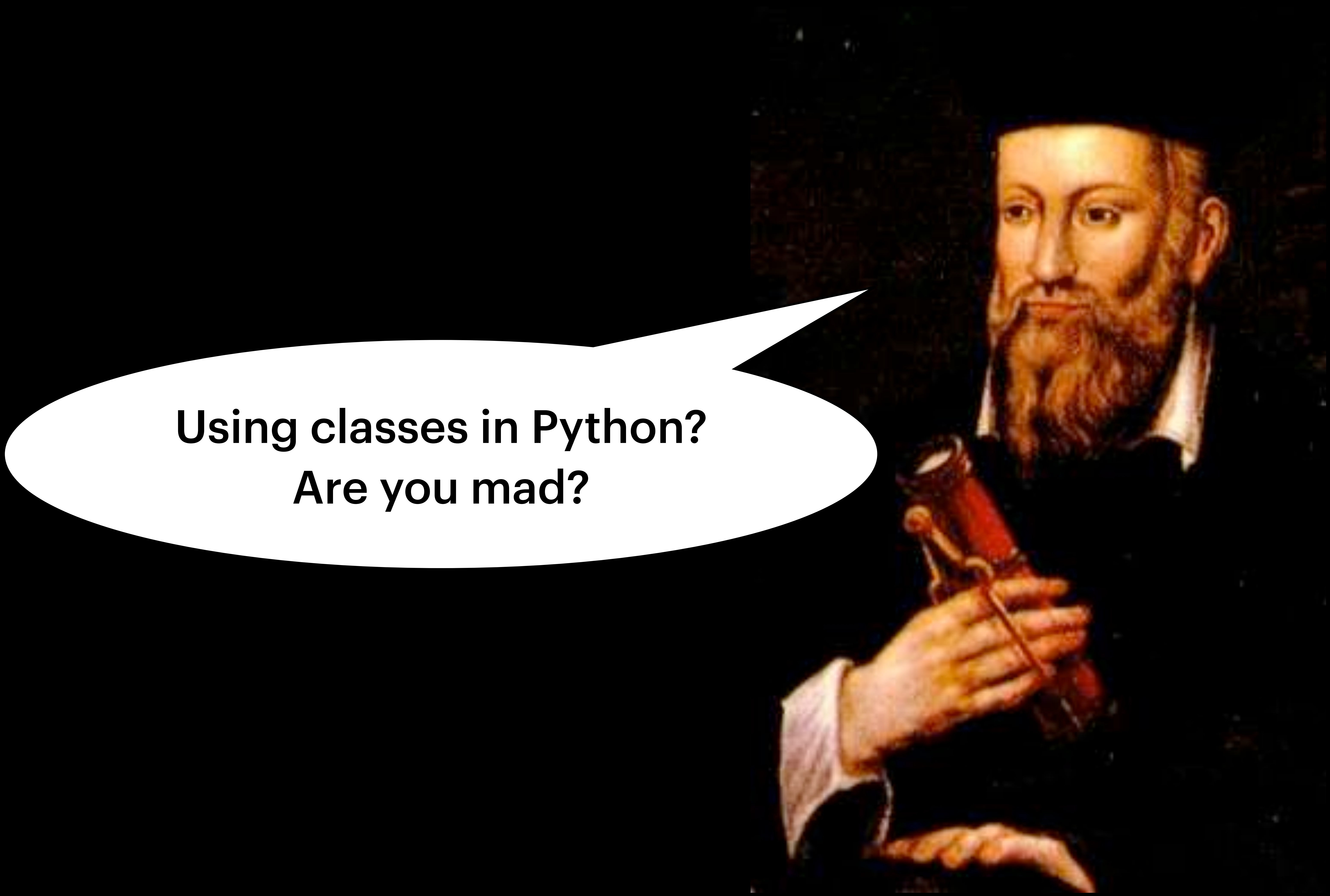
performance - when to bother?

required for a solution to „work”

required for User Experience

competition is faster and it matters

knowing where to improve

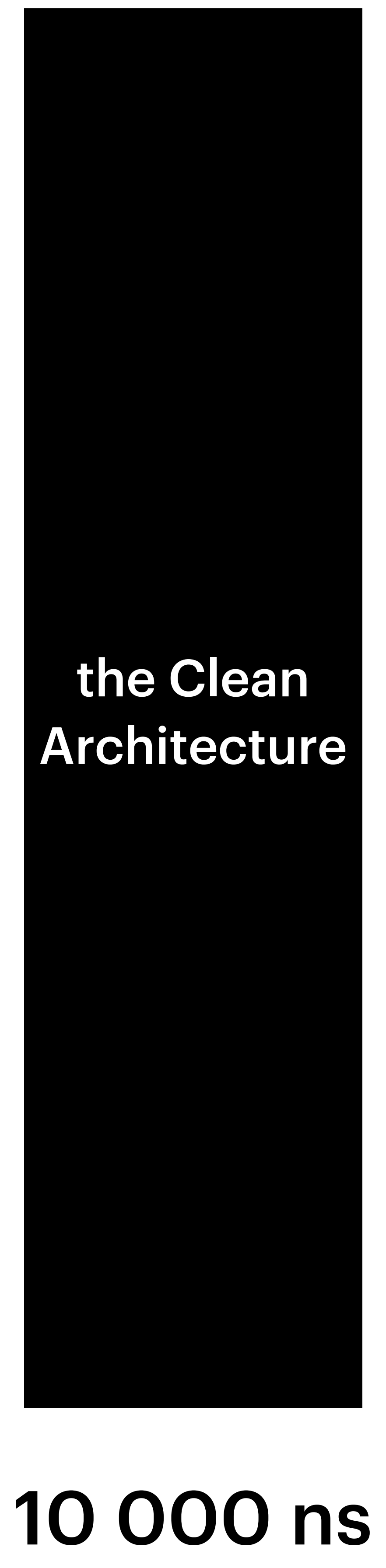

Using classes in Python? Are you mad?

