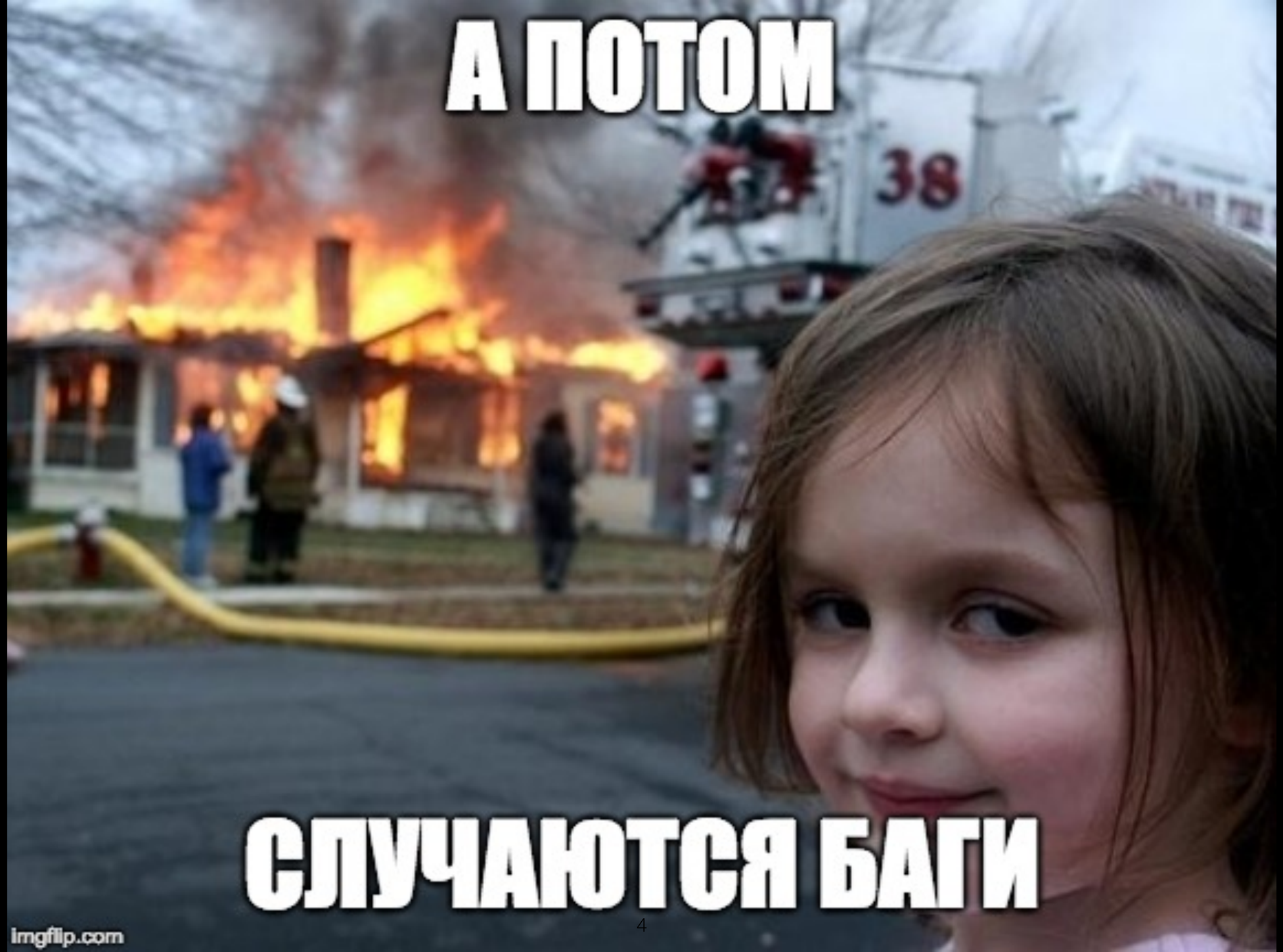
Никита Соболев

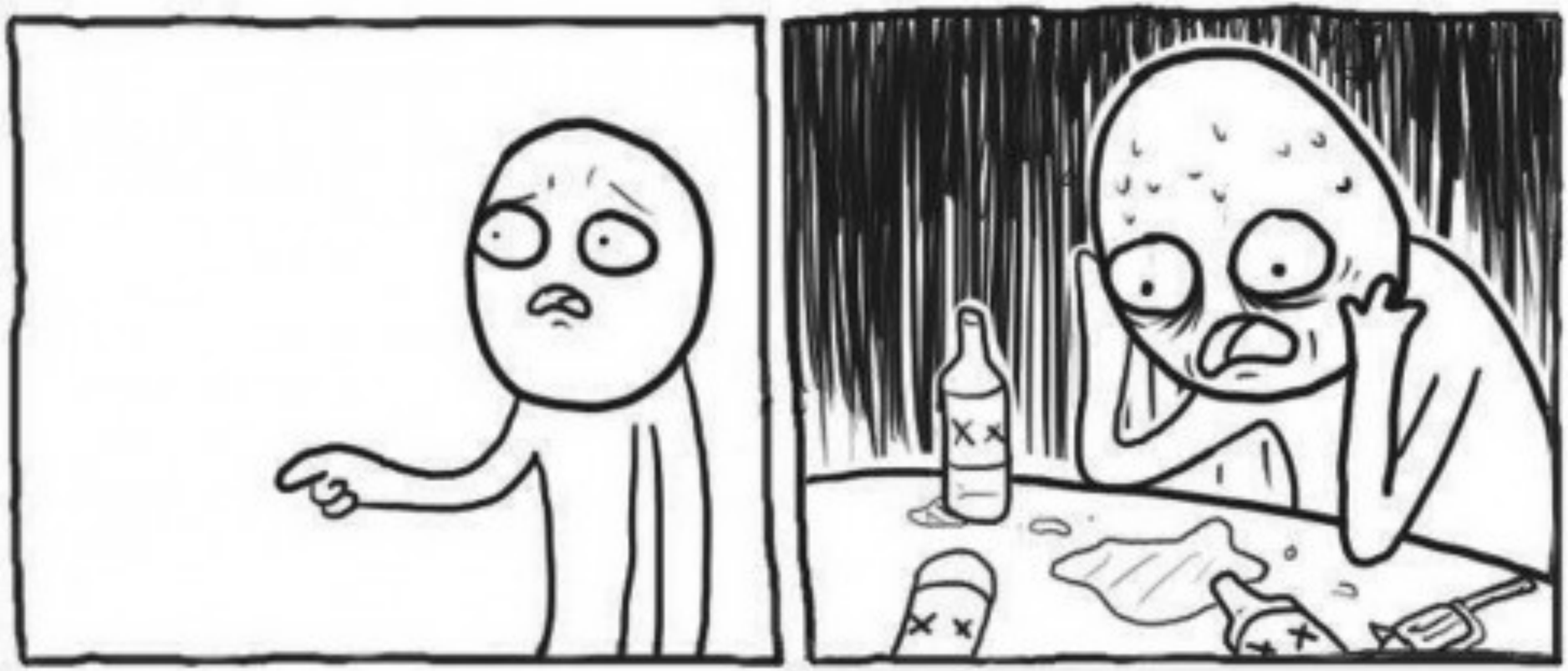
github.com/sobolevn

Мутационное тестирование

Как мы работаем?

