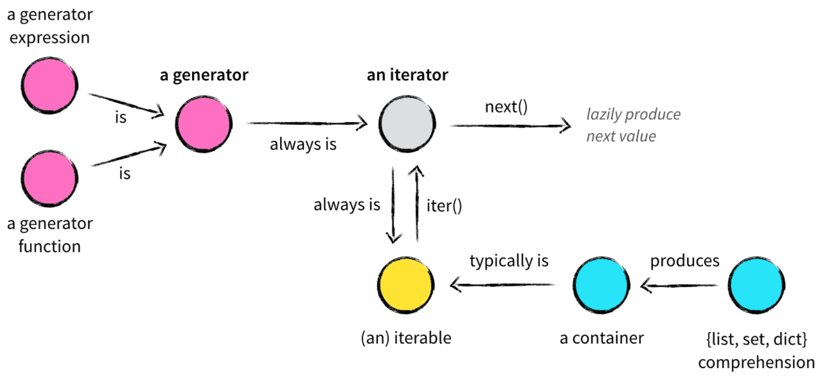
Testing

Alexander Evgin

06 марта 2020 г.

Outline

Recap



a generator expression
is
a generator function
is
a generator
always is
an iterator
next()
lazily produce next value
always is
iter()
(an) iterable
typically is
a container
produces
{list, set, dict} comprehension

Philosophy

> *The ratio of time spent reading versus writing is well over 10 to 1.*

Robert C. Martin

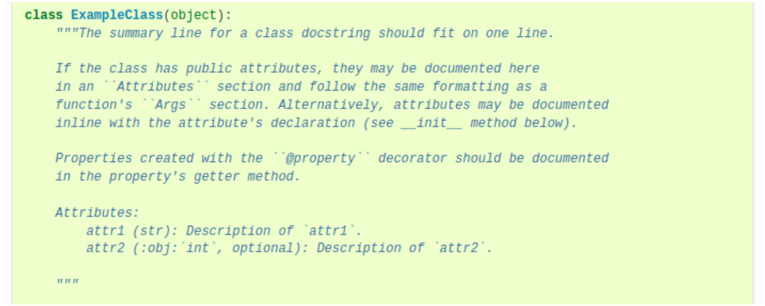
> *Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live.*

John Woods

docstring

```
def get_max(numbers):
    """Return max integer in the list.

    Args:
        numbers (list): List of numbers.

    Returns:
        int: Max value in the list of numbers.

    Raises:
        TypeError: If numbers contain not int object.

    """
    pass
```

Google Style Python Docstrings



```
class ExampleClass(object):
    """The summary line for a class docstring should fit on one line.

    If the class has public attributes, they may be documented here
    in an ``Attributes`` section and follow the same formatting as a
    function's ``Args`` section. Alternatively, attributes may be documented
    inline with the attribute's declaration (see __init__ method below).

    Properties created with the ``@property`` decorator should be documented
    in the property's getter method.

    Attributes:
        attr1 (str): Description of `attr1`.
        attr2 (:obj:`int`, optional): Description of `attr2`.

    """
```

https://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_
google.html

Type Annotations

```python
def pop(numbers: list, index: int = -1) -> int:
    pass
```

Type Annotations

```
def pop(numbers: list, index: int = -1) -> int:
    pass
```

Как указать, что `numbers` — список `int`-ов?

typing

```python
from typing import List, Dict, Optional, Union

def func(x: List[int],
         y: Dict[str, float],
         z: Union[str, List[str]],
         t: Optional[int] = None):
    pass
```

https://mypy.readthedocs.io/en/stable/cheat_sheet_py3.html

Type Annotations

```
class Logger:
    def new(self) -> Logger:  # unknown name!
        pass
```

```
class Logger:
    def new(self) -> 'Logger':
        pass
```

pycontracts

```
@contract(a='int,>0', b='list[N],N>0', returns='list[N]')
def my_function(a, b):
    pass

@contract
def my_function(a : 'int,>0', b : 'list[N],N>0') -> 'list[N]':
    # Python 3
    pass
```

pycontracts

```python
@contract(a='int,>0', b='list[N],N>0', returns='list[N]')
def my_function(a, b):
    pass

@contract
def my_function(a : 'int,>0', b : 'list[N],N>0') -> 'list[N]':
    # Python 3
    pass
```

```
>>> my_function(-1, [])
Traceback (most recent call last):
...
contracts.interface.ContractNotRespected: Breach for argument 'a'
to my_function().
Condition -1 > 0 not respected
checking: >0       for value: Instance of <class 'int'>: -1
checking: int,>0   for value: Instance of <class 'int'>: -1
Variables bound in inner context:
```

Testing

unit-testing

тестирование изолированных компонент / автоматизированное тестирование

integration-testing

тестирование взаимодействия компонент / не unit тестирование

Testing

- Простота написания
- Скорость работы (unit vs. integration)
- Стабильность при изменении кода (code churn)
- Стабильность при изменении внешних компонент
- Надёжность (тесты проходят, но работает ли программа?)