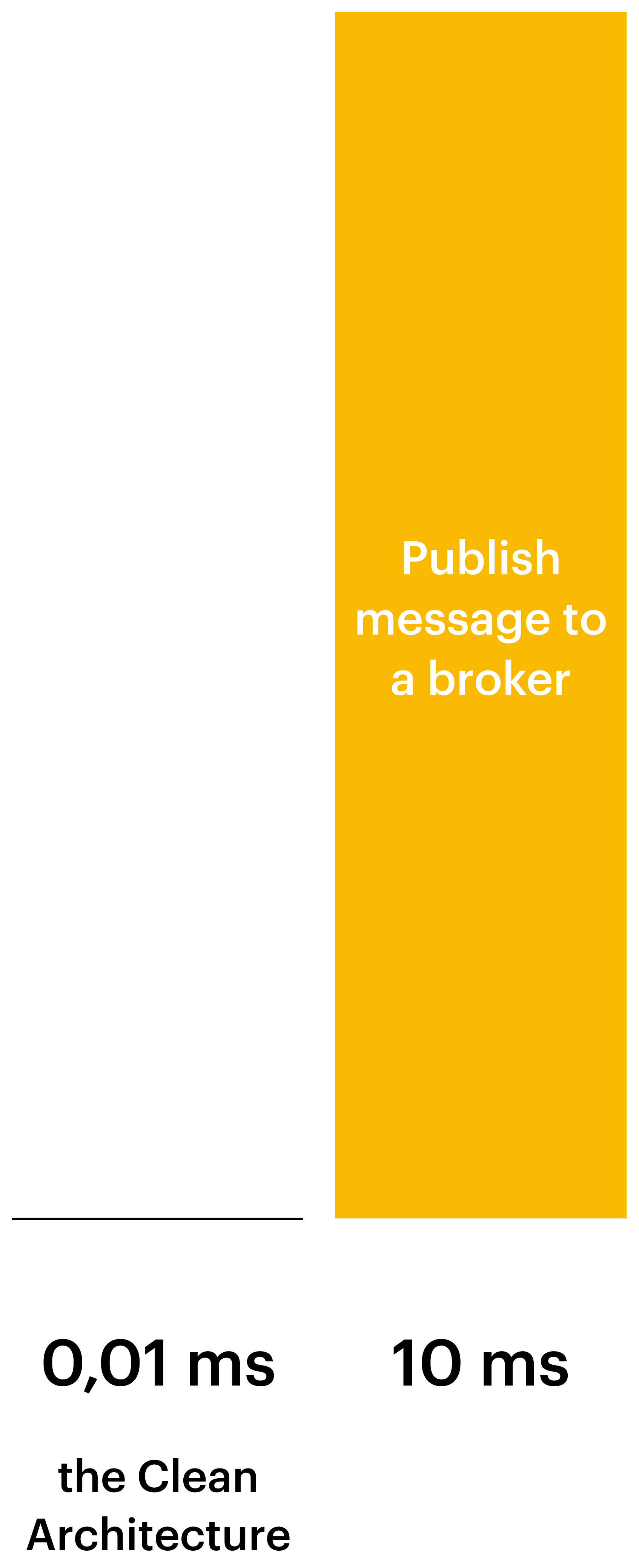
```python
class NewOrderService:
    def __init__(self, repository: Repository) → None:
        self._repository = repository

    @transaction
    def add_new_order(self) → None:
        order = Order(status="new")
        self._repository.add(order)
```

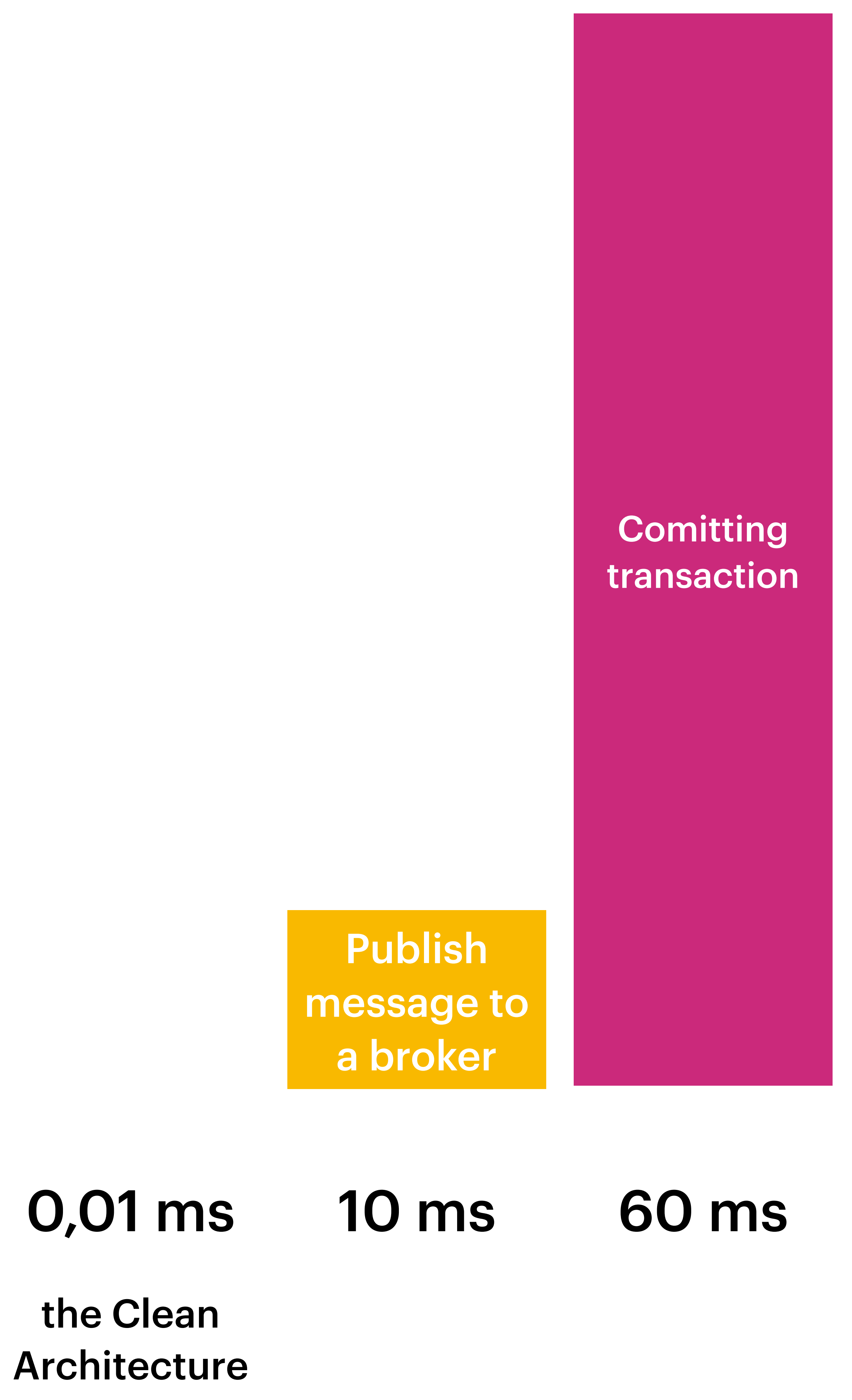
```python
class NewOrderService:
    def __init__(self, repository: Repository) -> None:
        self._repository = repository

    @transaction
    def add_new_order(self) -> None:
        order = Order(status="new")
        self._repository.add(order)


class SaRepository(Repository):
    def __init__(self, session: Session) -> None:
        self._session = session

    def add(self, order: Order) -> None:
        self._session.add(order)
        self._session.flush([order])
```

