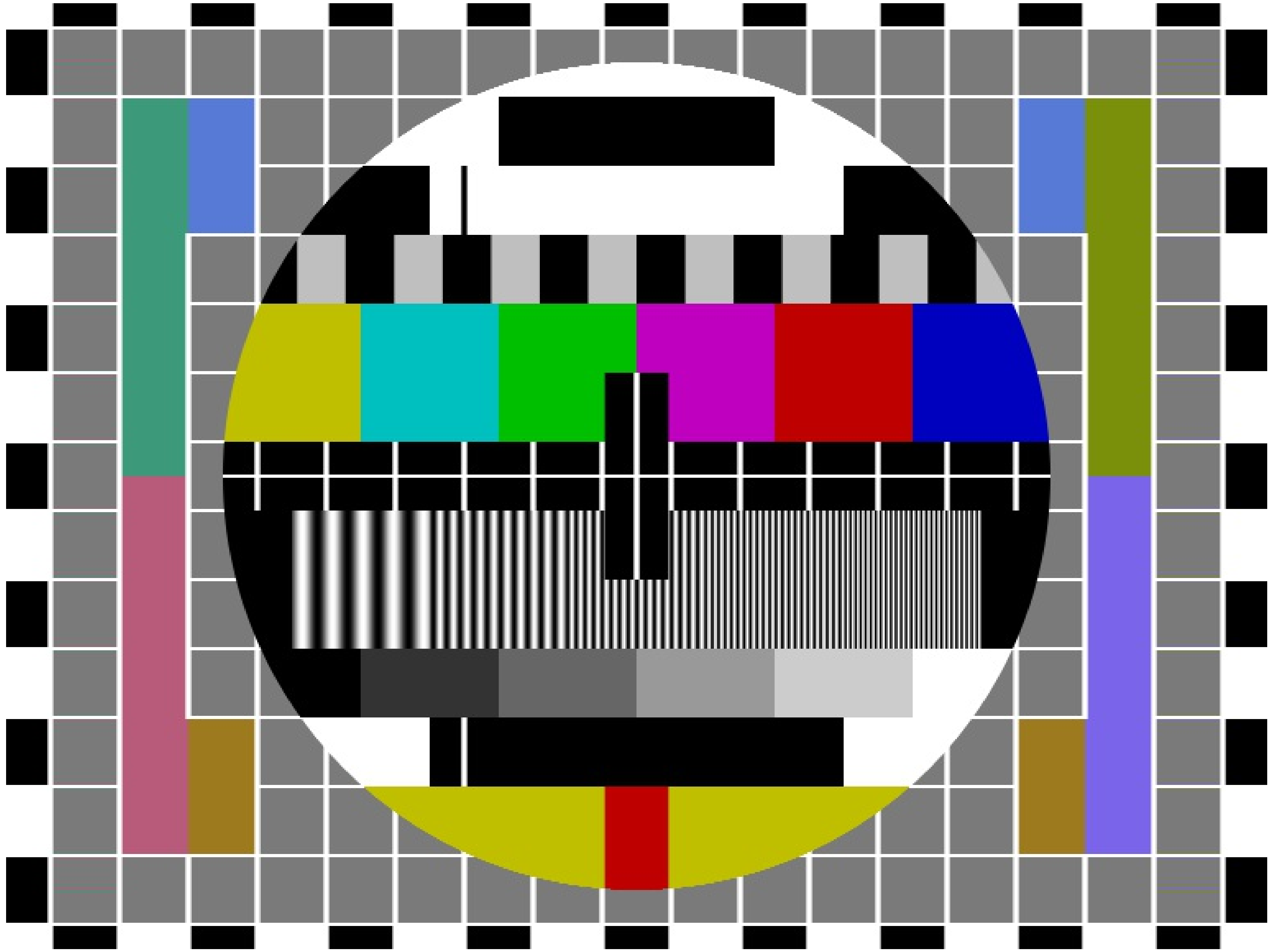
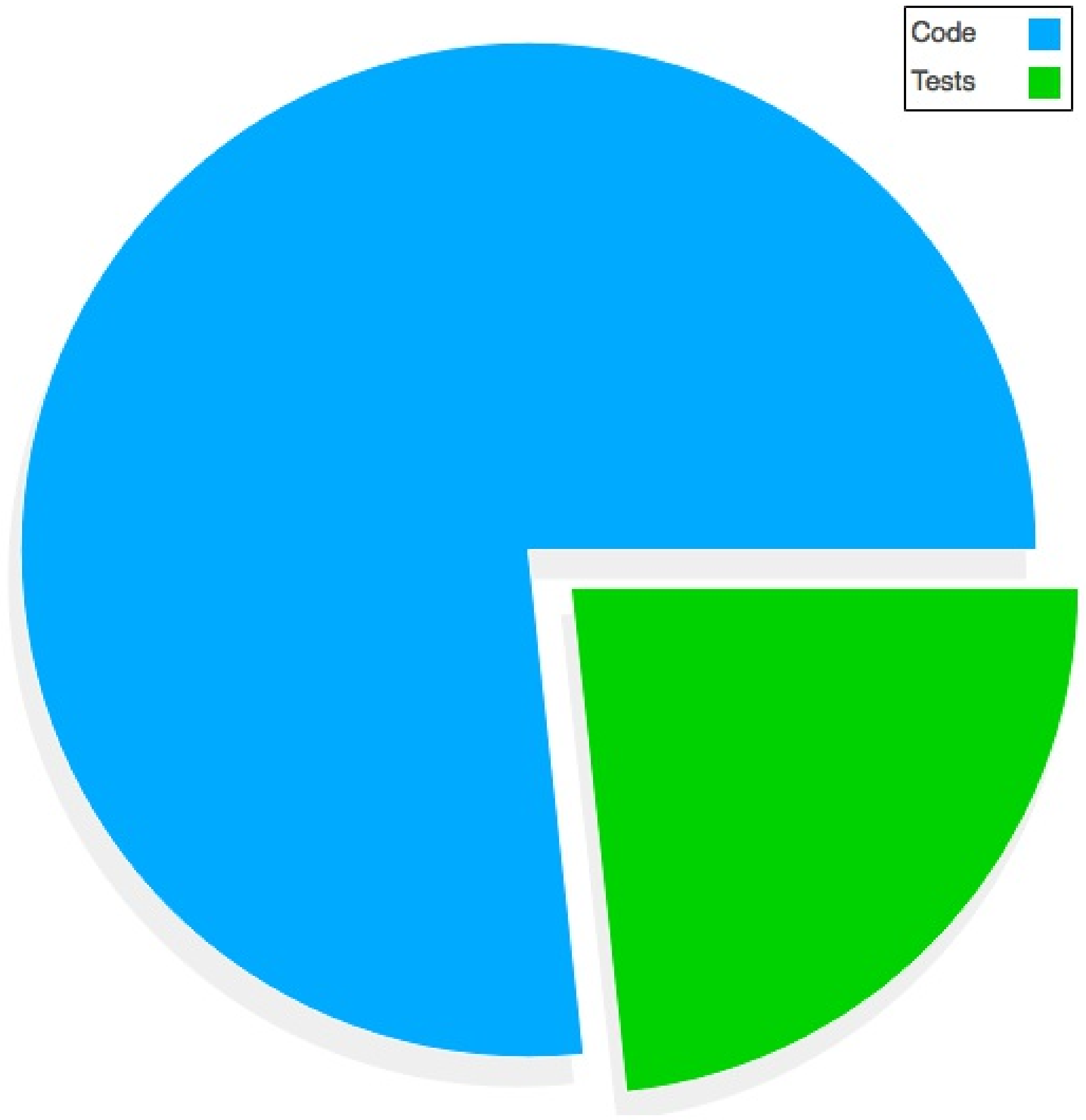


