Московский государственный технический университет им. Н.Э. Баумана

Факультет ИУ
Кафедра ИУ5

Курс «Основы информатики»
Отчет лабораторной работе №5

Выполнил студент группы ИУ5-33Б:
Бакушев И.О.
Подпись и дата:

Проверил преподаватель каф.:
Гапанюк Ю. Е.
Подпись и дата:

Москва, 2024 г

Описание задания

1. Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4.
2. Модифицируйте код таким образом, чтобы он был пригоден для модульного тестирования.
3. Разработайте модульные тесты. В модульных тестах необходимо применить следующие технологии:
    - TDD - фреймворк (не менее 3 тестов).
    - BDD - фреймворк (не менее 3 тестов).

Текст программы (BDD – фреймворк)

```
from behave import *
class Unique(object):
    def __init__(self, items, **kwargs):
        self.items = iter(items)
        self.seen = set()
        self.ignore_case = kwargs.get('ignore_case', False)

    def __next__(self):
        while True:
            item = next(self.items)
            if self.ignore_case and isinstance(item, str):
                item = item.lower()
            if item not in self.seen:
                self.seen.add(item)
                return item

    def __iter__(self):
        return self

@given('a list of items {items}')
def step_impl(context, items):
    items_list = eval(items)
    context.items = items_list

@given('ignore case is {ignore_case}')
def step_impl(context, ignore_case):
    context.ignore_case = ignore_case == "True"

@when('I create a Unique iterator')
def step_impl(context):
    kwargs = {}
    if hasattr(context, 'ignore_case'):
        kwargs['ignore_case'] = context.ignore_case
    context.unique_iter = Unique(context.items, **kwargs)
    context.result = list(context.unique_iter)


@then('the iterator should yield {expected_result}')
def step_impl(context, expected_result):
```

```
    expected = eval(expected_result)
    assert context.result == expected
```

unique.feature

Feature: Unique iterator

  Scenario: Iterate over unique items
    Given a list of items [1, 2, 2, 3, 4, 4, 5]
    When I create a Unique iterator
    Then the iterator should yield [1, 2, 3, 4, 5]

  Scenario: Iterate over unique items ignoring case
    Given a list of items ["a", "A", "b", "B", "a"]
    And ignore case is True
    When I create a Unique iterator
    Then the iterator should yield ["a", "b"]

  Scenario: Iterate over mixed-case items without case-insensitivity
    Given a list of items ["a", "A", "b", "B", "a"]
    When I create a Unique iterator
    Then the iterator should yield ["a", "A", "b", "B"]

Выполнение программы

```
Feature: Unique iterator # ../unique.feature:1

  Scenario: Iterate over unique items              # ../unique.feature:3
    Given a list of items [1, 2, 2, 3, 4, 4, 5]    # unique_steps.py:20
    When I create a Unique iterator                # unique_steps.py:29
    Then the iterator should yield [1, 2, 3, 4, 5] # unique_steps.py:38

  Scenario: Iterate over unique items ignoring case  # ../unique.feature:8
    Given a list of items ["a", "A", "b", "B", "a"]  # unique_steps.py:20
    And ignore case is True                          # unique_steps.py:25
    When I create a Unique iterator                  # unique_steps.py:29
    Then the iterator should yield ["a", "b"]        # unique_steps.py:38

  Scenario: Iterate over mixed-case items without case-insensitivity  # ../unique.feature:14
    Given a list of items ["a", "A", "b", "B", "a"]                   # unique_steps.py:20
    When I create a Unique iterator                                   # unique_steps.py:29
    Then the iterator should yield ["a", "A", "b", "B"]               # unique_steps.py:38

1 feature passed, 0 failed, 0 skipped
3 scenarios passed, 0 failed, 0 skipped
10 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.003s
(venv) PS C:\Users\encourage\PycharmProjects\lab5\features\steps>
```

Текст программы (TDD – фреймворк)

main.py

```python
import unittest

class Unique(object):
    def __init__(self, items, **kwargs):
        self.items = iter(items)
        self.seen = set()
        self.ignore_case = kwargs.get('ignore_case', False)

    def __next__(self):
        while True:
            item = next(self.items)
            if self.ignore_case and isinstance(item, str):
                item = item.lower()
            if item not in self.seen:
                self.seen.add(item)
                return item

    def __iter__(self):
        return self

class TestUnique(unittest.TestCase):

    def test_empty_sequence(self):
        unique_iter = Unique([])
        with self.assertRaises(StopIteration):
            next(unique_iter)

    def test_unique_sequence(self):
        unique_iter = Unique([1, 2, 3, 4, 5])
```

```
        self.assertEqual([next(unique_iter) for _ in range(5)], [1, 2, 3, 4, 5])
        with self.assertRaises(StopIteration):
            next(unique_iter)

    def test_duplicate_sequence(self):
        unique_iter = Unique([1, 2, 2, 3, 3, 3, 1, 4])
        self.assertEqual([next(unique_iter) for _ in range(4)], [1, 2, 3, 4])
        with self.assertRaises(StopIteration):
            next(unique_iter)

    def test_ignore_case(self):
        unique_iter = Unique(['a', 'A', 'b', 'B', 'c'], ignore_case=True)
        self.assertEqual([next(unique_iter) for _ in range(3)], ['a', 'b', 'c'])
        with self.assertRaises(StopIteration):
            next(unique_iter)

    def test_mixed_types(self):
        unique_iter = Unique([1, 'a', 1, 'A', 'b', 2])
        self.assertEqual([next(unique_iter) for _ in range(5)], [1, 'a', 'A', 'b', 2])


    def test_ignore_case_mixed_types(self):
        unique_iter = Unique([1, 'a', 1, 'A', 'b', 2], ignore_case=True)
        self.assertEqual([next(unique_iter) for _ in range(4)], [1, 'a', 'b', 2])


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

Выполнение программы

```
......
----------------------------------------------------------------------
Ran 6 tests in 0.001s

OK

Process finished with exit code 0
```