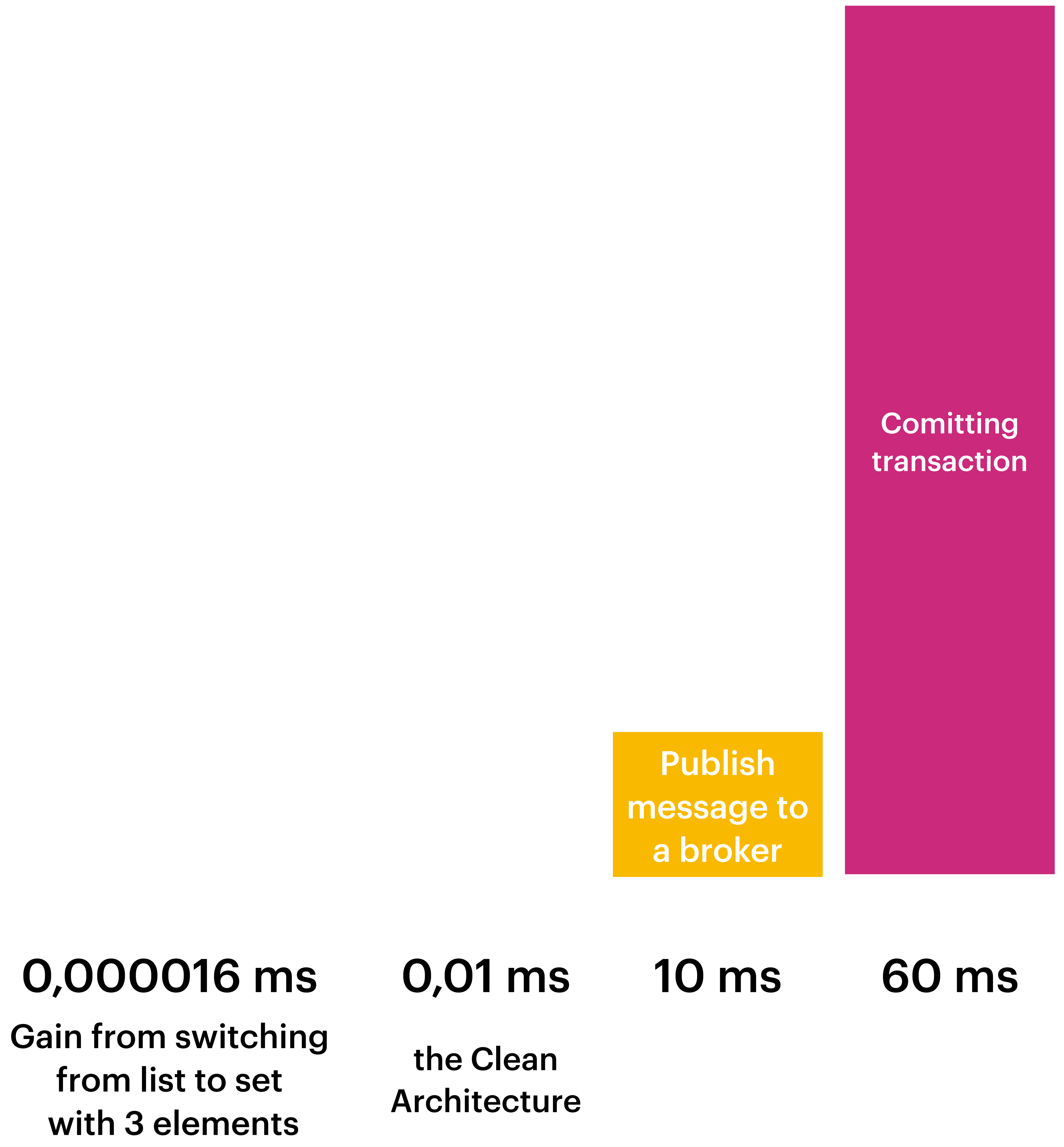
Overhead of the Clean Architecture?

