МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Лабораторная работа №6
По дисциплине “**Современные платформы программирования**”
Тема: “Тестирование кода с использованием пакета pytest”

Выполнил:
студент группы ПО-11
Надежук А.Г.
Проверил:
Козик И. Д.

Брест 2025

Цель: освоить приёмы тестирования кода на примере использования пакета pytest.

Вариант 3

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:
   - 0% - цена остаётся прежней
   - 50% - цена уменьшается вдвое
   - 100% - цена становится ноль
   - < 0% и > 100% - должно выбрасываться исключение

Используйте @pytest.mark.parametrize

3. Создайте фикстуру empty_cart, которая возвращает пустой экземпляр Cart

```
@pytest.fixture
def empty_cart():
    return Cart()
```

Используйте эту фикстуру в тестах, где нужно создать новую корзину.

4. Допустим, у нас есть функция, которая логирует покупку в удалённую систему:

```
import requests
def log_purchase(item):
    requests.post("https://example.com/log", json=item)
```

   - Замокайте requests.post, чтобы не было реального HTTP-запроса
   - Убедитесь, что он вызывается с корректными данными

5. Добавьте поддержку купонов:

```
def apply_coupon(cart, coupon_code):
    coupons = {"SAVE10": 10, "HALF": 50}
    if coupon_code in coupons:
      cart.apply_discount(coupons[coupon_code])
    else:
      raise ValueError("Invalid coupon")
```

   - Напишите тесты на apply_coupon
   - Замокайте словарь coupons с помощью monkeypatch или patch.dict.

Код программы:

shopping.py:

```
import requests
class Cart:
    def __init__(self):
        self.items = []

    def add_item(self, name, price):
        if price < 0:
            raise ValueError("Цена не может быть отрицательной")
        self.items.append({"name": name, "price": price})

    def total(self):
        return sum(item["price"] for item in self.items)

    def apply_coupon(self, coupon_code):
        coupons = {"SAVE10": 10, "HALF": 50}
        if coupon_code in coupons:
            discount = coupons[coupon_code]
            if 0 < discount <= 100:
                total = self.total()
                discount_amount = total * (discount / 100)
                for item in self.items:
                    item["price"] -= item["price"] * (discount / 100)
```

```
            else:
                raise ValueError("Скидка должна быть между 0 и 100")
        else:
            raise ValueError("Неверный купон")

def log_purchase(item):
    requests.post("https://example.com/log", json=item)
```

test_cart.py:

```
import pytest
from unittest.mock import patch
from shopping import log_purchase, Cart
@pytest.fixture
def empty_cart():
    return Cart()

def test_add_item(empty_cart):
    empty_cart.add_item("Яблоко", 10.0)
    assert len(empty_cart.items) == 1
    assert empty_cart.items[0]["name"] == "Яблоко"
    assert empty_cart.items[0]["price"] == 10.0

def test_add_item_negative_price(empty_cart):
    with pytest.raises(ValueError) as excinfo:
        empty_cart.add_item("Яблоко", -10.0)
    assert "Цена не может быть отрицательной" in str(excinfo.value)

def test_total(empty_cart):
    empty_cart.add_item("Яблоко", 10.0)
    empty_cart.add_item("Банан", 5.0)
    assert empty_cart.total() == 15.0

@pytest.mark.parametrize("discount, expected_price", [
    (0, 10.0),
    (50, 5.0),
    (100, 0.0),
])

def test_apply_discount(empty_cart, discount, expected_price):
    empty_cart.add_item("Товар", 10.0)
    assert empty_cart.apply_discount(discount) == expected_price
@pytest.mark.parametrize("discount", [-10, 110])

def test_apply_discount_invalid(empty_cart, discount):
    empty_cart.add_item("Товар", 10.0)
    with pytest.raises(ValueError) as excinfo:
        empty_cart.apply_discount(discount)
    assert "Скидка должна быть между 0 и 100" in str(excinfo.value)

@patch("requests.post")  # Mock the requests.post function
def test_log_purchase(mock_post):
    item = {"name": "Тестовый товар", "price": 20.0}
    log_purchase(item)
    mock_post.assert_called_once_with("https://example.com/log", json=item)

def test_apply_coupon_save10(empty_cart):
    empty_cart.add_item("Товар", 100.0)
    empty_cart.apply_coupon("SAVE10")
    assert empty_cart.total() == pytest.approx(90.0)

def test_apply_coupon_invalid(empty_cart):
     with pytest.raises(ValueError) as excinfo:
         empty_cart.apply_coupon("INVALID")
     assert "Неверный купон" in str(excinfo.value)
```