Sustainable way of testing your code

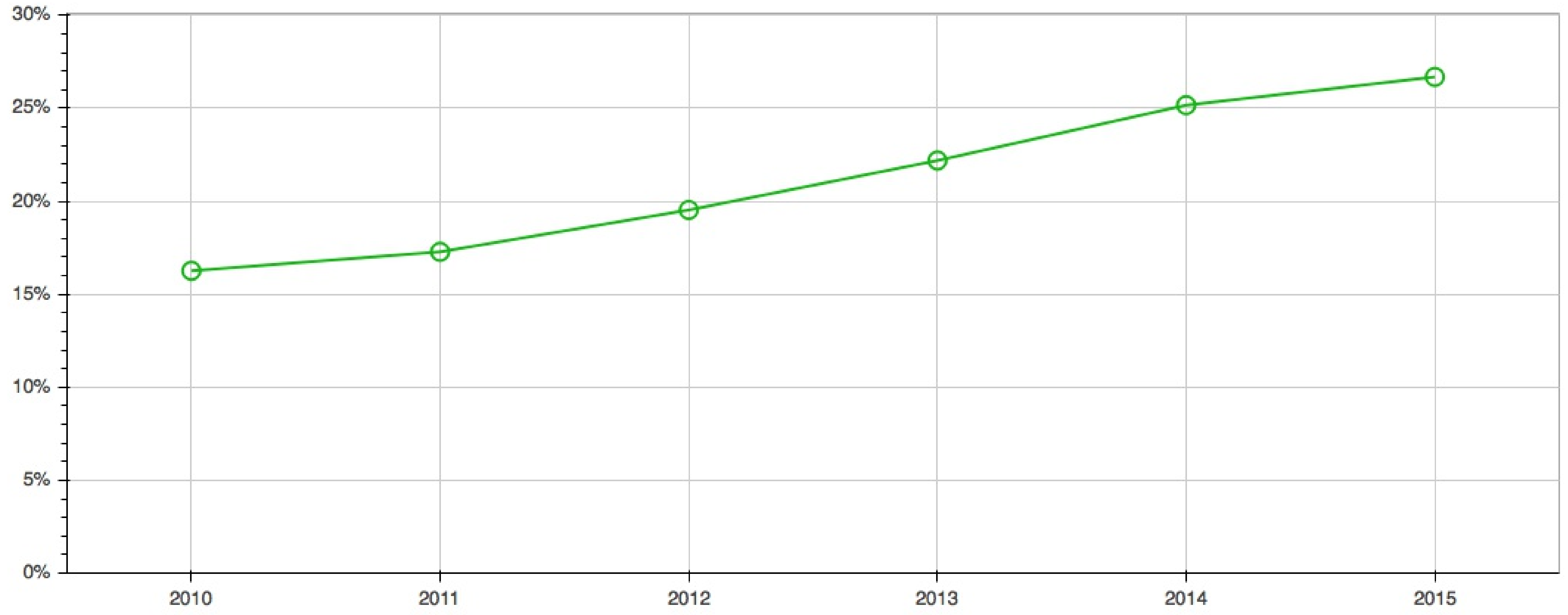
by *Eugene Amirov*

Teamlead at Scrapinghub

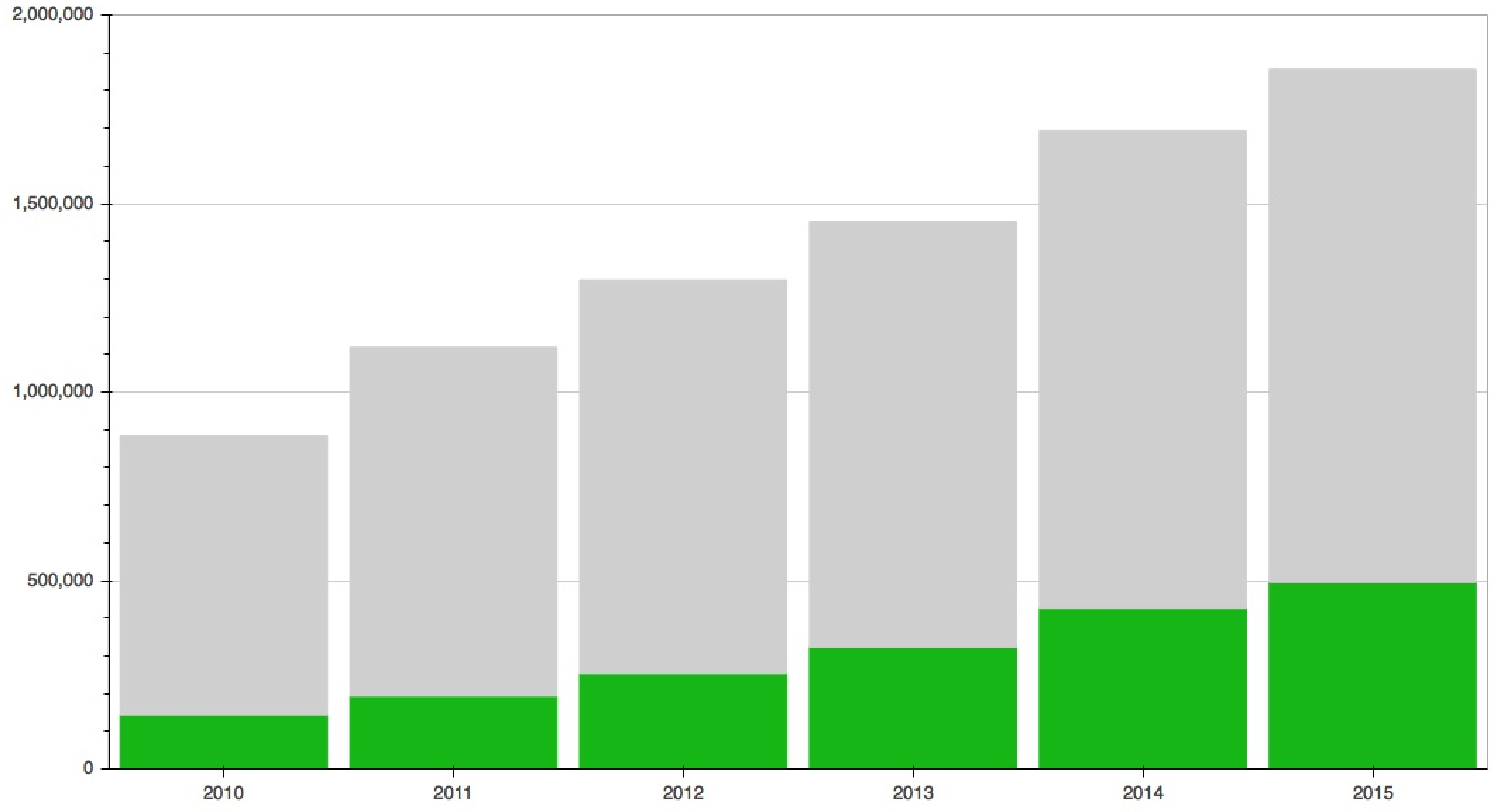
For top 100 most starred Python projects on GitHub the percentage of testing code is a little bit more that 23%.



Percentage of tests



Lines of code in tests



```python
def test_adding_same_dsn_multiple_times(self):
    logger = Mock()
    logger.handlers = []
    logger.addHandler = Mock(wraps=self.logger.handlers.append)
    dsn = 'http://user:pass@test/1'

    handler1 = register_sentry_logging(dsn, logger)
    self.assertIn(handler1, logger.handlers)

    handler2 = register_sentry_logging(dsn, logger)
    self.assertIsNone(handler2)

    self.assertEqual(len(logger.handlers), 1)
```

- **Initial condition**: We have some prepared *environment* which we know to be true

- **Initial condition**: We have some prepared *environment* which we know to be true