10 000 ns


the Clean Architecture
10 000 ns


Publish message to a broker
0,01 ms
10 ms
the Clean Architecture


Comitting transaction
Publish message to a broker
0,01 ms
10 ms
60 ms
the Clean Architecture


Comitting transaction
Publish message to a broker
0,000016 ms
Gain from switching from list to set with 3 elements
0,01 ms
the Clean Architecture
10 ms
60 ms

consider big picture

tens of milliseconds? sure
less? not really

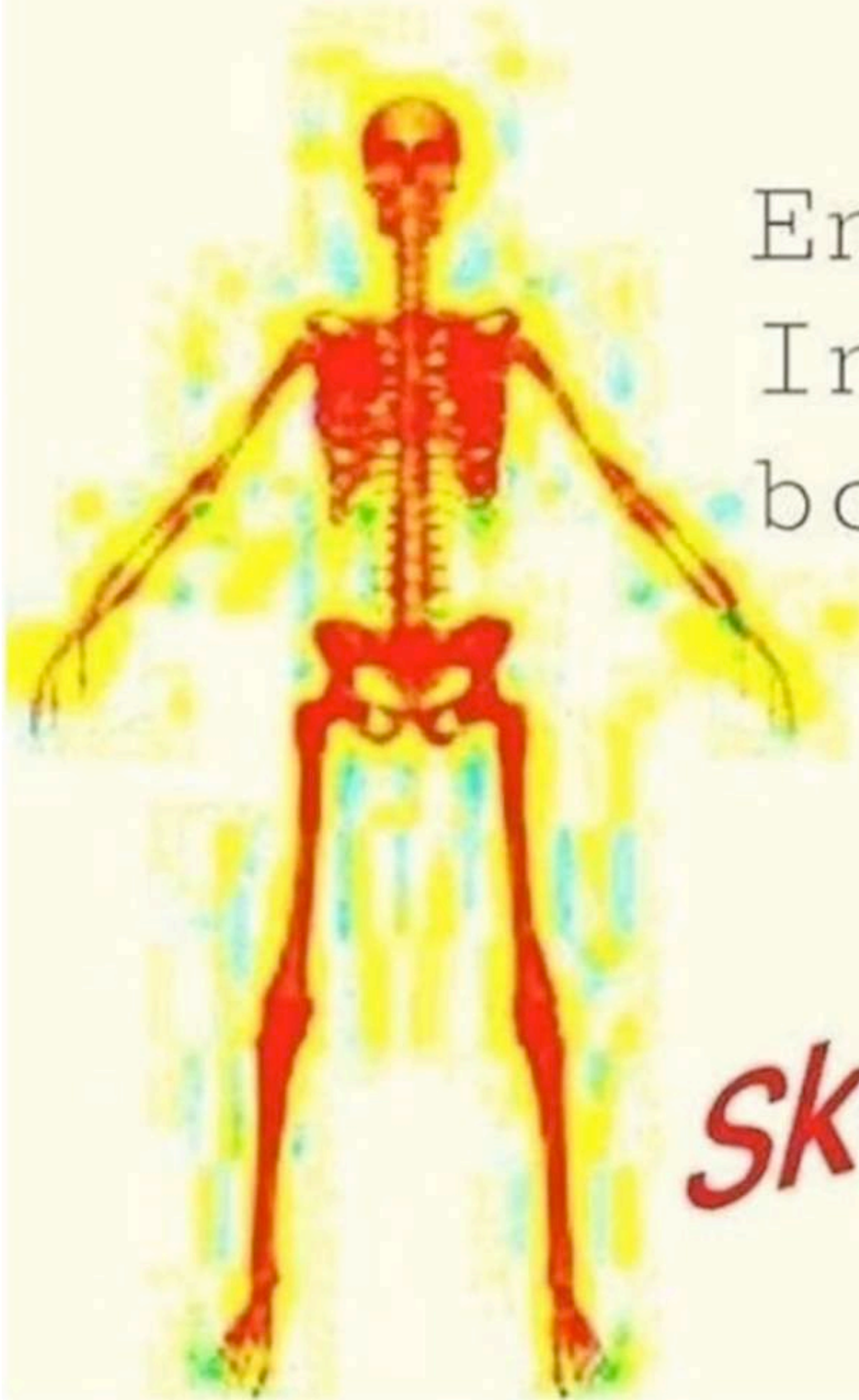
```
python3.13 -m timeit -s "from time import time
def foo():
    time()
" "foo()"

2000000 loops, best of 5: 84 nsec per loop
```

```
python3.13 -m timeit -s "from time import time"
"time()"

5000000 loops, best of 5: 55.5 nsec per loop (-28.5 nsec)
```

with experimental JIT it’s 16.4 ns difference

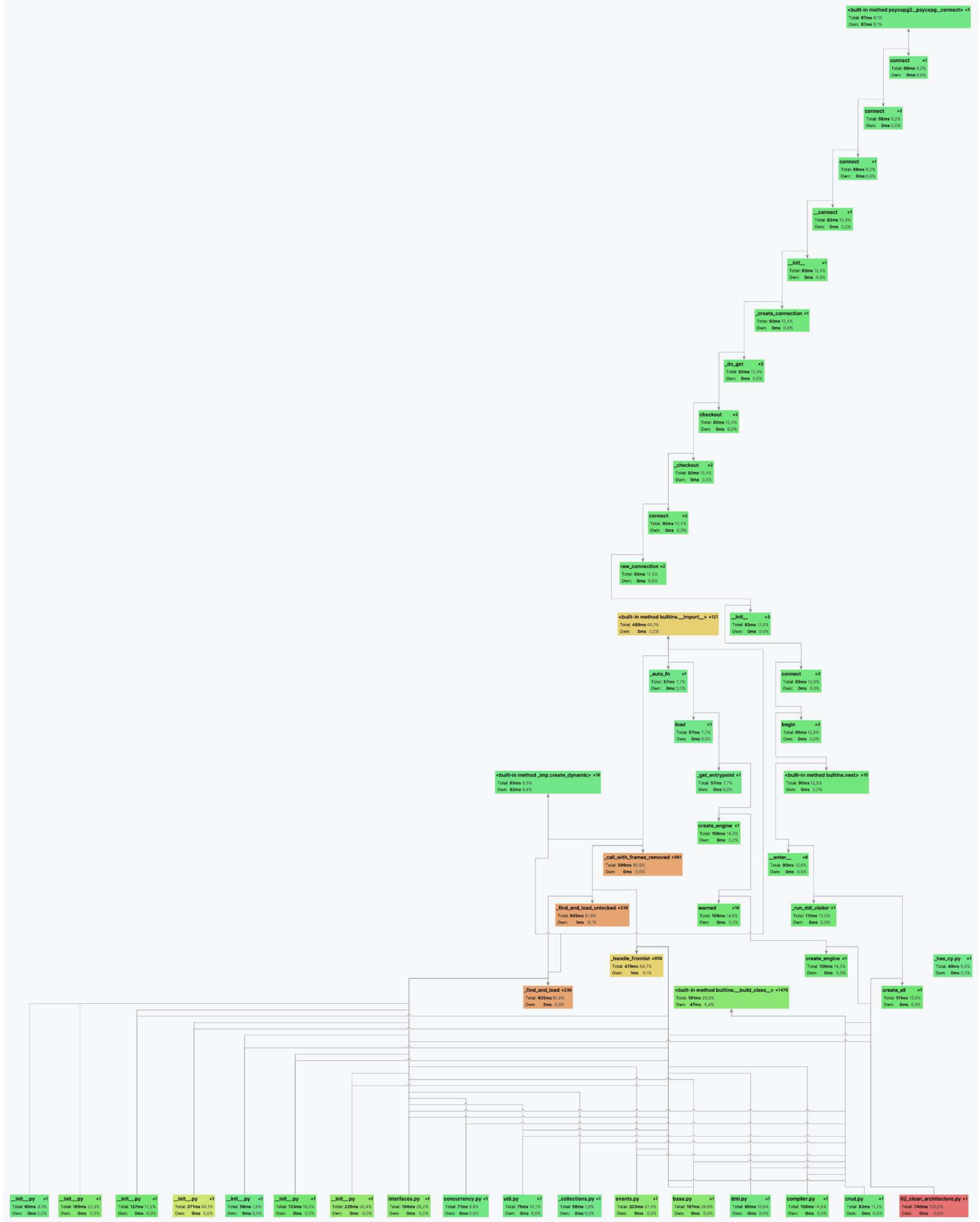
Your favourite libraries also have abstraction layers!