Testing

```python
import itertools

def rle(iterable):
    """Apply run-length encoding to an iterable."""
    for item, g in itertools.groupby(iterable):
        yield item, sum(1 for _ in g)
```

print

```
def rle(iterable):
    ...

print(list(rle("foo")))  # [('f', 1), ('o', 2)]
```

assert

```python
def rle(iterable):
    ...

assert list(rle("")) == []
assert list(rle("foo")) == [('f', 1), ('o', 2)]
```

assert

```python
def rle(iterable):
    ...

assert list(rle("foo")) == [('f', 1), ('o', 1)]
```

```
Traceback (most recent call last):
    File "rle.py", line 4, in <module>
        assert list(rle("foo")) == [('f', 1), ('o', 2)]
AssertionError
```

assert

```python
def rle(iterable):
    ...

assert list(rle("foo")) == [('f', 1), ('o', 1)], \
    'calculated {}'.format(list(rle("foo")))
```

```
Traceback (most recent call last):
    File "rle.py", line 4, in <module>
        assert list(rle("foo")) == [('f', 1), ('o', 1)]
AssertionError: calculated [('f', 1), ('o', 2)]
```

doctest

```
import itertools

def rle(iterable):
    """Apply run-length encoding to an iterable.

    >>> list(rle(""))
    []
    >>> list(rle("foo"))
    [('f', 1), ('o', 2)]
    """
    for item, g in itertools.groupby(iterable):
        yield item, sum(1for _ in g)
```

```
python -m doctest rle.py
```

unittest

```python
import unittest

class TestRle(unittest.TestCase):
    def test_rle_empty(self):
        self.assertEqual(list(rle("")), [])

    def test_rle(self):
        expected = [('f', 2), ('o', 2)]
        self.assertEqual(list(rle("foo")), expected)

if __name__ == '__main__':
    unittest.main()
```

```
python rle.py
```

unittest

```python
import unittest

class TestRle(unittest.TestCase):
    def test_rle_empty(self):
        self.assertEqual(list(rle("")), [])

    def test_rle(self):
        expected = [('f', 2), ('o', 2)]
        self.assertEqual(list(rle("foo")), expected)
```

```
python -m unittest rle.py
```

unittest

```
F.
======================================================
FAIL: test_rle (rle.TestHomework)
------------------------------------------------------
Traceback (most recent call last):
    File "rle.py", line 27, in test_rle
        self.assertEqual(list(rle("foo")), expected)
AssertionError: Lists differ: [] != [('f', 1), ('o', 2)]

Second list contains 2 additional elements.
First extra element0:
('f', 1)

- []
+ [('f', 1), ('o', 2)]
------------------------------------------------------
Ran 2 tests in 0.001s
FAILED (failures=1)
```

setUp/tearDown

```
import unittest
import tempfile

class TestWithTempFile(unittest.TestCase):
    def setUp(self):
        self.tempfile = tempfile.TemporaryFile()

    def tearDown(self):
        self.tempfile.close()

    def test_rle(self):
        self.tempfile.write('mississippi')
        self.assertEqual(rle_encode(self.tempfile),  [
            ...
        ])
```

setUp/tearDown

- тест A требует базу данных
- тест B требует временный файл
- тест C требует ***и*** базу данных, ***и*** временный файл

Но `setUp` один!

```
class TestWithTempFile(unittest.TestCase):
    def setUp(self):
        self.tempfile = tempfile.TemporaryFile()
        self.db = db.connect()
```

unittest

+ Из коробки
– Написать тест не так просто

pytest

+ The best of the best
+ Написать тест — просто и быстро
– Не из коробки

```python
def rle(iterable):
    return []

def test_rle_foo():
    assert rle("foo") == [('f', 1), ('o', 2)]
```

```
python -m pytest rle.py
```

pytest

```
======================= test session starts ========================
platform linux -- Python 3.6.9, pytest-5.3.5, py-1.8.1, pluggy-0.13.1
rootdir: /home/alea/misis/misis-py-adv/week_04_testing
collected 1 item

rle.py F                                                      [100%]

============================= FAILURES =============================
___________________________ test_rle_foo ___________________________

    def test_rle_foo():
>       assert rle("foo") == [('f', 1), ('o', 1)]
E       AssertionError: assert [] == [('f', 1), ('o', 1)]
E         Right contains 2 more items, first extra item: ('f', 1)
E         Use -v to get the full diff

rle.py:6: AssertionError
======================== 1 failed in 0.03s =========================
```