- **The event**: In this *environment* we execute some *action* that we want to test

- **Initial condition**: We have some prepared *environment* which we know to be true
- **The event**: In this *environment* we execute some *action* that we want to test
- **Expected outcome**: We *expect* some particular results of this *action*

Given a customer previously bought a black sweater from me
And I currently have three black sweaters left in stock
When he returns the sweater for a refund
Then I should have four black sweaters in stock

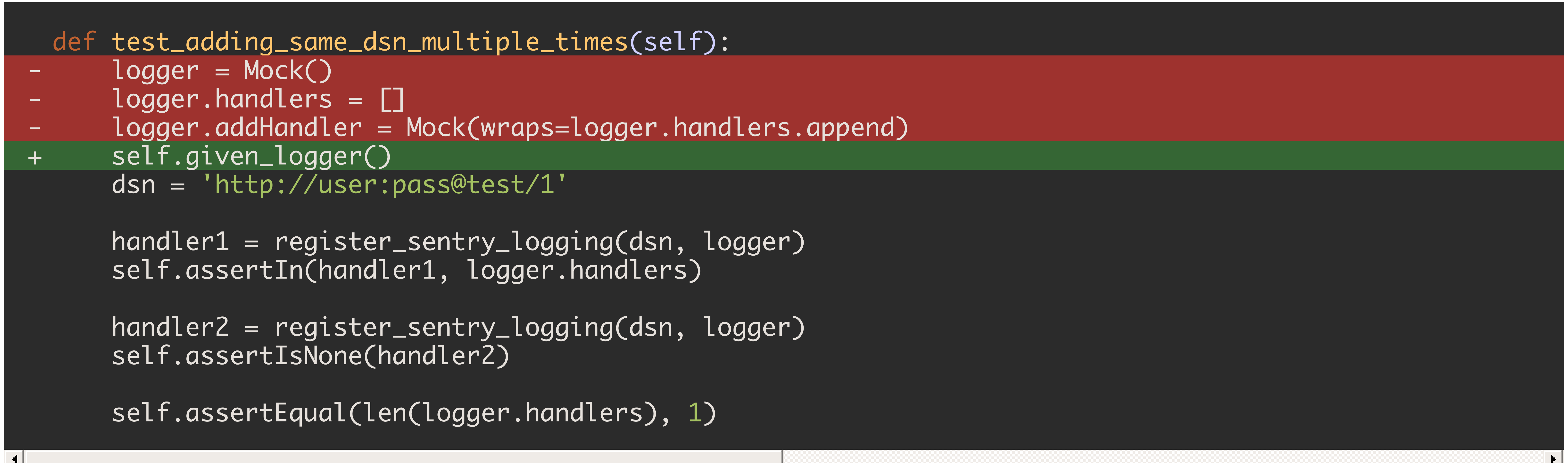
```python
def test_adding_same_dsn_multiple_times(self):
    logger = Mock()
    logger.handlers = []
    logger.addHandler = Mock(wraps=self.logger.handlers.append)
    dsn = 'http://user:pass@test/1'

    handler1 = register_sentry_logging(dsn, logger)
    self.assertIn(handler1, logger.handlers)

    handler2 = register_sentry_logging(dsn, logger)
    self.assertIsNone(handler2)

    self.assertEqual(len(logger.handlers), 1)
```