1. Прилетает Pull Request с изменения кода и тестов
2. Проходит CI
3. Происходит код ревью
4. Код попадает в прод


А ПОТОМ
38
СЛУЧАЮТСЯ БАГИ
imgflip.com



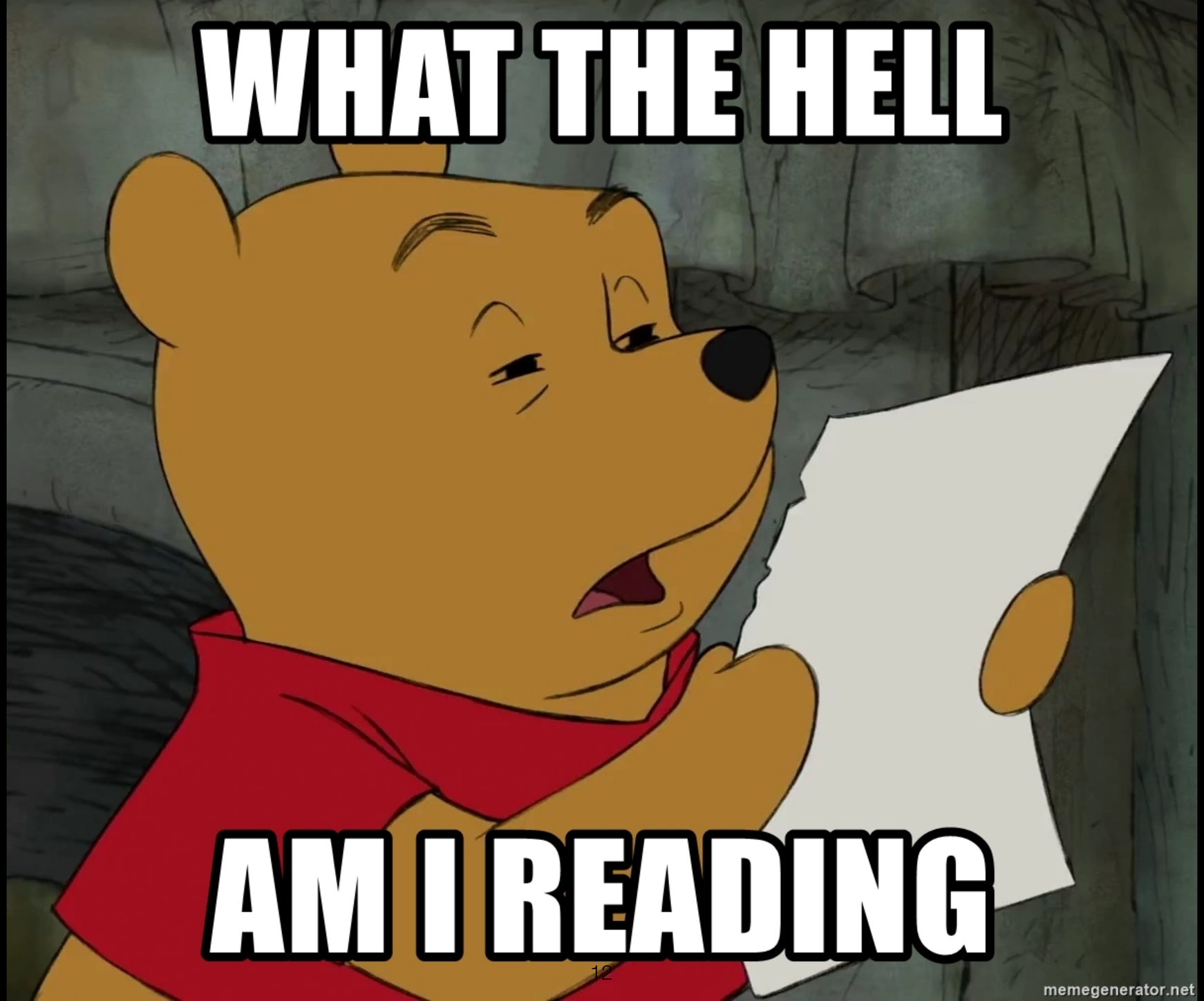
Потому что наши тесты ничего не проверяют!

Выход логичен

- Будем строже ревьюить!
- Мы напишем больше тестов!
- Мы повысим покрытие!

Или нет? 

Будем строже ревьюить


WHAT THE HELL
AM I READING
memegenerator.net

tests/test_bot_app/test_tasks/test_queue_verification_email.py del...

0 24

```
1  - # -*- coding: utf-8 -*-
2  -
3  - import pytest
4  -
5  - from server.bot_app.models import ApplicationUser,
     TelegramSession
6  - from server.bot_app.tasks import queue_verification_email
7  -
8  -
9  - @pytest.mark.django_db
10 - def test_queue_verification_email_normal(mailoutbox):
11 -     user = ApplicationUser.objects.create(
12 -         email='mail@sobolevn.me',
13 -     )
14 -
15 -     instance = TelegramSession.objects.create(
16 -         user=user,
17 -         uid=12345,
18 -         verification_code=123,
19 -     )
20 -
21 -     queue_verification_email(instance.uid)
22 -
23 -     assert len(mailoutbox) == 1
24 -     assert mailoutbox[0].to == [user.email]
```

Нужно что-то еще!

Больше тестов =
больше кода

Больше кода =
больше багов

Больше тестов =
больше дубликатов

Больше тестов = затык на CI

Нужно меньше,
но лучше

Повысим покрытие

```
--cov-fail-under=80
```

```
--cov-fail-under=90
```

```
--cov-fail-under=100
```