Fixtures

```python
import pytest

@pytest.fixture()
def tempfile():
    with TemporaryFile() as f:
        yield f

def test_with_tempdir(tempfile):
    tempfile.write(b'hello')
    tempfile.seek(0)
    assert tempfile.read() == b'hello'
```

Fixtures

```
@pytest.fixture()
def db():
    ...

@pytest.fixture()
def tempfile():
    ...

def test_dump_db_to_file(db, tempfile):
    pass
```

Fixtures

```
@pytest.fixture()
def db():
    ...

@pytest.fixture()
def tempfile():
    ...

def test_dump_db_to_file(db, tempfile):
    pass
```

Much better!

Fixtures

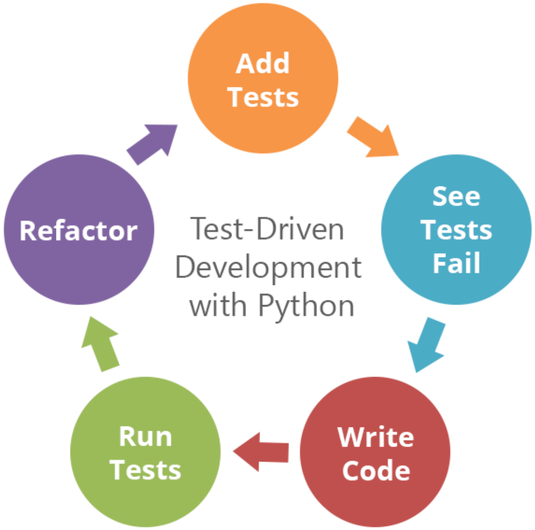
```
@pytest.fixture()
def tempfile():
    with TemporaryFile() as f:
        yield f

@pytest.fixture()
def hello_file(tempfile):
    tempfile.write(b'hello')
    tempfile.seek(0)
    return tempfile

def test_hello(hello_file):
    assert hello_file.read() == b'hello'
```

Very much better!

To test or not to test

- Написал функцию — напиши тест
- Неплохо: покрытие user-visible фичей (API)
- Тесты зависимостей (медленные)
- В продакшне — целая система

Test-driven development (TDD)



Define

```python
from typing import List

def sort(numbers: List[int]) -> List[int]:
    """Sort integers.

    Args:
        numbers: List to sort.

    Returns:
        list: Sorted list of numbers in increasing order.

    Raises:
        TypeError: If the is a non-integer element in list.
    """
    pass
```

Write test

```
def test_sort():
    assert sort([1, 3, 2]) == [1, 2, 3]
    assert sort([2, 1, 1]) == [1, 1, 2]
    assert sort([0, 4, -2, 1, 0, -1]) == [-2, -1, 0, 0, 1, 4]
```

See test fails

```
======================== test session starts =========================
platform linux -- Python 3.6.9, pytest-5.3.5, py-1.8.1, pluggy-0.13.1
rootdir: /home/alea/misis/misis-py-adv/week_04_testing
collected 1 item

tdd.py F                                                        [100%]

============================= FAILURES ===============================
_____________________________ test_sort ______________________________

    def test_sort():
>       assert sort([1, 3, 2]) == [1, 2, 3]
E       assert None == [1, 2, 3]
E        +  where None = sort([1, 3, 2])

tdd.py:21: AssertionError
========================= 1 failed in 0.03s ==========================
```

Code

```python
def sort(numbers: List[int]) -> List[int]:
    """Sort integers.

    Args:
        numbers: List to sort.

    Returns:
        list: Sorted list of numbers in increasing order.

    Raises:
        TypeError: If the is a non-integer element in list.
    """
    return sorted(numbers)
```

Pass test

```
========================= test session starts ==========================
platform linux -- Python 3.6.9, pytest-5.3.5, py-1.8.1, pluggy-0.13.1
rootdir: /home/alea/misis/misis-py-adv/week_04_testing
collected 1 item

tdd.py .                                                          [100%]

========================== 1 passed in 0.01s ===========================
```

Refactor

Write test

```python
import pytest

def test_sort_raises():
    with pytest.raises(TypeError):
        sort(['a', 'b', 'c'])
    with pytest.raises(TypeError):
        sort(['a', 1, None])
```

And so on...

Q & A