environment



```diff
  def test_adding_same_dsn_multiple_times(self):
-     logger = Mock()
-     logger.handlers = []
-     logger.addHandler = Mock(wraps=logger.handlers.append)
+     self.given_logger()
      dsn = 'http://user:pass@test/1'

      handler1 = register_sentry_logging(dsn, logger)
      self.assertIn(handler1, logger.handlers)

      handler2 = register_sentry_logging(dsn, logger)
      self.assertIsNone(handler2)

      self.assertEqual(len(logger.handlers), 1)
```

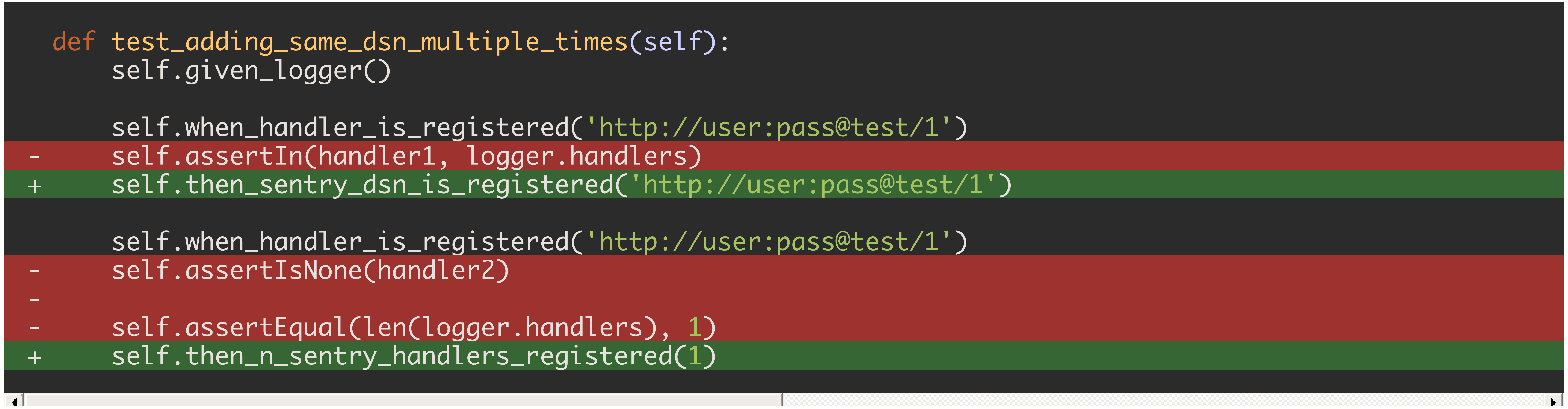
action

```diff
 def test_adding_same_dsn_multiple_times(self):
     self.given_logger()
-    dsn = 'http://user:pass@test/1'
-
-    handler1 = register_sentry_logging(dsn, logger)
+    self.when_handler_is_registered('http://user:pass@test/1')
     self.assertIn(handler1, logger.handlers)

-    handler2 = register_sentry_logging(dsn, logger)
+    self.when_handler_is_registered('http://user:pass@test/1')
     self.assertIsNone(handler2)

     self.assertEqual(len(logger.handlers), 1)
```

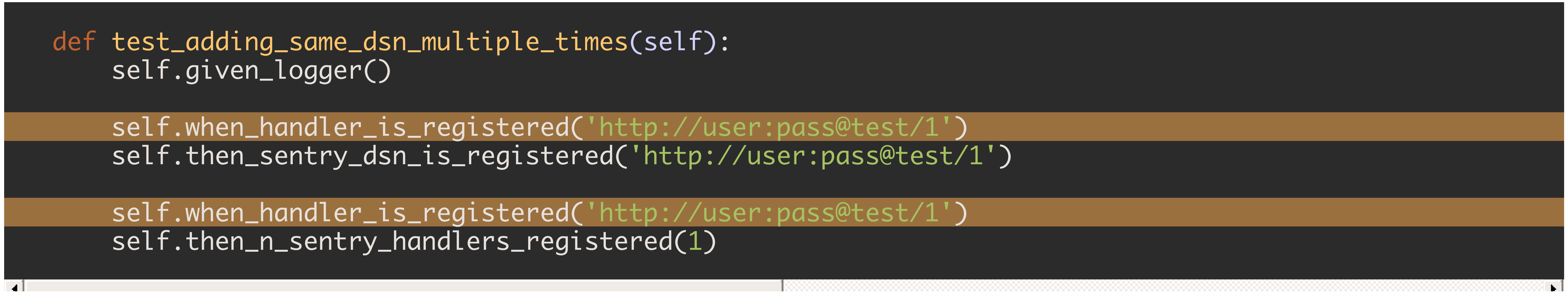
expectation



```
  def test_adding_same_dsn_multiple_times(self):
      self.given_logger()

      self.when_handler_is_registered('http://user:pass@test/1')
-     self.assertIn(handler1, logger.handlers)
+     self.then_sentry_dsn_is_registered('http://user:pass@test/1')

      self.when_handler_is_registered('http://user:pass@test/1')
-     self.assertIsNone(handler2)
-
-     self.assertEqual(len(logger.handlers), 1)
+     self.then_n_sentry_handlers_registered(1)
```