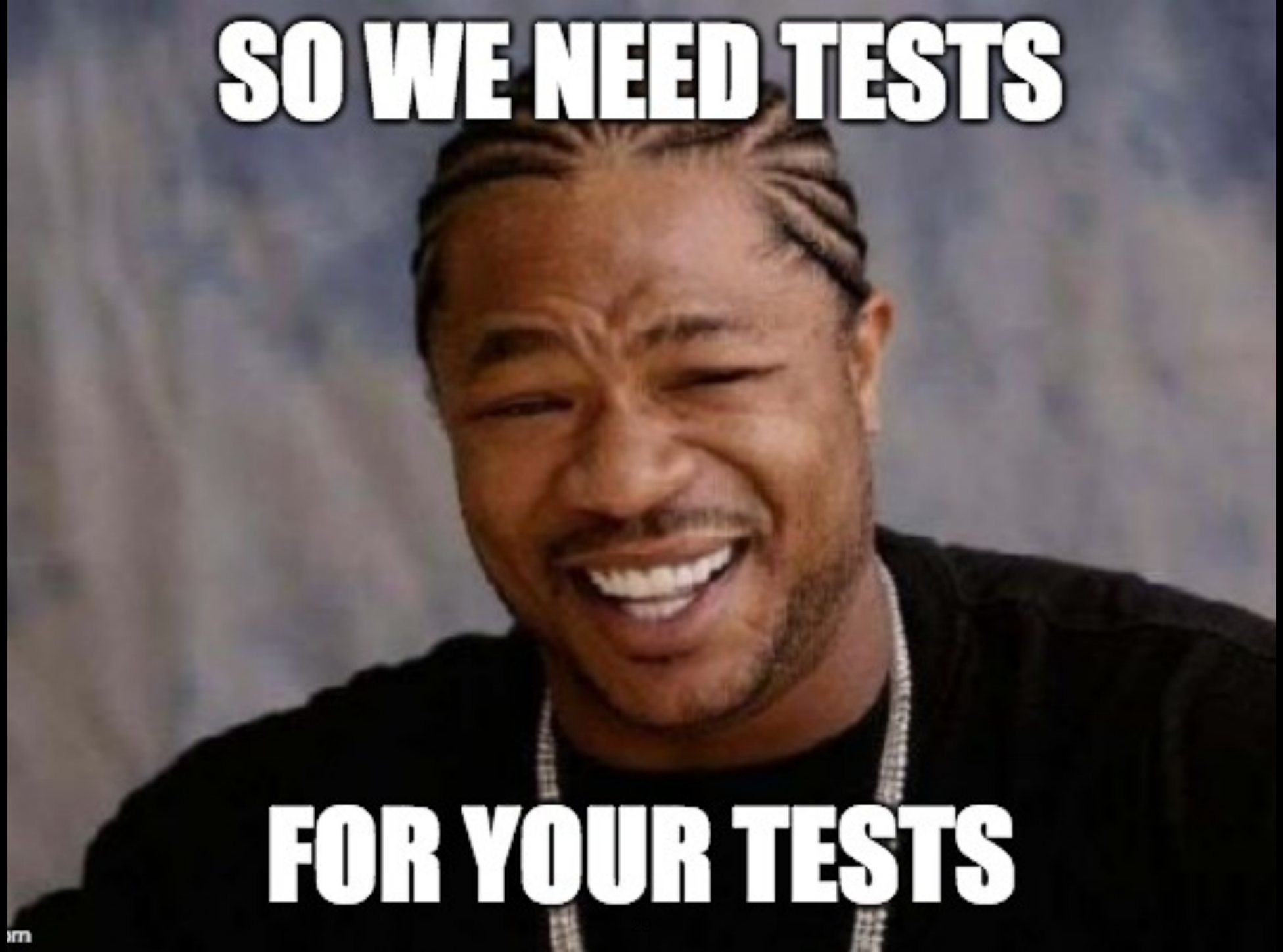
```
1
2 def negate(first: float):
3     """Return the negated number."""
4     return 0 - first
```

```python
@pytest.mark.parametrize('given, expected', [
    (-1, 1),
    (0, 0),
    (0.5, -0.5),
])
def test_negate(given, expected):
    function_result = negate(given)

    # TODO: uncomment this line:
    # assert function_result == expected
```

Покрытия мало,
нужны сами проверки

Тестируете ли вы ваши тесты?


SO WE NEED TESTS
FOR YOUR TESTS

План

- О ментальной модели
- О внутреннем устройстве
- Какие проблемы найдем?
- Как внедрить?

Как тестировать тесты?

Дано

- Сотни тысяч строк кода
- Десятки тысяч тестов
- Одна новая фича

Software can be chaotic, but we make it work

Expert

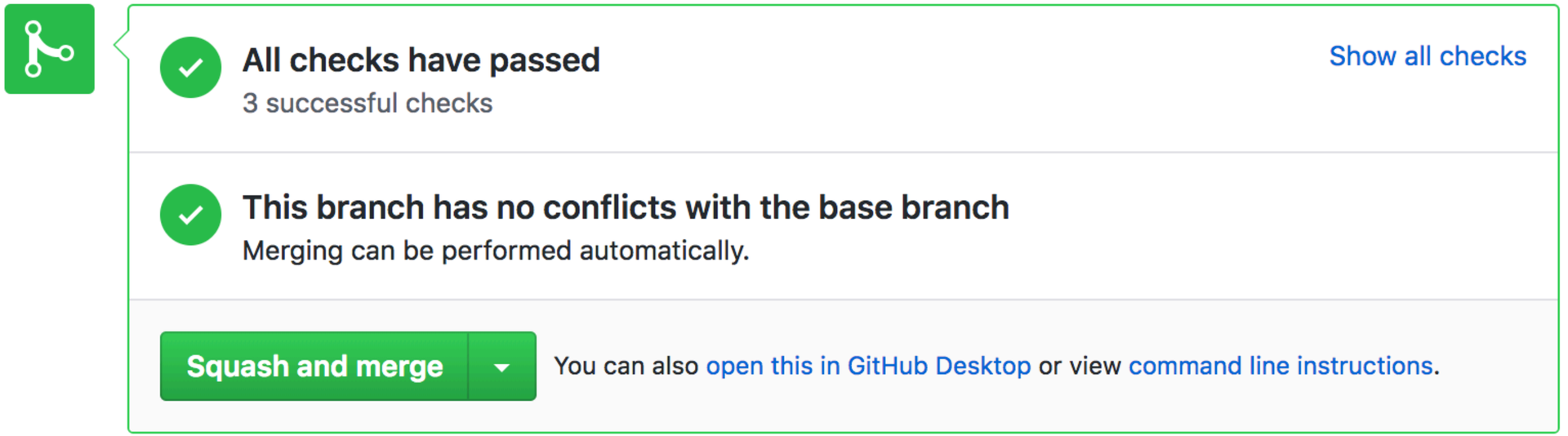
Trying Stuff Until it Works

O RLY?

The Practical Developer
@ThePracticalDev

Меняем код в "случайных" местах

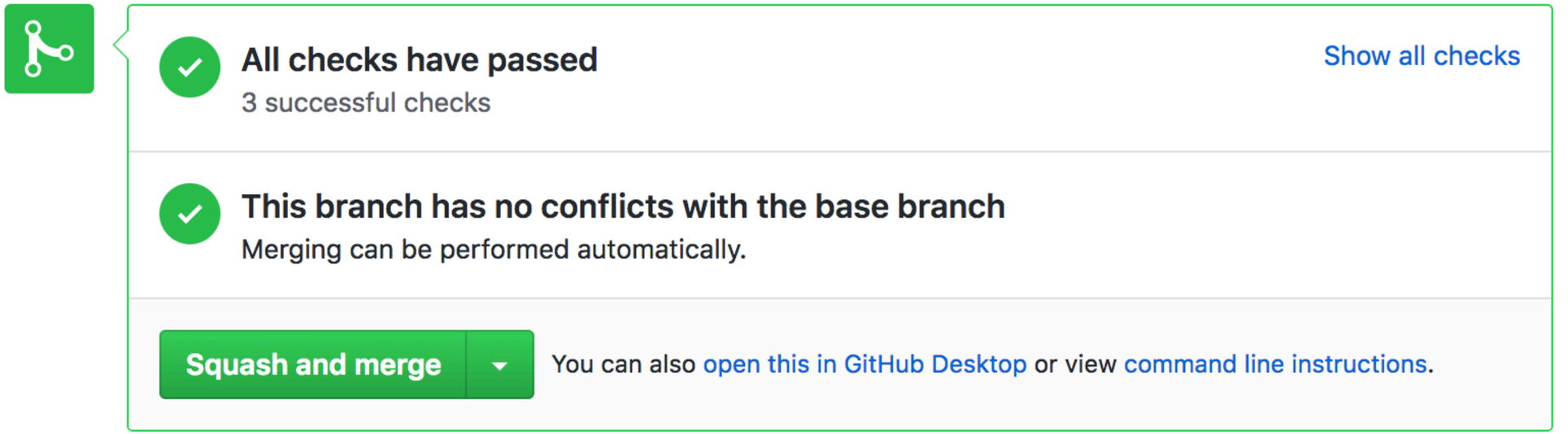
```
    length = len(array)
    for first in range(length - 1):
+++     swapped = False
        for second in range(length - 1 - first):
            if array[second] > array[second + 1]:
+++             swapped = True
                array[second], array[second + 1] = (
                    array[second + 1],
                    array[second],
                )
+++     if not swapped:
+++         break
    return array
```