Результат выполнения:

```
platform win32 -- Python 3.13.1, pytest-8.3.5, pluggy-1.5.0
D:\PythonProjects\SPPLab6\.venv\Scripts\python.exe
collected 11 items
```

```
test_cart.py::test_add_item PASSED
test_cart.py::test_add_item_negative_price PASSED
test_cart.py::test_total PASSED
test_cart.py::test_apply_discount[0-10.0] PASSED
test_cart.py::test_apply_discount[50-5.0] PASSED
test_cart.py::test_apply_discount[100-0.0] PASSED
test_cart.py::test_apply_discount_invalid[-10] PASSED
test_cart.py::test_apply_discount_invalid[110] PASSED
test_cart.py::test_log_purchase PASSED
test_cart.py::test_apply_coupon_save10 PASSED
test_cart.py::test_apply_coupon_invalid PASSED
```

Задание 2. Напишите тесты к реализованным функциям из лабораторной работы №1. Проверьте тривиальные и граничные случае, а также варианты, когда может возникнуть исключительная ситуация. Если при реализации не использовались отдельные функции, необходимо провести рефакторинг кода.

functions.py

```
def find_modes(numbers):
    if not numbers:
        return []
    counts = {}
    for number in numbers:
        if not isinstance(number, (int, float)):
            raise TypeError("Элементы списка должны быть числами (int или float)")
        counts[number] = counts.get(number, 0) + 1
    max_count = 0
    modes = []
    for number, count in counts.items():
        if count > max_count:
            max_count = count
            modes = [number]
        elif count == max_count:
            modes.append(number)
    if len(counts) == len(modes) and len(set(counts.values())) == 1:
        return []
    return modes

def find_substring_in_string(substring, string):
    if not isinstance(substring, str) or not isinstance(string, str):
        raise TypeError("И substring, и string должны быть строками")
    index = string.find(substring)
    return index
```

test_find_modes.py

```
import pytest
from functions import find_modes
def test_find_modes_empty_list():
    assert find_modes([]) == []
def test_find_modes_single_mode():
    assert find_modes([1, 2, 2, 3]) == [2]
def test_find_modes_multiple_modes():
    assert find_modes([1, 2, 2, 3, 3]) == [2, 3]
def test_find_modes_no_modes():
    assert find_modes([1, 2, 3, 4]) == []
def test_find_modes_all_same():
    assert find_modes([5, 5, 5]) == []
def test_find_modes_mixed_types():
    with pytest.raises(TypeError):
        find_modes([1, "2", 2])
def test_find_modes_negative_numbers():
    assert find_modes([-1, -1, 0, 1]) == [-1]
def test_find_modes_floats():
     assert find_modes([1.0, 1.0, 2.0]) == [1.0]
def test_find_modes_large_numbers():
    assert find_modes([1000000, 1000000, 1]) == [1000000]
def test_find_modes_complex_example():
    assert find_modes([1, 2, 3, 2, 1, 4, 2, 5, 2]) == [2]
```

Результат выполнения:

```
platform win32 -- Python 3.13.1, pytest-8.3.5, pluggy-1.5.0 --
D:\PythonProjects\SPPLab6\.venv\Scripts\python.exe
collected 10 items
test_find_modes.py::test_find_modes_empty_list PASSED
test_find_modes.py::test_find_modes_single_mode PASSED
test_find_modes.py::test_find_modes_multiple_modes PASSED
test_find_modes.py::test_find_modes_no_modes PASSED
test_find_modes.py::test_find_modes_all_same PASSED
test_find_modes.py::test_find_modes_mixed_types PASSED
test_find_modes.py::test_find_modes_negative_numbers PASSED
test_find_modes.py::test_find_modes_floats PASSED
test_find_modes.py::test_find_modes_large_numbers PASSED
test_find_modes.py::test_find_modes_complex_example PASSED
```