Did u know



There is
Enough bones
In the human
body to make
An entire

Skeleton?


<built-in method psycopg2._psycopg._connect> ×1 Total: 67ms 9,1% Own: 67ms 9,1%
connect ×1 Total: 68ms 9,2% Own: 0ms 0,0%
connect ×1 Total: 68ms 9,2% Own: 0ms 0,0%
connect ×1 Total: 68ms 9,2% Own: 0ms 0,0%
__connect ×1 Total: 92ms 12,4% Own: 0ms 0,0%
__init__ ×1 Total: 92ms 12,4% Own: 0ms 0,0%
_create_connection ×1 Total: 92ms 12,4% Own: 0ms 0,0%
_do_get ×2 Total: 92ms 12,4% Own: 0ms 0,0%
checkout ×2 Total: 92ms 12,4% Own: 0ms 0,0%
_checkout ×2 Total: 92ms 12,4% Own: 0ms 0,0%
connect ×2 Total: 92ms 12,4% Own: 0ms 0,0%
raw_connection ×2 Total: 92ms 12,4% Own: 0ms 0,0%
<built-in method builtins.__import__> ×121 Total: 489ms 66,1% Own: 0ms 0,0%
__init__ ×3 Total: 93ms 12,6% Own: 0ms 0,0%
_auto_fn ×1 Total: 57ms 7,7% Own: 0ms 0,0%
connect ×2 Total: 93ms 12,6% Own: 0ms 0,0%
load ×1 Total: 57ms 7,7% Own: 0ms 0,0%
begin ×2 Total: 95ms 12,8% Own: 0ms 0,0%
<built-in method _imp.create_dynamic> ×16 Total: 63ms 8,5% Own: 62ms 8,4%
_get_entrypoint ×1 Total: 57ms 7,7% Own: 0ms 0,0%
<built-in method builtins.next> ×15 Total: 95ms 12,8% Own: 0ms 0,0%
create_engine ×1 Total: 106ms 14,3% Own: 0ms 0,0%
_call_with_frames_removed ×561 Total: 599ms 80,9% Own: 0ms 0,0%
__enter__ ×6 Total: 93ms 12,6% Own: 0ms 0,0%
_find_and_load_unlocked ×236 Total: 605ms 81,8% Own: 1ms 0,1%
warned ×16 Total: 108ms 14,6% Own: 0ms 0,0%
_run_ddl_visitor ×1 Total: 111ms 15,0% Own: 0ms 0,0%
_handle_fromlist ×858 Total: 479ms 64,7% Own: 1ms 0,1%
create_engine ×1 Total: 106ms 14,3% Own: 0ms 0,0%
_has_cy.py ×1 Total: 49ms 6,6% Own: 0ms 0,0%
_find_and_load ×236 Total: 605ms 81,8% Own: 2ms 0,3%
<built-in method builtins.__build_class__> ×1475 Total: 191ms 25,8% Own: 47ms 6,4%
create_all ×1 Total: 111ms 15,0% Own: 0ms 0,0%
__init__.py ×1 Total: 60ms 8,1% Own: 0ms 0,0%
__init__.py ×1 Total: 165ms 22,3% Own: 0ms 0,0%
__init__.py ×1 Total: 127ms 17,2% Own: 0ms 0,0%
__init__.py ×1 Total: 371ms 50,1% Own: 0ms 0,0%
__init__.py ×1 Total: 56ms 7,6% Own: 0ms 0,0%
__init__.py ×1 Total: 133ms 18,0% Own: 0ms 0,0%
__init__.py ×1 Total: 225ms 30,4% Own: 0ms 0,0%
interfaces.py ×1 Total: 194ms 26,2% Own: 0ms 0,0%
concurrency.py ×1 Total: 71ms 9,6% Own: 0ms 0,0%
util.py ×1 Total: 75ms 10,1% Own: 0ms 0,0%
_collections.py ×1 Total: 56ms 7,6% Own: 0ms 0,0%
events.py ×1 Total: 203ms 27,4% Own: 0ms 0,0%
base.py ×1 Total: 197ms 26,6% Own: 0ms 0,0%
dml.py ×1 Total: 80ms 10,8% Own: 0ms 0,0%
compiler.py ×1 Total: 108ms 14,6% Own: 0ms 0,0%
crud.py ×1 Total: 83ms 11,2% Own: 0ms 0,0%
02_clean_architecture.py ×1 Total: 740ms 100,0% Own: 0ms 0,0%