All checks have passed
3 successful checks
Show all checks
This branch has no conflicts with the base branch
Merging can be performed automatically.
Squash and merge
You can also open this in GitHub Desktop or view command line instructions.

Значит все хорошо!

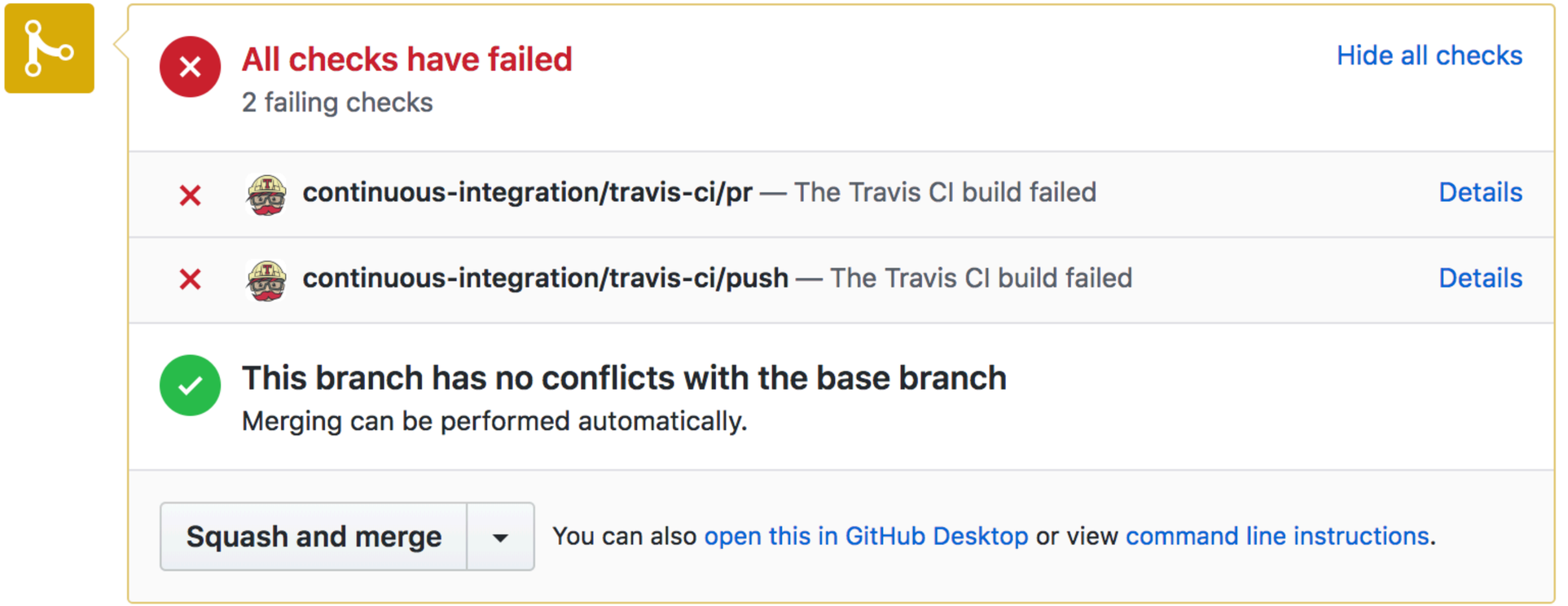
А если ошибка? 🤔

```
        length = len(array)
        for first in range(length - 1):
+++         raise ValueError('Should fail!')
```


All checks have passed
3 successful checks
Show all checks
This branch has no conflicts with the base branch
Merging can be performed automatically.
Squash and merge
You can also open this in GitHub Desktop or view command line instructions.


WELL, I'M
SCREWED
memegenerator.net

Нужно поправить тесты!


All checks have failed
2 failing checks
Hide all checks
continuous-integration/travis-ci/pr — The Travis CI build failed
Details
continuous-integration/travis-ci/push — The Travis CI build failed
Details
This branch has no conflicts with the base branch
Merging can be performed automatically.
Squash and merge
You can also open this in GitHub Desktop or view command line instructions.

Когда тесты падают
на сломанном коде,
то мы довольны


А ДАВАЙТЕ
ВСЕ СЛОМАЕМ

Методика, как ломать код

How to actually learn any new programming concept

Essential

Changing Stuff and Seeing What Happens

O RLY?

@ThePracticalDev

Будем вносить небольшие отклонения

Создавать мутантов

```
---  if oversize > 0:
+++  if oversize > 1:
         print('{0} exceeds {1} limit by {2}'.format(
             arguments.image,
             arguments.size,
             format_size(oversize, binary=True),
         ))
```

```
      if oversize > 0:
---       print('{0} exceeds {1} limit by {2}'.format(
+++       print('XX{0} xx {1} xxx {2}XX'.format(
              arguments.image,
              arguments.size,
              format_size(oversize, binary=True),
          ))
```

```
        if oversize > 0:
            print('{0} exceeds {1} limit by {2}'.format(
                arguments.image,
                arguments.size,
---             format_size(oversize, binary=True),
+++             format_size(oversize, binary=False),
            ))
```

После каждого изменения запускаем тесты

Варианты развития событий

- Тесты упадут и убьют мутанта
- Таймаут
- WTF
- Тесты выполнятся успешно и пропустят мутанта

3/3 🎉 1 ⏰ 0 🤔 0 😕 2

Пример

Исходник

```
def is_bigger(first, second) -> bool:
    return first > second


def test_is_bigger():
    assert is_bigger(2, 3) is True
    assert is_bigger(2, 2) is False
    assert is_bigger(2, 1) is False
```

Запускаем

$ mutmut run

» mutmut run

- Mutation testing starting -

Mutants are written to the cache in the .mutmut-cache
directory. Print found mutants with `mutmut results`.

Legend for output:
🎉 Killed mutants. The goal is for everything to end up in this bucket.
⏰ Timeout. Test suite took 10 times as long as the baseline so were killed.
🤔 Suspicious. Tests took a long time, but not long enough to be fatal.
🙁 Survived. This means your tests needs to be expanded.