test_find_substring_in_string.py

```
import pytest
from functions import find_substring_in_string
def test_find_substring_in_string_found():
    assert find_substring_in_string("sad", "butsad") == 3
def test_find_substring_in_string_not_found():
    assert find_substring_in_string("foo", "bar") == -1
def test_find_substring_in_string_empty_substring():
    assert find_substring_in_string("", "hello") == 0
def test_find_substring_in_string_empty_string():
    assert find_substring_in_string("abc", "") == -1
def test_find_substring_in_string_both_empty():
    assert find_substring_in_string("", "") == 0
def test_find_substring_in_string_substring_at_beginning():
    assert find_substring_in_string("hello", "hello world") == 0
def test_find_substring_in_string_substring_at_end():
    assert find_substring_in_string("world", "hello world") == 6
def test_find_substring_in_string_overlapping():
     assert find_substring_in_string("aaa", "aaaaaa") == 0
def test_find_substring_in_string_case_sensitive():
    assert find_substring_in_string("Hello", "hello world") == -1
def test_find_substring_in_string_non_string_input():
   with pytest.raises(TypeError):
        find_substring_in_string(123, "abc")
   with pytest.raises(TypeError):
        find_substring_in_string("abc", 123)
```

Результат выполнения:

```
platform win32 -- Python 3.13.1, pytest-8.3.5, pluggy-1.5.0 --
D:\PythonProjects\SPPLab6\.venv\Scripts\python.exe
collected 10 items
test_find_substring_in_string.py::test_find_substring_in_string_found PASSED
test_find_substring_in_string.py::test_find_substring_in_string_not_found PASSED
test_find_substring_in_string.py::test_find_substring_in_string_empty_substring PASSED
test_find_substring_in_string.py::test_find_substring_in_string_empty_string PASSED
test_find_substring_in_string.py::test_find_substring_in_string_both_empty PASSED
test_find_substring_in_string.py::test_find_substring_in_string_substring_at_beginning PASSED
test_find_substring_in_string.py::test_find_substring_in_string_substring_at_end PASSED
test_find_substring_in_string.py::test_find_substring_in_string_overlapping PASSED
test_find_substring_in_string.py::test_find_substring_in_string_case_sensitive PASSED
test_find_substring_in_string.py::test_find_substring_in_string_non_string_input PASSED
```

Задание 3. Написать тесты к методу, а затем реализовать сам метод по заданной спецификации. Напишите метод String keep(String str, String pattern) который оставляет в первой строке все символы, которые присутствуют во второй.

Спецификация метода:

keep (None , None) = TypeError

keep (None, *) = None

keep ("", *) = ""

keep (* , None) = ""

keep (* , "") = ""
keep (" hello ", "hl") = " hll "
keep (" hello ", "le") = " ell "

test_keep.py

```
import pytest
from keep_function import keep
def test_both_none():
    with pytest.raises(TypeError):
        keep(None, None)
def test_str_none_pattern_not_none():
    assert keep(None, "abc") is None
def test_empty_str():
    assert keep("", "abc") == ""
def test_str_not_none_pattern_none():
    assert keep("hello", None) == ""
def test_pattern_empty():
    assert keep("hello", "") == ""
def test_keep_h_and_l():
    assert keep(" hello ", "hl") == " hll "
def test_keep_l_and_e():
    assert keep(" hello ", "le") == " ell "
def test_no_matching_chars():
    assert keep("hello", "xia") == ""
def test_all_chars_match():
    assert keep("hello", "helo") == "hello"
def test_pattern_with_duplicates():
    assert keep("hello", "hheellllloo") == "hello"
def test_unicode_chars():
    assert keep("привет", "пвт") == "пвт"
def test_mixed_case():
    assert keep("Hello World", "HlWd") == "Hll Wld"
```

keep_functions.py

```
def keep(str, pattern):
    if str is None and pattern is None:
        raise TypeError("Оба аргумента не должны быть None")
    if str is None:
        return None
    if pattern is None:
        return ""
    result = ""
    pattern_chars = set(pattern)
    for char in str:
        if char == " " or char in pattern_chars:
            result += char
    return result
```

Результат выполнения:

```
platform win32 -- Python 3.13.1, pytest-8.3.5, pluggy-1.5.0 --
D:\PythonProjects\SPPLab6\.venv\Scripts\python.exe
collected 12 items
test_keep.py::test_both_none PASSED
test_keep.py::test_str_none_pattern_not_none PASSED
test_keep.py::test_empty_str PASSED
test_keep.py::test_str_not_none_pattern_none PASSED
test_keep.py::test_pattern_empty PASSED
test_keep.py::test_keep_h_and_l PASSED
test_keep.py::test_keep_l_and_e PASSED
test_keep.py::test_no_matching_chars PASSED
test_keep.py::test_all_chars_match PASSED
test_keep.py::test_pattern_with_duplicates PASSED
test_keep.py::test_unicode_chars PASSED
test_keep.py::test_mixed_case PASSED
```

Вывод: освоил приёмы тестирования кода на примере использования пакета pytest.