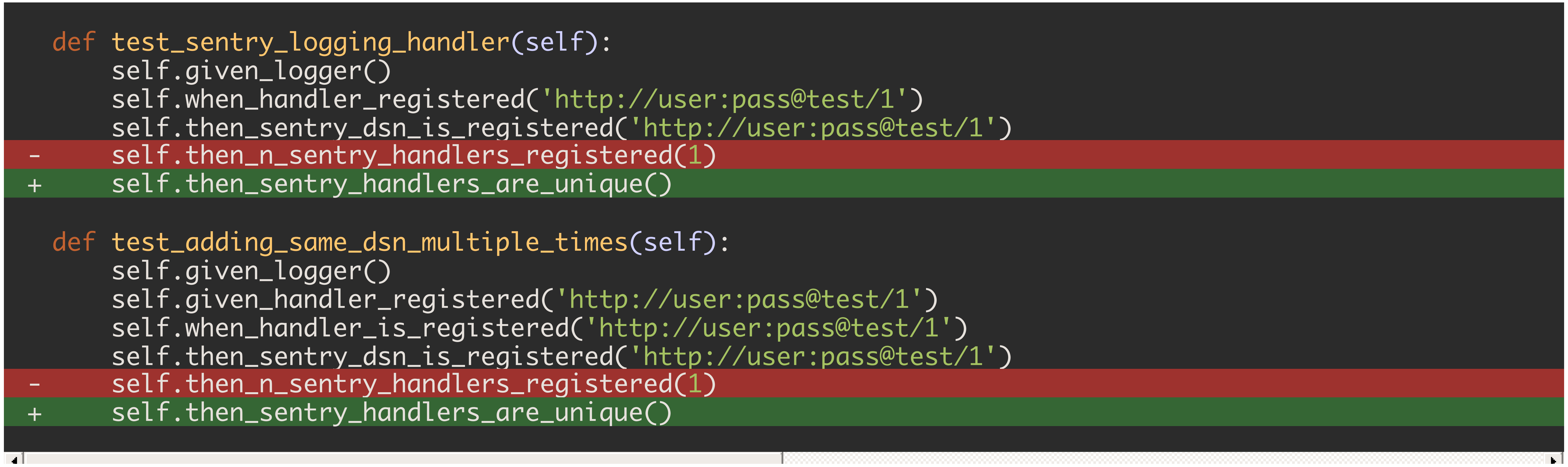
```
def test_adding_same_dsn_multiple_times(self):
    self.given_logger()

    self.when_handler_is_registered('http://user:pass@test/1')
    self.then_sentry_dsn_is_registered('http://user:pass@test/1')

    self.when_handler_is_registered('http://user:pass@test/1')
    self.then_n_sentry_handlers_registered(1)
```