1. Using cached time for baseline tests, to run baseline again delete the cache file

2. Checking mutants
⠸ 1/1 🎉 1 ⏰ 0 🤔 0 🙁 0

Смотрим

```
$ mutmut show 1
```

Мутант

```
def is_bigger(first, second) -> bool:
--- return first > second
+++ return first >= second
```

```
def test_is_bigger():
    assert is_bigger(2, 3) is True
    assert is_bigger(2, 2) is False
    assert is_bigger(2, 1) is False
```

Тест упал!

1/1 🎉 1 ⏰ 0 🤔 0 😕 0

Как заменять кусочки кода?

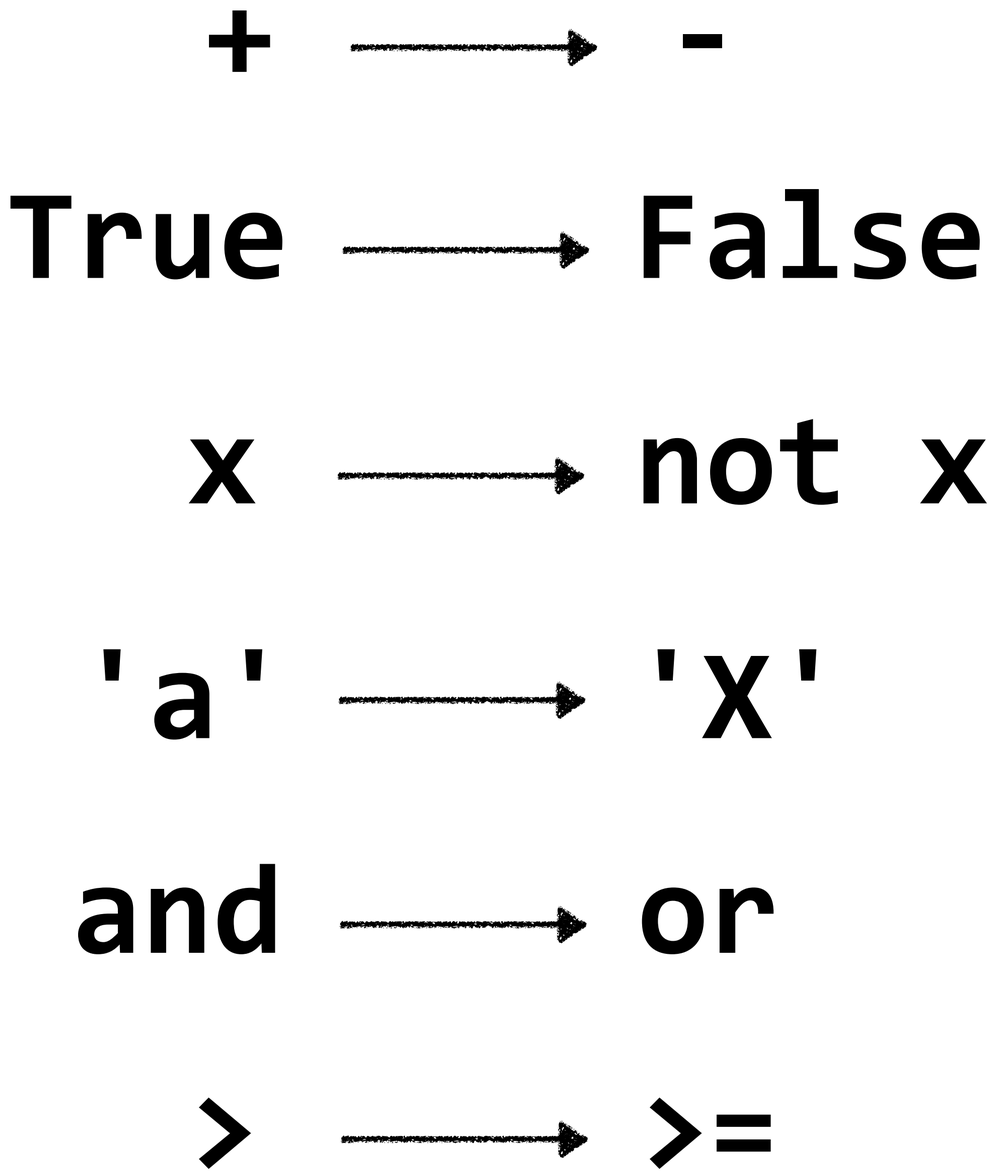
Abstract Syntax Tree

```
def sum_two(a, b):
    return a + b
```

```
Module(
   body=[
      FunctionDef(
         name='sum_two',
         args=arguments([arg(arg='a'), arg(arg='b')])
         body=[
            Return(
               value=BinOp(
                  left=Name(id='a'),
                  op=Add(),
                  right=Name(id='b'),
               ),

         ...
```

```
def sum_two(a, b):
--- return a + b
+++ return a - b
```


+ → -
True → False
x → not x
'a' → 'X'
and → or
> → >=

- AOD - arithmetic operator deletion
- AOR - arithmetic operator replacement
- ASR - assignment operator replacement
- BCR - break continue replacement
- COD - conditional operator deletion
- COI - conditional operator insertion
- CRP - constant replacement
- DDL - decorator deletion
- EHD - exception handler deletion
- EXS - exception swallowing
- IHD - hiding variable deletion
- IOD - overriding method deletion
- IOP - overridden method calling position change
- LCR - logical connector replacement
- LOD - logical operator deletion
- LOR - logical operator replacement
- ROR - relational operator replacement
- SCD - super calling deletion
- SCI - super calling insert
- SIR - slice index remove


РАССМОТРИМ
ИНСТРУМЕНТЫ

Что есть для Python?

- CosmicRay
- MutPy
- mutmut

	Интеграция с pytest	Отчеты	Работает
CosmicRay	✗	✓	✓
MutPy	✗	✓	✗
mutmut	✓	✓	✓

Еще сыровато 

Какие ошибки можно найти?

```python
def add(first: float, second: float):
    return first + second


def test_add():
    assert add(0, 0) == 0
    assert add(2, 2) == 4
```

Плохие данные

```
def add(first: float, second: float):
--- return first + second
+++ return first * second


def test_add():
    assert add(0, 0) == 0
    assert add(2, 2) == 4
```

Мутации помогут найти плохие данные в сложных случаях

```
app = Flask(__name__)


@app.route('/<int:index>')
def hello(index: int):
    return 'Hello, world! {0} faith in you.'.format(
        1 * index,
    )
```

```python
def test_hello_view(flask_client):
    response = flask_client.get('/0')

    assert response.status_code == 200
    assert b'world' in response.data # localization
    assert b'0' in response.data
```

```
@app.route('/<int:index>')
def hello(index: int):
    return 'Hello, world! {0} faith in you.'.format(
        1 / index,  # new formula!
    )
```

```
@app.errorhandler(Exception)
def log_to_sentry_and_show_sorry_page(exception):
    return 'S0rry, world :(', 200
```

```python
def test_hello_view(flask_client):
    response = flask_client.get('/0')

    assert response.status_code == 200
    assert b'world' in response.data
    assert b'0' in response.data
```