People who know
People who don't know

1 Billion Nested Loop Iterations

In

Different Languages

Here's the shocking truth

@myakcoding

tests

never enough

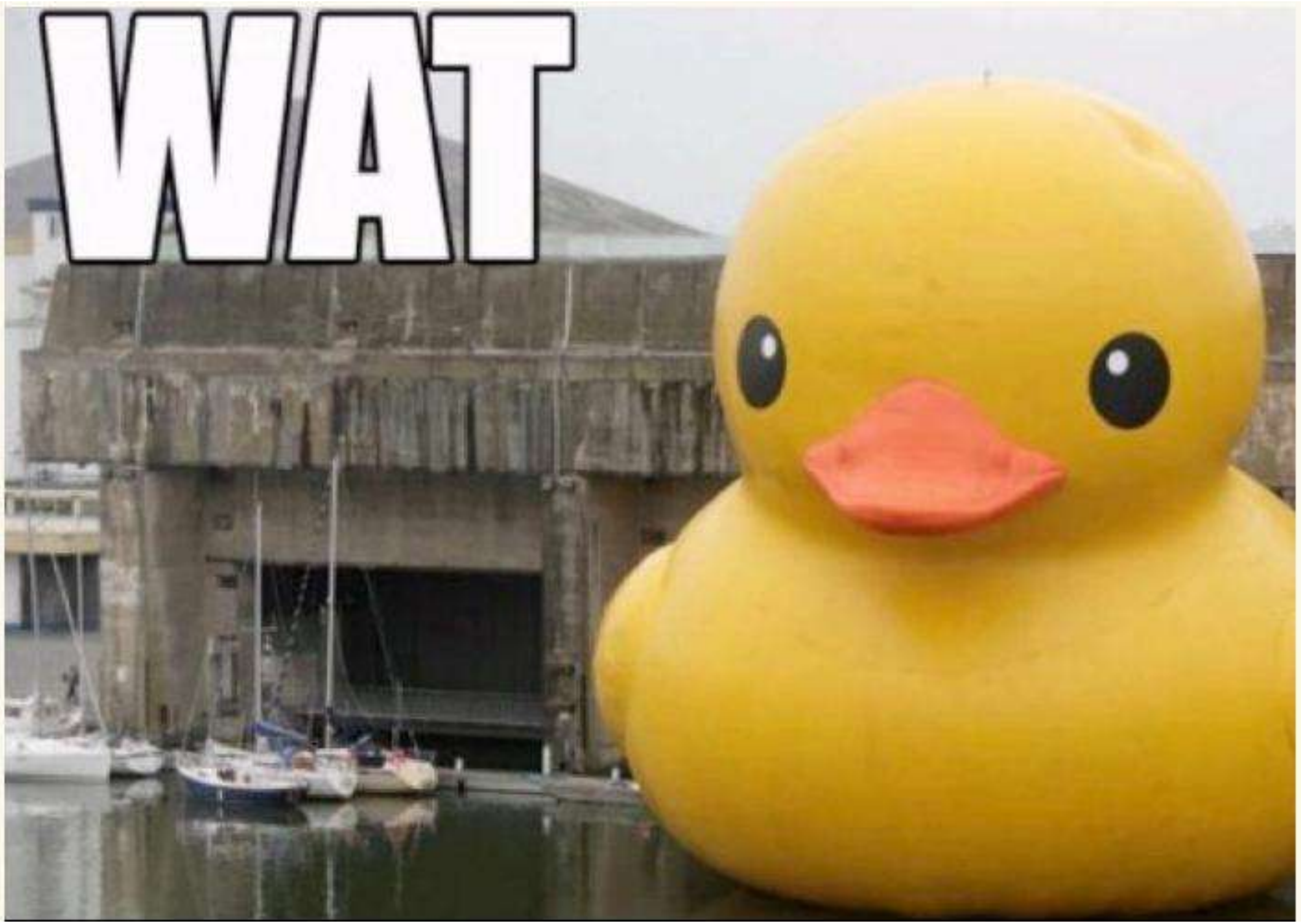
never fast enough

~ 500 tests

< 10 tests using DB

besides, everything uses Mocks

10 minutes


WAT

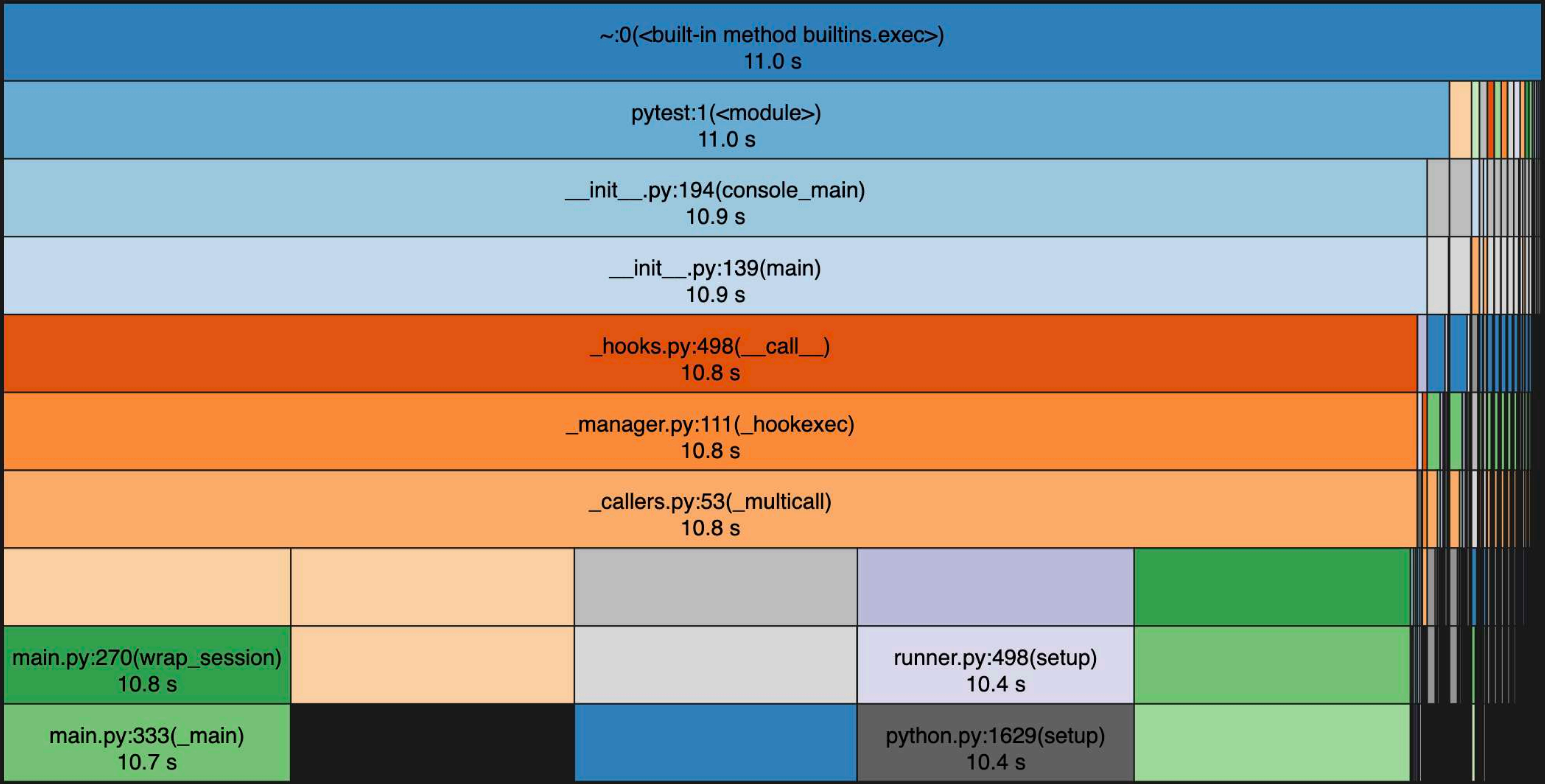
```python
@pytest.mark.parametrize("param", range(1_000))
def test_(mock_with_autospec: MagicMock, param: int) -> None:
    ...
```

```python
@pytest.fixture()
def mock_with_autospec() -> MagicMock:
    return create_autospec(SomeClass)


@pytest.mark.parametrize("param", range(1_000))
def test_(mock_with_autospec: MagicMock, param: int) -> None:
    ...
```

```python
@pytest.fixture()
def mock_with_autospec() -> MagicMock:
    return create_autospec(MonsterClass)


@pytest.mark.parametrize("param", range(1_000))
def test_(mock_with_autospec: MagicMock, param: int) -> None:
    ...
```

```
python -m cProfile -o out.prof $(which pytest) file_tests.py
```

```
pip install snakeviz

snakeviz out.prof
```

