Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №6

По дисциплине:“ Современные платформы программирования”

Выполнил:
Студент 3 курса
Группы ПО-11
Сологуб А.В.
Проверил:
Козик И.Д.

Брест 2025

Цель: освоить приемы тестирования кода на примере использования пакета pytest
Задание: Задание 1: Написание тестов для мини-библиотеки покупок (shopping.py)
1. Создайте файл test_cart.py. Реализуйте следующие тесты:
• Проверка добавления товара: после add_item("Apple", 10.0) в корзине должен быть один элемент.
• Проверка выброса ошибки при отрицательной цене.
• Проверка вычисления общей стоимости (total()).
2. Протестируйте метод apply_discount с разными значениями скидки:
Используйте @pytest.mark.parametrize
3. Создайте фикстуру empty_cart, которая возвращает пустой экземпляр Cart
Используйте эту фикстуру в тестах, где нужно создать новую корзину.
4. Допустим, у нас есть функция, которая логирует покупку в удалённую систему:
• Замокайте requests.post, чтобы не было реального HTTP-запроса
• Убедитесь, что он вызывается с корректными данными
5. Добавьте поддержку купонов:
• Напишите тесты на apply_coupon
• Замокайте словарь coupons с помощью monkeypatch или patch.dict
5) Реализуйте и протестируйте метод int indexOfDifference(String str1, String str2), который сравнивает две строки и возвращает индекс той позиции, в которой они различаются. Например, indexOfDifference("i am a machine" , "i am a robot") должно вернуть 7.

shopping.py

```
coupons = {
    "SAVE10": 10,
    "SAVE20": 20,
    "BLACKFRIDAY": 50
}

class Cart:
    def __init__(self):
        self.items = []
        self.discount = 0

    def add_item(self, name, price):
        if price < 0:
            raise ValueError("Price cannot be negative")
        self.items.append((name, price))

    def total_cost(self):
        total = sum(price for _, price in self.items)
        return total * (1 - self.discount / 100)

    def apply_discount(self, percent):
        if not (0 <= percent <= 100):
            raise ValueError("Discount must be between 0 and 100")
        self.discount = percent

    def apply_coupon(self, code):
        if code in coupons:
            self.discount = coupons[code]
        else:
            raise ValueError("Invalid coupon code")
```

```
def log_purchase(cart):
    total = cart.total_cost()
    print(f"Total purchase: ${total:.2f}")
```

utils.py

```
def indexOfDifference(str1, str2):
    min_len = min(len(str1), len(str2))
    for i in range(min_len):
        if str1[i] != str2[i]:
            return i
    if len(str1) != len(str2):
        return min_len
    return -1
```

test_cart.py

```
import pytest
from shopping import Cart, log_purchase, apply_coupon

def test_add_item():
    cart = Cart()
    cart.add_item("Book", 20)
    assert cart.total_cost() == 20

def test_negative_price_raises():
    cart = Cart()
    with pytest.raises(ValueError):
        cart.add_item("Free Item", -10)

def test_total_cost():
    cart = Cart()
    cart.add_item("Book", 10)
    cart.add_item("Pen", 5)
    assert cart.total_cost() == 15

@pytest.mark.parametrize("discount,expected", [(0, 100), (50, 50), (100, 0)])
def test_apply_discount_valid(discount, expected):
    cart = Cart()
    cart.add_item("Item", 100)
    cart.apply_discount(discount)
    assert cart.total_cost() == expected

@pytest.mark.parametrize("discount", [-10, 110])
def test_apply_discount_invalid(discount):
    cart = Cart()
    with pytest.raises(ValueError):
        cart.apply_discount(discount)

def test_log_purchase_mock(capfd):
    cart = Cart()
    cart.add_item("Book", 25)
    log_purchase(cart)
    out, _ = capfd.readouterr()
    assert "Total purchase: $25.00" in out

def test_apply_coupon_valid():
    cart = Cart()
    cart.add_item("Phone", 1000)
    cart.apply_coupon("SAVE10")
    assert cart.total_cost() == 900

def test_apply_coupon_invalid():
```

```
    cart = Cart()
    cart.add_item("Phone", 1000)
    with pytest.raises(ValueError):
        cart.apply_coupon("INVALIDCODE")
```

test_utils.py

```
import pytest
from shopping.utils import indexOfDifference

@pytest.mark.parametrize("str1,str2,expected", [
    ("", "", -1),
    ("abc", "", 0),
    ("", "abc", 0),
    ("abc", "abc", -1),
    ("abc", "abX", 2),
    ("abc", " abc", 0),  # <-- это был FAIL, теперь ожидаем 0
    ("abc", "abc ", 3),
    ("ab", "abxyz", 2),
    ("abcde", "abxyz", 2),
    ("abcde", "xyz", 0),
])
def test_index_of_difference(str1, str2, expected):
    assert indexOfDifference(str1, str2) == expected

def test_index_of_difference_none():
    assert indexOfDifference(None, None) == -1
```

```
============================= test session starts =============================
platform win32 -- Python 3.12.6, pytest-8.3.5, pluggy-1.5.0 -- D:\papka\SPP6\.venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: D:\papka\SPP6
configfile: pytest.ini
testpaths: tests
collected 22 items

tests/test_cart.py::test_add_item PASSED                                 [  4%]
tests/test_cart.py::test_negative_price_raises PASSED                    [  9%]
tests/test_cart.py::test_total_cost PASSED                               [ 13%]
tests/test_cart.py::test_apply_discount_valid[0-100] PASSED              [ 18%]
tests/test_cart.py::test_apply_discount_valid[50-50] PASSED              [ 22%]
tests/test_cart.py::test_apply_discount_valid[100-0] PASSED              [ 27%]
tests/test_cart.py::test_apply_discount_invalid[-10] PASSED              [ 31%]
tests/test_cart.py::test_apply_discount_invalid[110] PASSED              [ 36%]
tests/test_cart.py::test_log_purchase_mock PASSED                        [ 40%]
tests/test_cart.py::test_apply_coupon_valid PASSED                       [ 45%]
tests/test_cart.py::test_apply_coupon_invalid PASSED                     [ 50%]
tests/test_utils.py::test_index_of_difference[---1] PASSED               [ 54%]
tests/test_utils.py::test_index_of_difference[abc--0] PASSED             [ 59%]
tests/test_utils.py::test_index_of_difference[-abc-0] PASSED             [ 63%]
tests/test_utils.py::test_index_of_difference[abc-abc--1] PASSED         [ 68%]
tests/test_utils.py::test_index_of_difference[abc-abX-2] PASSED          [ 72%]
tests/test_utils.py::test_index_of_difference[abc- abc-0] PASSED         [ 77%]
tests/test_utils.py::test_index_of_difference[abc-abc -3] PASSED         [ 81%]
tests/test_utils.py::test_index_of_difference[ab-abxyz-2] PASSED         [ 86%]
tests/test_utils.py::test_index_of_difference[abcde-abxyz-2] PASSED      [ 90%]
tests/test_utils.py::test_index_of_difference[abcde-xyz-0] PASSED        [ 95%]
tests/test_utils.py::test_index_of_difference_none PASSED                [100%]

============================= 22 passed in 0.09s ==============================
```

Вывод: освоила приемы тестирования кода на примере использования пакета pytest