Мутации помогают вам искать плохие тесты в больших системах

```
WRONG_LETTERS = [
    'A',
    'B',
    'C',
]

def is_wrong_letter(letter: str) -> bool:
    return letter in WRONG_LETTERS
```

```
from source import WRONG_LETTERS, is_wrong_letter


@pytest.mark.parametrize('letter', WRONG_LETTERS)
def test_is_wrong_letter(letter):
    assert is_wrong_letter(letter) is True
```

```
WRONG_LETTERS = [
    'X',
    'Y',
    'q',
]

def is_wrong_letter(letter: str) -> bool:
    return letter in WRONG_LETTERS
```

```
from source import WRONG_LETTERS, is_wrong_letter


@pytest.mark.parametrize('letter', WRONG_LETTERS)
def test_is_wrong_letter(letter):
    assert is_wrong_letter(letter) is True
```

Мутации делают связанные данные явными

CTLR-C + CTRL-V

```
from source import is_wrong_letter


@pytest.mark.parametrize('letter', ['a'])
def test_is_wrong_letter(letter):
    assert is_wrong_letter(letter) is True
```

```
def save_subscription(form):
    subscription = form.save()
    queue_welcome_email.delay(subscription.id)
    return subscription
```

```
def test_save_subscription(form):
    instance = save_subscription(form)

    assert instance.id > 0
    assert instance.name == form.data['name']
```

```
def save_subscription(form):
    subscription = form.save()
    --- queue_welcome_email.delay(subscription.id)
    +++ queue_welcome_email.delay(None)
    return subscription
```

Частичные тесты

Мутации покажут непокрытые side-effects

```
def test_save_subscription(form, redis):
    instance = save_subscription(form)

    assert instance.id > 0
    assert instance.name == form.data['name']
    assert redis.get(queue(instance))
```

```
CELERY_BROKER_URL = 'redis://{host}:{port}'.format(
    host=config('HOST', default='localhost'),
)
```

```
CELERY_BROKER_URL = 'redis://{host}:{port}'.format(
    --- host=config('HOST', default='localhost'),
    +++ host=config('HOST', default='XXXX'),
)
```

 1 ||  1

Мутации находят edge-cases

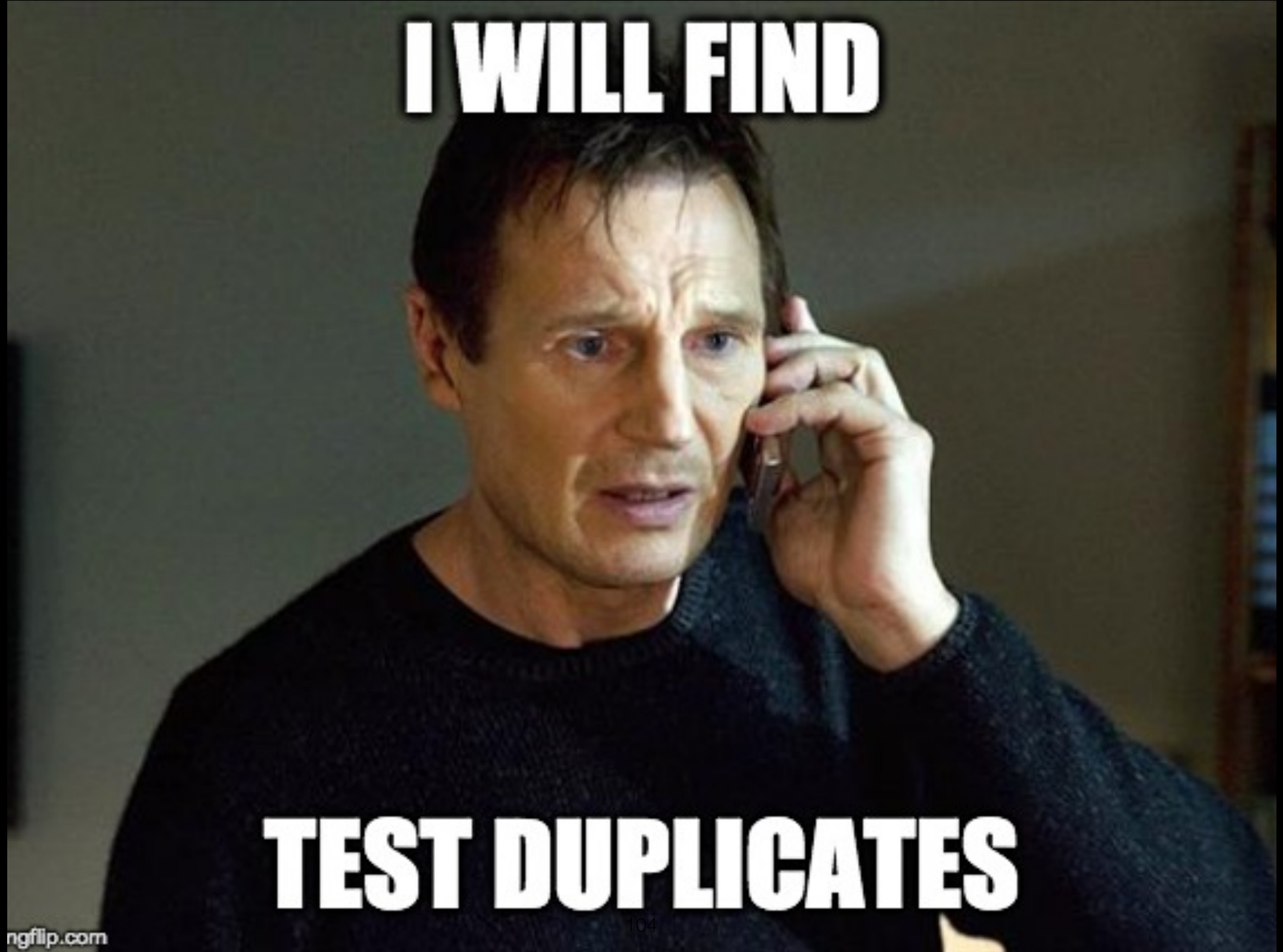
```
def test_user_create():
    user = User.objects.create(username='some')

    assert user.id is not None


def test_user_can_be_saved():
    user = User()
    user.username = 'some'
    user.save()

    assert user.id > 0
```

Нужно меньше,
но лучше!