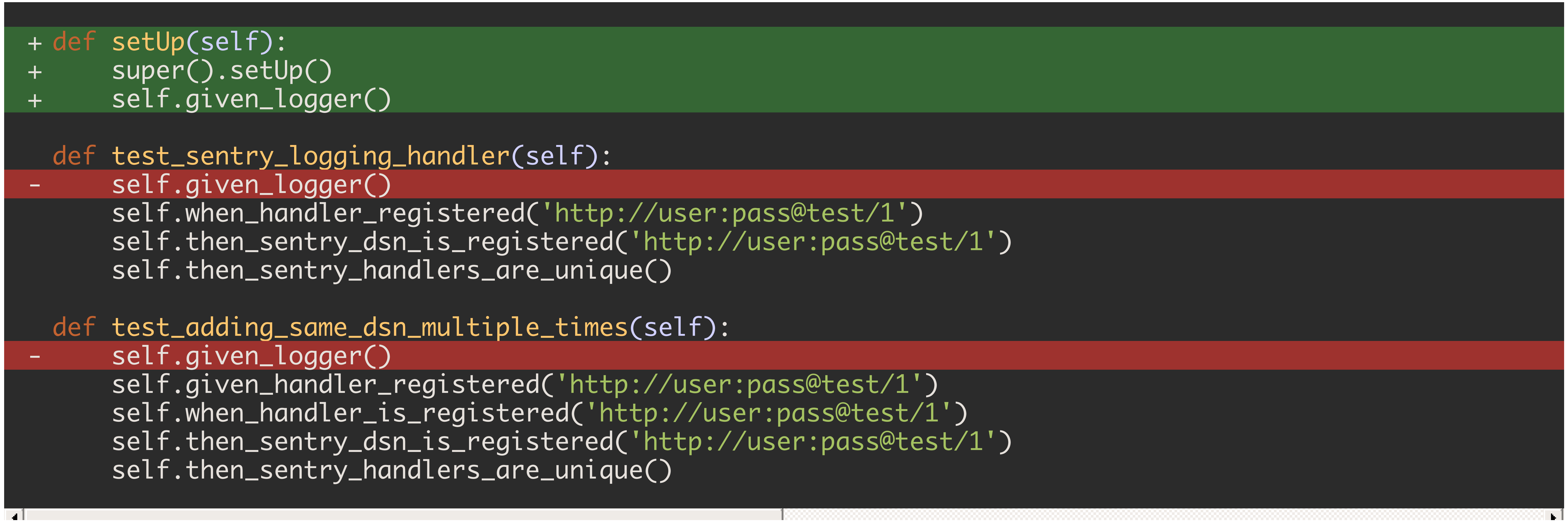
```python
def test_sentry_logging_handler(self):
    self.given_logger()
    self.when_handler_registered('http://user:pass@test/1')
    self.then_sentry_dsn_is_registered('http://user:pass@test/1')
    self.then_n_sentry_handlers_registered(1)

def test_adding_same_dsn_multiple_times(self):
    self.given_logger()
    self.given_handler_registered('http://user:pass@test/1')
    self.when_handler_is_registered('http://user:pass@test/1')
    self.then_sentry_dsn_is_registered('http://user:pass@test/1')
    self.then_n_sentry_handlers_registered(1)
```



```diff
   def test_sentry_logging_handler(self):
       self.given_logger()
       self.when_handler_registered('http://user:pass@test/1')
       self.then_sentry_dsn_is_registered('http://user:pass@test/1')
-      self.then_n_sentry_handlers_registered(1)
+      self.then_sentry_handlers_are_unique()

   def test_adding_same_dsn_multiple_times(self):
       self.given_logger()
       self.given_handler_registered('http://user:pass@test/1')
       self.when_handler_is_registered('http://user:pass@test/1')
       self.then_sentry_dsn_is_registered('http://user:pass@test/1')
-      self.then_n_sentry_handlers_registered(1)
+      self.then_sentry_handlers_are_unique()
```



```diff
+ def setUp(self):
+     super().setUp()
+     self.given_logger()

  def test_sentry_logging_handler(self):
-     self.given_logger()
      self.when_handler_registered('http://user:pass@test/1')
      self.then_sentry_dsn_is_registered('http://user:pass@test/1')
      self.then_sentry_handlers_are_unique()

  def test_adding_same_dsn_multiple_times(self):
-     self.given_logger()
      self.given_handler_registered('http://user:pass@test/1')
      self.when_handler_is_registered('http://user:pass@test/1')
      self.then_sentry_dsn_is_registered('http://user:pass@test/1')
      self.then_sentry_handlers_are_unique()
```

```python
class Aquarium(object):
    ...
    def is_habitable_by(self, fish):
        ...


class MarineAquarium(Aquarium):
    ...


class FreshwaterAquarium(Aquarium):
    ...


class DutchAquarium(FreshwaterAquarium):
    ...
```

```python
def test_habitability(self):
    self.given_aquarium(FreshwaterAquarium)

    self.when_checking_habitability_for(Guppi)
    self.then_aquarium_is_habitable()

    self.when_checking_habitability_for(Rasbora)
    self.then_aquarium_is_habitable()

    self.when_checking_habitability_for(Goldfish)
    self.then_aquarium_is_habitable()

    self.when_checking_habitability_for(Leopoldi)
    self.then_aquarium_is_habitable()
```

```python
def test_habitability(self):
    self.given_aquarium(FreshwaterAquarium)

    for fish in Guppi, Rasbora, Goldfish, Leopoldi:
        self.when_checking_habitability_for(fish)
        self.then_aquarium_is_habitable()
```

```python
def test_habitability(self):
    self.given_aquarium(FreshwaterAquarium)

    for fish in Guppi, Rasbora, Goldfish, Leopoldi:
        self.when_checking_habitability_for(fish)
        self.then_aquarium_is_habitable()
```

```python
def test_habitability(self):
    for fish in Guppi, Rasbora, Goldfish, Leopoldi:
        self._test_habitability(fish)

def _test_habitability(self, fish):
    self.given_aquarium(FreshwaterAquarium)
    self.when_checking_habitability_for(fish)
    self.then_aquarium_is_habitable()
```

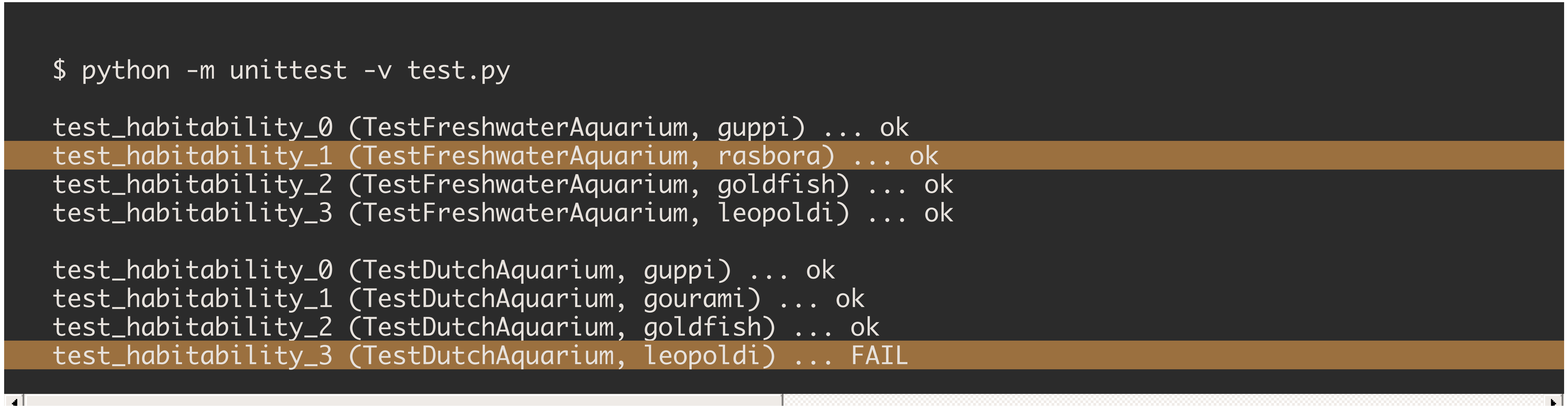
```python
from nose_parameterized import parameterized, param
...

class TestFreshwaterAquarium(BaseTestCase):
    @parameterized.expand([
        param(Guppi),
        param(Rasbora),
        param(Goldfish),
        param(Leopoldi),
    ])
    def test_habitability(self, fish):
        self.given_aquarium(FreshwaterAquarium)
        self.when_checking_habitability_for(fish)
        self.then_aquarium_is_habitable()
```

```python
from nose_parameterized import parameterized, param
...

class TestFreshwaterAquarium(BaseTestCase):
    @parameterized.expand([
        param(Guppi),
        param(Rasbora),
        param(Goldfish),
        param(Leopoldi),
    ])
    def test_habitability(self, fish):
        self.given_aquarium(FreshwaterAquarium)
        self.when_checking_habitability_for(fish)
        self.then_aquarium_is_habitable()
```

```python
class TestDutchAquarium(TestFreshwaterAquarium):
    @parameterized.expand([
        param(Guppi),
        param(Gourami),
        param(Goldfish),
    ])
    def test_habitability(self, fish):
        self.given_aquarium(FreshwaterAquarium)
        self.when_checking_habitability_for(fish)
        self.then_aquarium_is_habitable()
```



```
$ python -m unittest -v test.py

test_habitability_0 (TestFreshwaterAquarium, guppi) ... ok
test_habitability_1 (TestFreshwaterAquarium, rasbora) ... ok
test_habitability_2 (TestFreshwaterAquarium, goldfish) ... ok
test_habitability_3 (TestFreshwaterAquarium, leopoldi) ... ok

test_habitability_0 (TestDutchAquarium, guppi) ... ok
test_habitability_1 (TestDutchAquarium, gourami) ... ok
test_habitability_2 (TestDutchAquarium, goldfish) ... ok
test_habitability_3 (TestDutchAquarium, leopoldi) ... FAIL
```