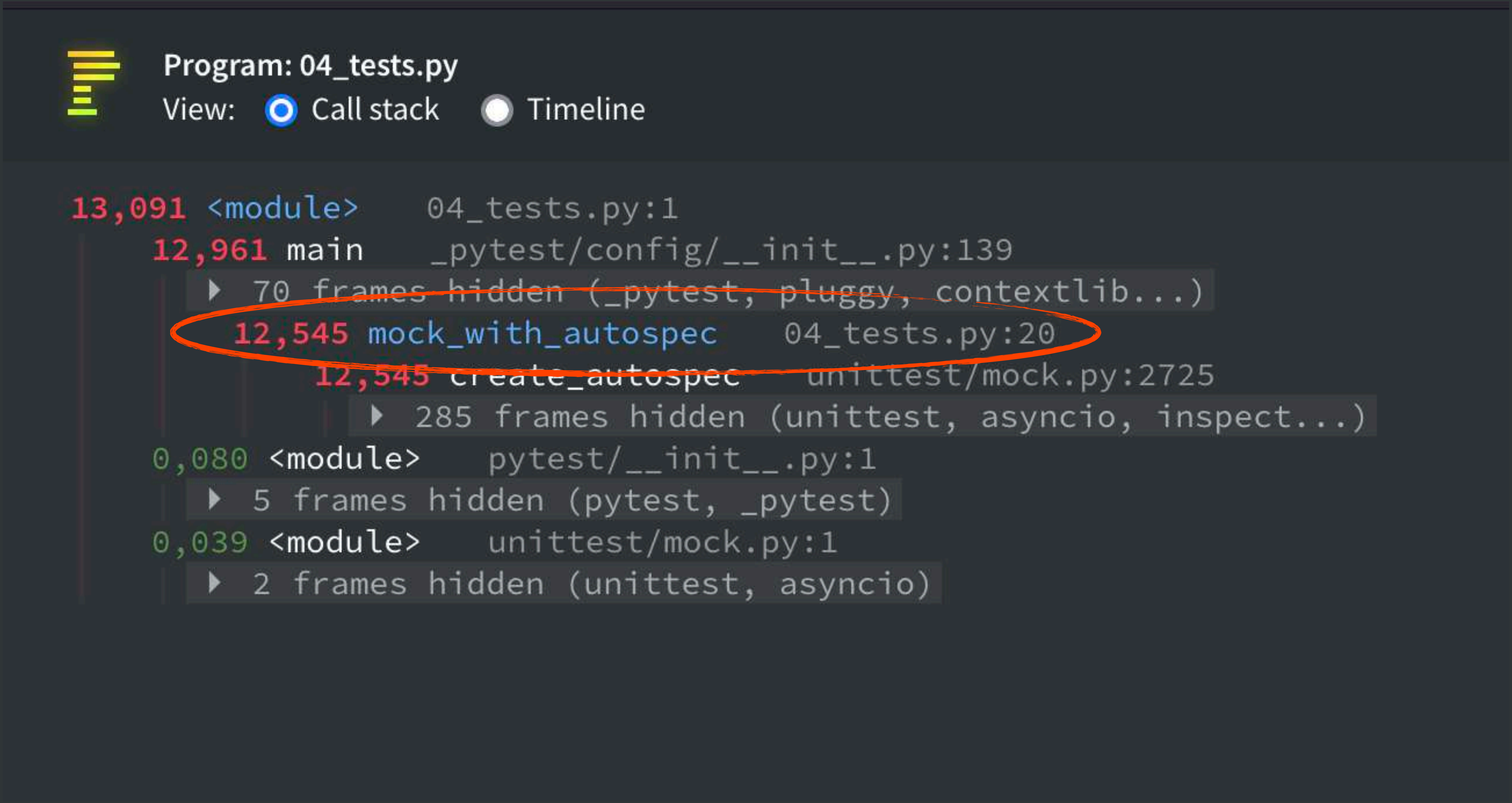

~:0(<built-in method builtins.exec>)
11.0 s
pytest:1(<module>)
11.0 s
__init__.py:194(console_main)
10.9 s
__init__.py:139(main)
10.9 s
_hooks.py:498(__call__)
10.8 s
_manager.py:111(_hookexec)
10.8 s
_callers.py:53(_multicall)
10.8 s
main.py:270(wrap_session)
10.8 s
main.py:333(_main)
10.7 s
runner.py:498(setup)
10.4 s
python.py:1629(setup)
10.4 s

Search:

04_tests.py:20(mock_with_autospec)
10.4 s
mock.py:2725(create_autospec)
10.4 s
mock.py:2156(__init__)
8.84 s
mock.py:1145(__init__)
5.84 s
mock.py:2162(_mock_set_magics)
2.88 s
mock.py:463(__init__)
5.73 s
mock.py:532(_mock_add_spec)
5.67 s
coroutines.py:20(iscoroutinefunction)
2.72 s
inspect.py:437(iscoroutinefunction)
2.52 s
inspect.py:399(_has_code_flag)
1.86 s
Search: 04_tests

```
pip install pyinstrument

pyinstrument -r html 04_tests.py
```

```python
@pytest.fixture()
def mock_with_autospec() -> MagicMock:
    return create_autospec(SomeClass)


@pytest.mark.parametrize("param", range(1_000))
def test_(mock_with_autospec: MagicMock, param: int) -> None:
    ...


if __name__ == "__main__":
    pytest.main(["-v", __file__])
```



Program: 04_tests.py

View: [● Call stack] ○ Timeline

```
13,091 <module>    04_tests.py:1
    12,961 main    _pytest/config/__init__.py:139
       ▸ 70 frames hidden (_pytest, pluggy, contextlib...)
          12,545 mock_with_autospec    04_tests.py:20
              12,545 create_autospec    unittest/mock.py:2725
                 ▸ 285 frames hidden (unittest, asyncio, inspect...)
    0,080 <module>    pytest/__init__.py:1
       ▸ 5 frames hidden (pytest, _pytest)
    0,039 <module>    unittest/mock.py:1
       ▸ 2 frames hidden (unittest, asyncio)
```

```python
@pytest.fixture()
def mock_with_autospec() -> MagicMock:
    return create_autospec(SomeClass)


@pytest.mark.parametrize("param", range(1_000))
def test_(mock_with_autospec: MagicMock, param: int) -> None:
    ...
```

```python
@pytest.fixture(scope="session")
def mock_with_autospec() -> MagicMock:
    return create_autospec(SomeClass)


@pytest.mark.parametrize("param", range(1_000))
def test_(mock_with_autospec: MagicMock, param: int) -> None:
    ...
```

```python
@pytest.fixture(scope="session")