I WILL FIND
TEST DUPLICATES
ngflip.com

--post-mutation

Удаляем лишние тесты

- Запускаем мутации
- Записываем названия тестов, которые упали
- Тесты, которые не упали, очевидно, не нужны

```
failing_regex = re.compile(r'test_... F')
failing_tests = failing_regex.search(tests_output)

for test in failing_tests:
    ...
```

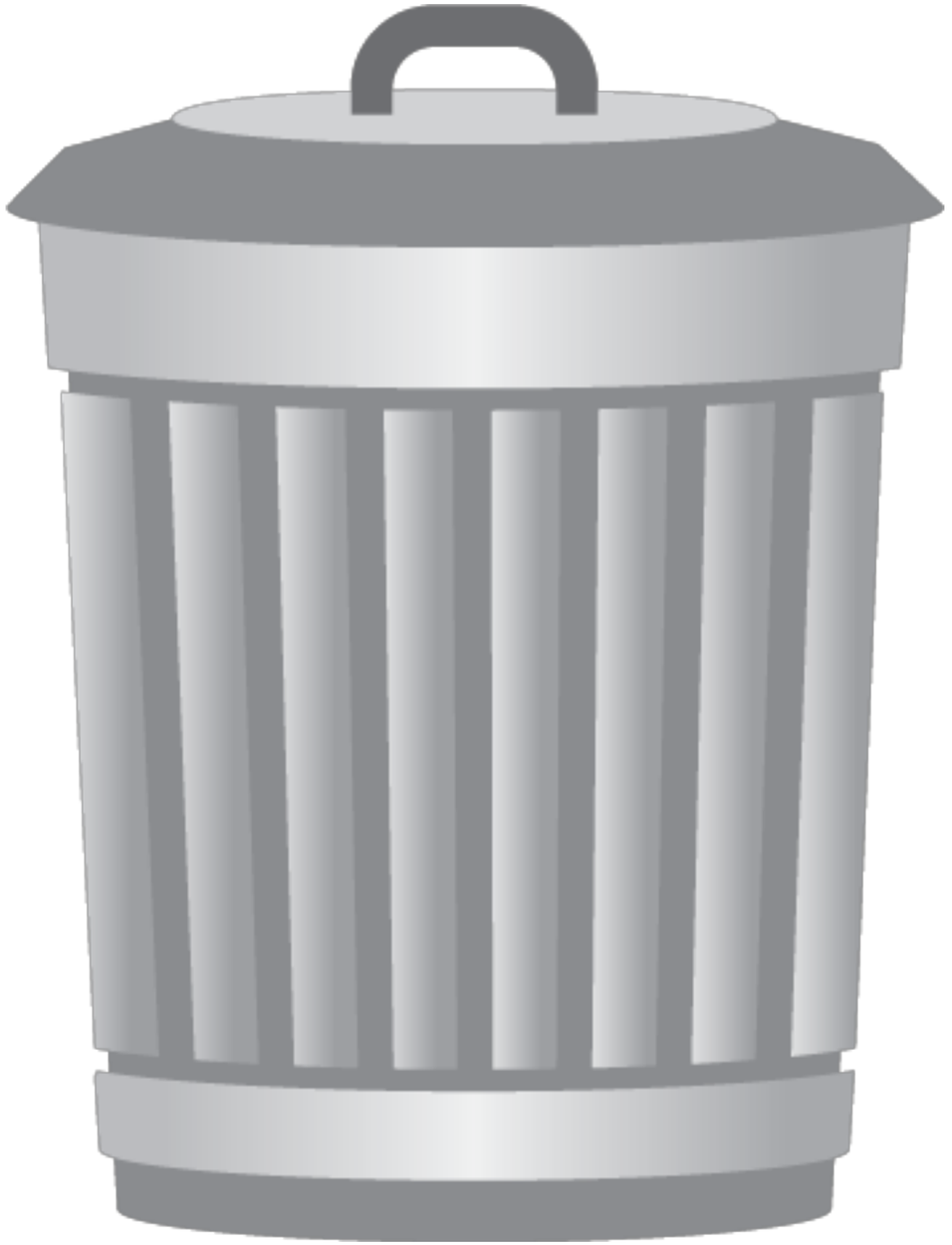
```
failing_regex = re.compile(r'test_... F')
failing_tests = failing_regex.search(tests_output)

for test in failing_tests:
    ...
```

И много других!

Как круто!
Давайте брать в прод!

29/29 🎉 19 ⏰ 0 🤔 2 😕 8



```
def parse_args():
    parser = argparse.ArgumentParser(
---         description='Keep your images small',
+++         description='XXXX',
    )
```

Игнорируем строки

```
# pragma: no mutate
```

```
exit_code = 0  # pragma: no mutate
if oversize > 0:  # pragma: no mutate
    print('{0} exceeds limit'.format(  # pragma: no mutate
        format_size(binary=True),  # pragma: no mutate
    ))
    exit_code = 1  # pragma: no mutate
```

Запускаем выборочно .patch

```
--- a/foo.py
+++ b/foo.py
@@ -1,5 +1,5 @@

 def is_lower(first, second):
     return first < second

-print(is_lower(1, 2))
+print(is_lower(1, 1))
```

Запускаем точные куски кода

1. Запускаем конкретный тест

2. Собираем coverage

3. Мутируем только нужный код

```
--path-to-mutate=
src/module.py
```

FEATURE PLAN:
стратегии или выбор,
что мутировать

TODO: одинаковые мутанты

```
x > 0 == 0 < x
```

Что со скоростью?

Очень долго!

- Тесты выполняются за 1 секунду
- 1000 возможных мутаций
- 1 секунда * 1000 мутаций = 16.6 минут

