Московский государственный технический университет им. Н.Э. Баумана

Факультет «Радиотехнический»
Кафедра «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования »

Отчет по лабораторной работе №3 «Модульное тестирование в Python.»

Выполнил:

студент
группы
РТ5-31Б

Ермаков И.А.

Москва, 2024 г

Описание задания:

Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4. Модифицируйте код таким образом, чтобы он был пригоден для модульного тестирования.

Разработайте модульные тесты. В модульных тестах необходимо применить следующие технологии:

- TDD - фреймворк (не менее 3 тестов).
- BDD - фреймворк (не менее 3 тестов).

Создание Mock-объектов (необязательное дополнительное задание).

Текст программы:

Unique.py

```
class Unique:
    def __init__(self, items, **kwargs):
        self.items = iter(items)  # Преобразуем входные данные в итератор
        self.seen = set()  # Множество для отслеживания уникальных элементов
        self.ignore_case = kwargs.get('ignore_case', False)  # Получаем параметр ignore_case

    def __iter__(self):
        return self

    def __next__(self):
        while True:
            current = next(self.items)  # Получаем следующий элемент из итератора

            # Преобразуем в нижний регистр, если ignore_case=True и элемент — строка
            check_value = current.lower() if self.ignore_case and isinstance(current, str) else current

            # Проверяем, был ли элемент уже обработан
            if check_value not in self.seen:
                self.seen.add(check_value)  # Добавляем в seen, чтобы не повторять
                return current  # Возвращаем исходное значение без изменения регистра

if __name__ == "__main__":
    data = [1, 1, 1, 1, 1, 2, 2, 2, 2, 2]
    print(list(Unique(data)))
```

```
    data = ['a', 'A', 'b', 'B', 'a', 'A', 'b', 'B']
    print(list(Unique(data)))
    print(list(Unique(data, ignore_case=True)))
```

Test_unique.py

```
import unittest
from unique import Unique

class TestUnique(unittest.TestCase):

    def test_unique_integers(self):
        data = [1, 1, 2, 2, 3, 3]
        unique_iterator = Unique(data)
        result = list(unique_iterator)
        self.assertEqual(result, [1, 2, 3])

    def test_unique_strings_case_sensitive(self):
        data = ['a', 'A', 'b', 'B']
        unique_iterator = Unique(data)
        result = list(unique_iterator)
        self.assertEqual(result, ['a', 'A', 'b', 'B'])

    def test_unique_strings_ignore_case(self):
        data = ['a', 'A', 'b', 'B']
        unique_iterator = Unique(data, ignore_case=True)
        result = list(unique_iterator)
        self.assertEqual(result, ['a', 'b'])

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

Test_unique_bdd.py

```
from unique import Unique
from pytest_bdd import given, when, then, scenario

@scenario("unique.feature", "Remove duplicates from a list of integers")
def test_unique_integers():
    pass

@scenario("unique.feature", "Remove duplicates from a case-sensitive list of
strings")
def test_unique_strings_case_sensitive():
```

```
    pass

@scenario("unique.feature", "Remove duplicates from a case-insensitive list of strings")
def test_unique_strings_ignore_case():
    pass

@given("a list of integers with duplicates")
def data_integers():
    return [1, 1, 2, 2, 3, 3]

@given("a case-sensitive list of strings with duplicates")
def data_strings_case_sensitive():
    return ['a', 'A', 'b', 'B']

@given("a case-insensitive list of strings with duplicates")
def data_strings_ignore_case():
    return ['a', 'A', 'b', 'B']

@when("we create a Unique iterator for integers")
def unique_integers(data_integers):
    return list(Unique(data_integers))

@when("we create a Unique iterator for case-sensitive strings")
def unique_strings_case_sensitive(data_strings_case_sensitive):
    return list(Unique(data_strings_case_sensitive))

@when("we create a Unique iterator for case-insensitive strings")
def unique_strings_ignore_case(data_strings_ignore_case):
    return list(Unique(data_strings_ignore_case, ignore_case=True))

@then("the result should only contain unique integers")
def verify_unique_integers(unique_integers):
    assert unique_integers == [1, 2, 3]

@then("the result should only contain unique case-sensitive strings")
def verify_unique_strings_case_sensitive(unique_strings_case_sensitive):
    assert unique_strings_case_sensitive == ['a', 'A', 'b', 'B']

@then("the result should only contain unique case-insensitive strings")
def verify_unique_strings_ignore_case(unique_strings_ignore_case):
    assert unique_strings_ignore_case == ['a', 'b']
```

Пример выполнения программы

```
Ran 3 tests in 0.000s

OK
```