Goals

- Parameterized tests

Goals

- Parameterized tests
- Inherited tests data

Goals

- Parameterized tests
- Inherited tests data
- No test repetition

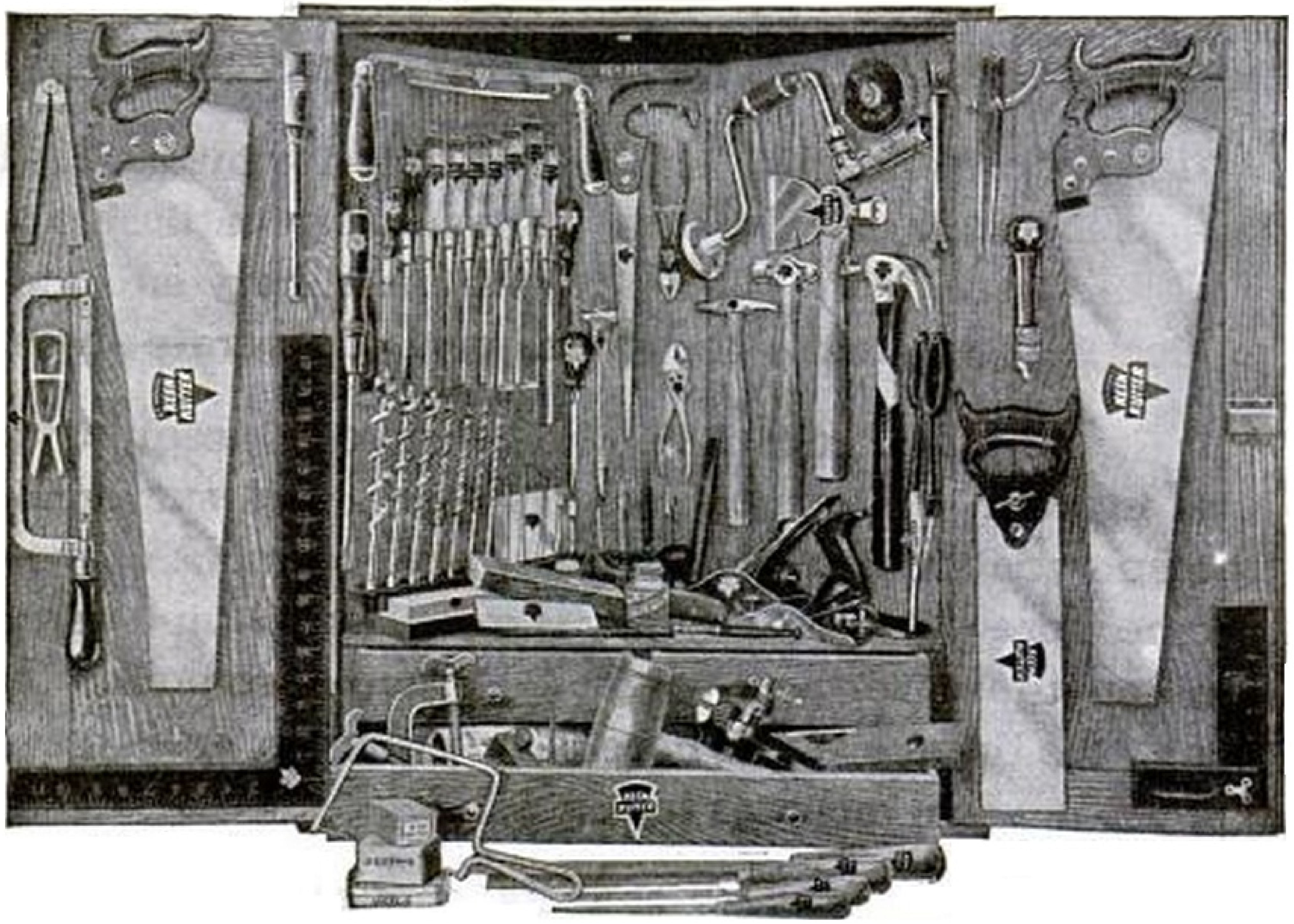
Goals

- Parameterized tests
- Inherited tests data
- No test repetition
- Controlled execution