16 минут

`--use-coverage`

Отключаем плагины

- Coverage
- Random ordering
- Дополнительные проверки

15 минут

```
--tb=no --quiet
```

10 минут

--exitfirst

6 минут

- Запускаем наши тесты один раз
- Генерируем coverage
- Запускаем мутационные

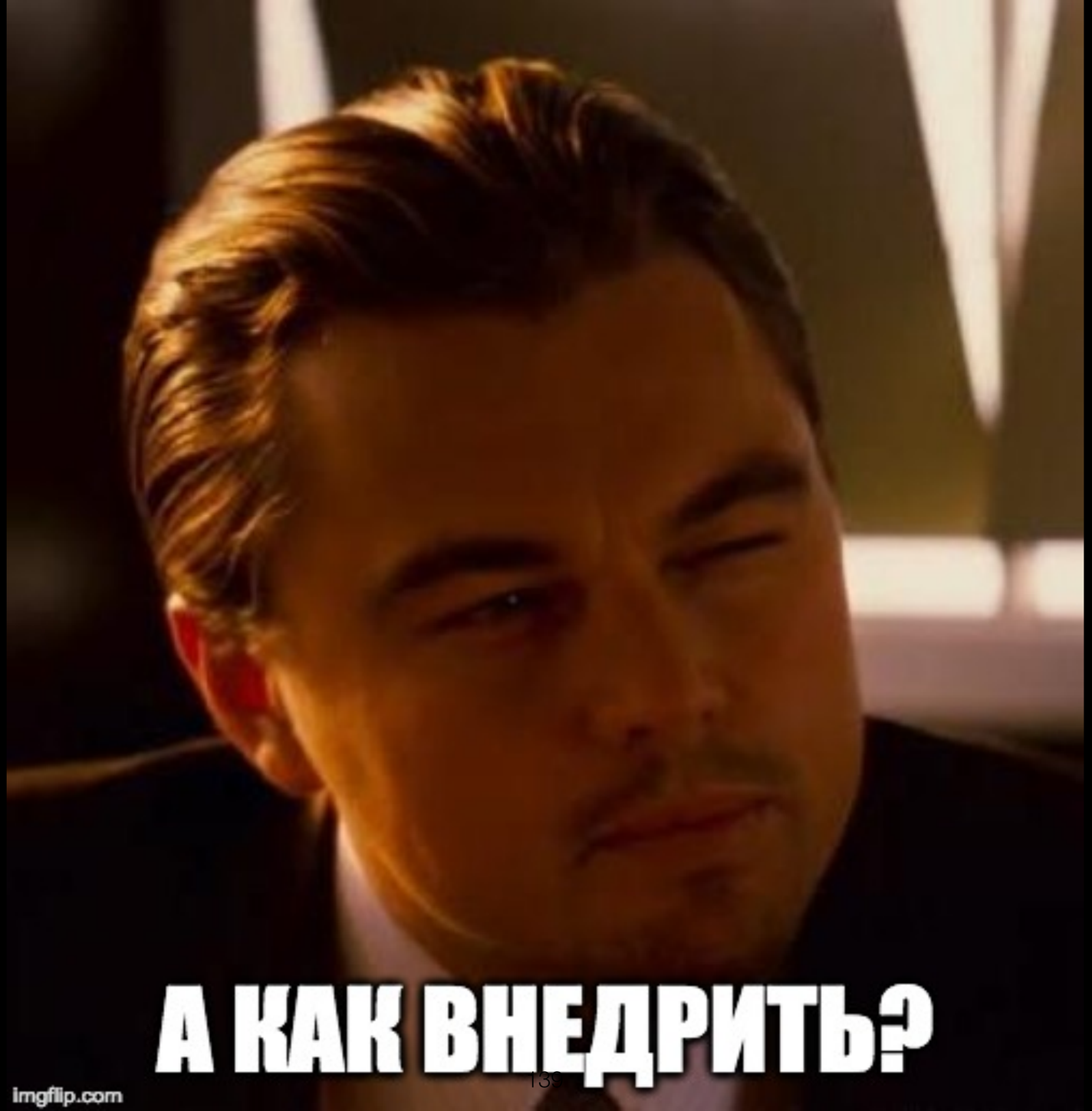
--testmon

4 минуты

16 минут ->
4 минуты

```
mutmut run --use-coverage -s
```

```
[mutmut]
paths_to_mutate=src/
backup=False
runner=pytest -x -q --tb=no \
  --testmon -o addopts=""
tests_dir=tests/
```


А КАК ВНЕДРИТЬ?
imgflip.com

Пишем тест - сразу тестируем

```
[mutmut]
paths_to_mutate=src/wip.py
runner=pytest test_src/test_wip.py
```

Отчеты

```
---

mutations:
  stage: test
  allow_failure: true
  image: wemakeservices/mutmut:latest
  script:
    - mutmut run
    - mutmut html
  when: manual
```

lib/tri/declarative/__init__.py

Killed 338 out of 473 mutants

Survived

Survived mutation testing. These mutants show holes in your test suite.

Mutant 378

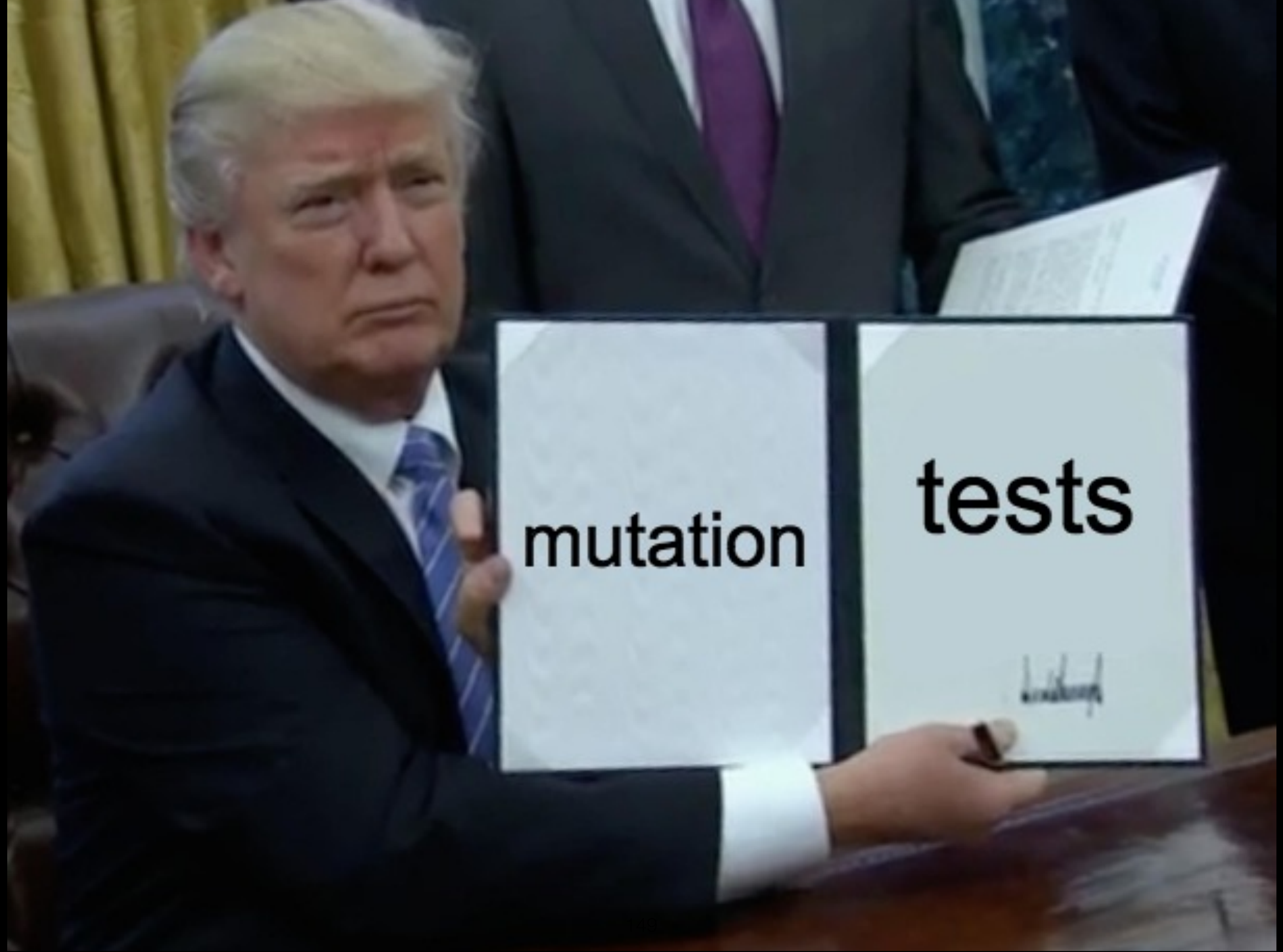
```
--- lib/tri/declarative/__init__.py
+++ lib/tri/declarative/__init__.py
@@ -764,7 +764,7 @@
      :param classes: list of classes to generate documentation for
      :param missing_objects: tuple of objects to count as missing markers, if applicable
      """
-     if missing_objects is None:
+     if missing_objects is not None:
          missing_objects = tuple()

      import re
```

--post-mutation

Поиски халявщиков

Поиски дубликатов


mutation
tests

Выводы

- Нашли кучу проблем, которые не находят наши обычные тесты
- Узнали способы внедрения
- Протестировали тесты!

github.com/boxed/mutmut

github.com/sobolevn/heisenbug-2019

Вопросы?

github.com/sobolevn

sobolevn.me