def _mock_with_autospec() -> MagicMock:
    return create_autospec(ComplexClass)


@pytest.fixture()
def mock_with_autospec(_mock_with_autospec: MagicMock) -> MagicMock:
    yield _mock_with_autospec
    _mock_with_autospec.reset_mock(return_value=True, side_effect=True)


@pytest.mark.parametrize("param", range(1_000))
def test_(mock_with_autospec: MagicMock, param: int) -> None:
    ...
```

profile your tests to find opportunities

Python is not suitable for building high-performance systems

NDA

```
class Tank:
    max_speed: Speed
```

```
class Tank:
    max_speed: Speed
    armor: Armor
    weaponry: list[Weapon]
    ...
```

82
37
92

will Python be fast enough?

~~will Python be fast enough?~~

how fast is enough?

latency or throughput

what are requirements?

pick right data structures and database(s)

where will the bottlenecks be?

how will we scale it?

asyncio? threads?

Understand the problem > design architecture > programming language

„It’s not just Rust that makes Ruff fast”

Charlie Marsh, Pycon US 2024, https://youtu.be/r1EZ3GXuwBA?si=fiVFO2ugeqBdkX6P&t=632

Python is not the right choice in all cases

But it can do a lot.

Takeaways

learn data structures

learn how to profile

timeit, pyinstrument

intuition and performance improvements do NOT go well together

Python is getting faster

you don’t have to wait for anything

~~will X be fast enough?~~

how fast is enough?

Sebastian Buczyński

breadcrumbscollector.tech

Used resources

- https://pixabay.com/cs/photos/kompas-orientace-mapa-adresa-sever-5261062/
- https://pixabay.com/cs/photos/boy-black-white-hair-summer-k%C3%A1men-4696117/
- https://pixabay.com/pl/photos/czas-chronometr-chronograf-stoper-7683808/
- Tengr.ai
- https://pixabay.com/pl/photos/ma%C5%82pa-dzikiej-przyrody-tr%C4%85ba-4738505/
- https://pixabay.com/pl/photos/nosacz-ma%C5%82pa-rzadko-spotykany-dziki-216219/
- https://pixabay.com/pl/photos/nosacz-ma%C5%82pa-rzadko-spotykany-dziki-216215/
- https://pixabay.com/photos/measuring-tape-measurement-tools-926716/
- https://pixabay.com/photos/battle-abbey-monastery-vault-369004/
- https://pixabay.com/photos/ibex-mountain-goats-water-animals-8052387/
- https://pixabay.com/photos/car-racing-race-track-n%C3%BCrburgring-4394450/