Requirements

- Apply one test method to many data
- Access to parent class
- Exclude data

Tools / approaches



decorator

decorator

```
def new_function(*args, **kwargs):
    ...
    result = original_function(*args, **kwargs)
    ...
    return result
```

decorator

```
def new_function(*args, **kwargs):
    ...
    result = original_function(*args, **kwargs)
    ...
    return result
```

original function -> anything

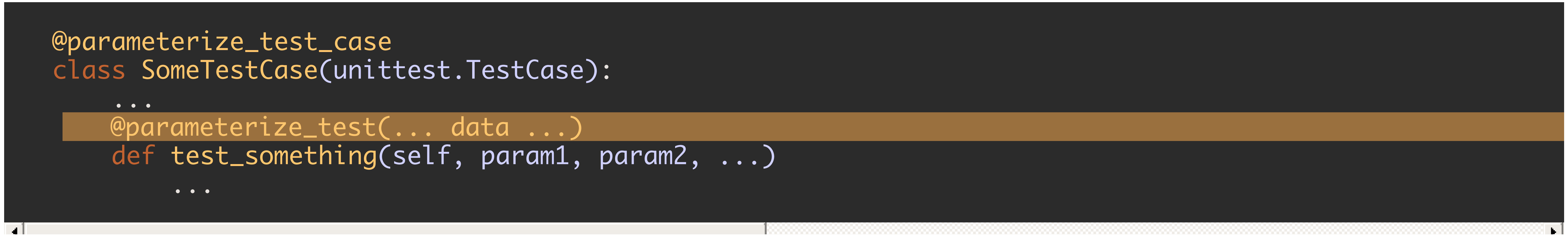
decorator

```python
def new_function(*args, **kwargs):
    ...
    result = original_function(*args, **kwargs)
    ...
    return result
```

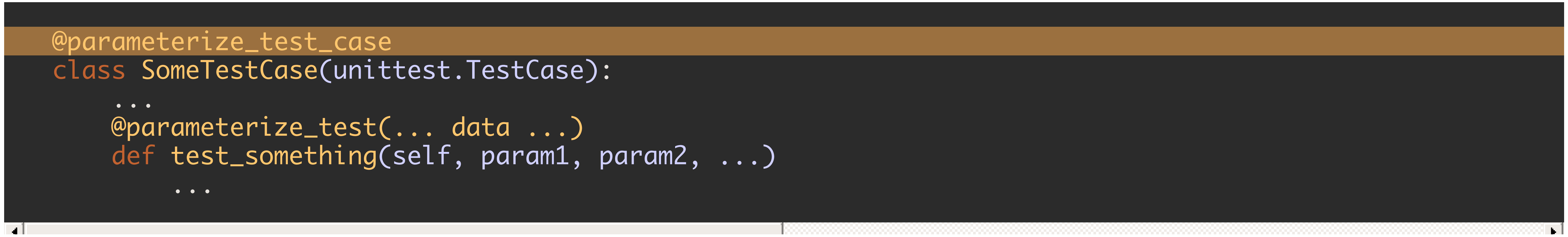
original function -> anything

original class -> anything

```
@parameterize_test_case
class SomeTestCase(unittest.TestCase):
    ...
    @parameterize_test(... data ...)
    def test_something(self, param1, param2, ...)
        ...
```



```
@parameterize_test_case
class SomeTestCase(unittest.TestCase):
    ...
    @parameterize_test(... data ...)
    def test_something(self, param1, param2, ...)
        ...
```



```
@parameterize_test_case
class SomeTestCase(unittest.TestCase):
    ...
    @parameterize_test(... data ...)
    def test_something(self, param1, param2, ...)
        ...
```

metaclass

metaclass

```python
class metacls(type):
    def __new__(mcs, name, bases, dict):
        dict.update(...)
        return type.__new__(mcs, name, bases, dict)
```

(name, bases, namespace) -> class

metaclass

```
class metacls(type):
    def __new__(mcs, name, bases, dict):
        dict.update(...)
        return type.__new__(mcs, name, bases, dict)
```

(name, bases, namespace) -> class

(name, bases, namespace) -> anything

frames

frames

```
Traceback (most recent call last):
  File "example.py", line 11, in
    sys.exit(main())
  File "example.py", line 8, in main
    module.function()
  File "/Users/allactaga/Development/Sources/Miscellaneous/grandson/package/module.py", line 4, in func
    1 / 0
ZeroDivisionError: division by zero
```

```python
from nose_parameterized import parameterized, param
...

class TestFreshwaterAquarium(BaseTestCase):
    @parameterized.expand([
        param(Guppi),
        param(Rasbora),
        param(Goldfish),
    ])
    def test_habitability(self, fish):
        ...
```

```python
from nose_parameterized import parameterized, param
...

class TestFreshwaterAquarium(BaseTestCase):
    @parameterized.expand([
        param(Guppi),
        param(Rasbora),
        param(Goldfish),
    ])
    def test_habitability(self, fish):
        ...
```

```python
class TestFreshwaterAquarium(BaseTestCase):
    def test_habitability_0(self, Guppi):
        ...

    def test_habitability_1(self, Rasbora):
        ...

    def test_habitability_2(self, Goldfish):
        ...

    @nottest
    def test_habitability(self, fish):
        ...
```

custom loader

test case classes -> test suites

```
def loadTestsFromTestCase(self, testCaseClass):
    ....
    testCaseNames = self.getTestCaseNames(testCaseClass)
    ...
    for name in testCaseNames:
        method = getattr(testCaseClass, name)
        if self.is_parametrized(method):
            suites.extend(self.create_test_cases_from(testCaseClass, method)
        else:
            suites.append(testCaseClass(name))
    return self.suiteClass(suites)
```

Skipping inherired data (controlled execution)

Skipping inherired data (controlled execution)

```
@parameterized.extend
@parameterized.remove
@parameterized.replace
```

Skipping inherired data (controlled execution)

```
@parameterized.extend
@parameterized.remove
@parameterized.replace
```

```
    ...
    def test_habitability(self, fish):
        self.assume_is_not_stingray(fish)
        ...

    def assume_is_not_stingray(self, fish)
        if fish.name in ['leopoldi']:
            self.skipTest("{} is a stingray")
```

git and github

git and github

svn

git and github

svn

dated folders (_my_code_2006_05_19_)

git and github

svn

dated folders (_my_code_2006_05_19_)

changing code live on production server

Thank you!