

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ
Кафедра интеллектуальных информационных технологий

Отчёт по лабораторной работе №6

Специальность ПО11

Выполнил
Гулевич Е.А.
студент группы ПО11

Проверил
А. А. Крощенко
ст. преп. кафедры ИИТ,
03.05.2025 г.

Цель работы: освоить приемы тестирования кода на примере использования пакета pytest

Вариант 7

Задание 1: Написание тестов для мини-библиотеки покупок (shopping.py)

Код программы :

test_cart.py

```
import pytest
from unittest.mock import patch, MagicMock
from shopping import Cart, log_purchase, apply_coupon

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

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

def test_negative_price(empty_cart):
    with pytest.raises(ValueError, match="Price cannot be negative"):
        empty_cart.add_item("Apple", -10.0)

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

@pytest.mark.parametrize("discount,expected", [
    (0, 10.0),
    (50, 5.0),
    (100, 0.0),
])
def test_apply_discount(empty_cart, discount, expected):
    empty_cart.add_item("Apple", 10.0)
    assert empty_cart.apply_discount(discount) == expected

@pytest.mark.parametrize("invalid_discount", [-10, 110])
def test_invalid_discount(empty_cart, invalid_discount):
    empty_cart.add_item("Apple", 10.0)
    with pytest.raises(ValueError, match="Discount must be between 0 and 100"):
        empty_cart.apply_discount(invalid_discount)

@patch('shopping.requests.post')
def test_log_purchase(mock_post):
    item = {"name": "Apple", "price": 10.0}
    log_purchase(item)
    mock_post.assert_called_once_with("https://example.com/log", json=item)
```

```

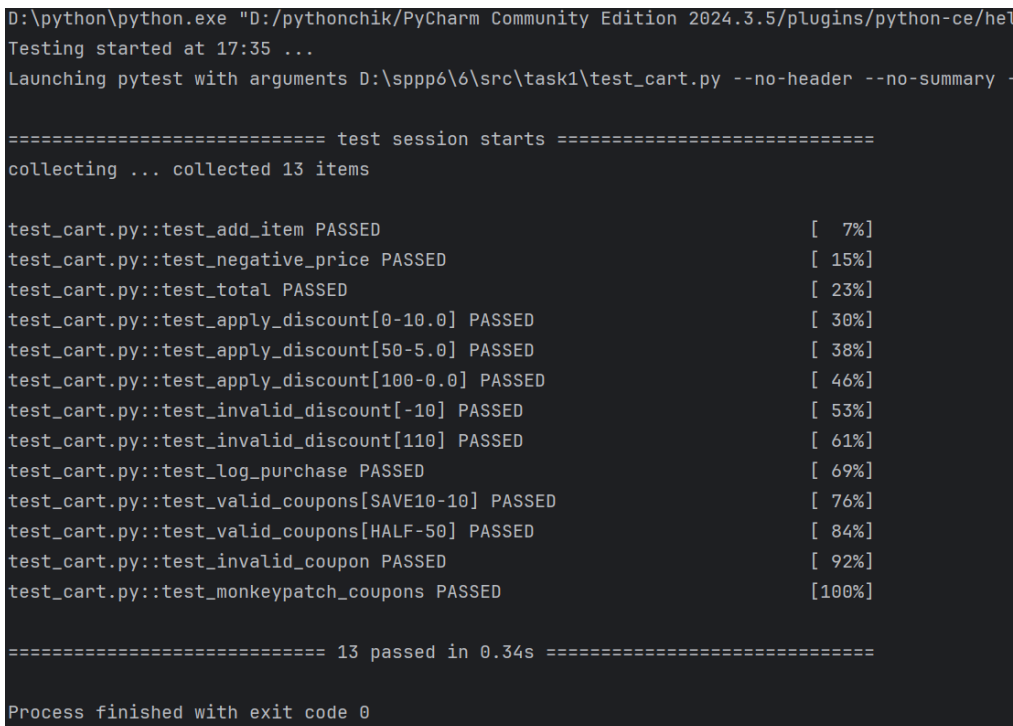
@pytest.mark.parametrize("coupon_code,discount", [
    ("SAVE10", 10),
    ("HALF", 50),
])
def test_valid_coupons(empty_cart, coupon_code, discount):
    empty_cart.add_item("Apple", 100.0)
    apply_coupon(empty_cart, coupon_code)
    assert empty_cart.apply_discount(discount) == 100.0 * (1 - discount / 100)

def test_invalid_coupon(empty_cart):
    with pytest.raises(ValueError, match="Invalid coupon"):
        apply_coupon(empty_cart, "INVALID")

@patch('shopping.coupons', {"TEST": 20})
def test_monkeypatch_coupons(empty_cart):
    empty_cart.add_item("Apple", 100.0)
    apply_coupon(empty_cart, "TEST")
    assert empty_cart.apply_discount(20) == 80.0

```

Изображение с результатами работы программы:



```

D:\python\python.exe "D:/pythonchik/PyCharm Community Edition 2024.3.5/plugins/python-ce/he
Testing started at 17:35 ...
Launching pytest with arguments D:\sppp6\6\src\task1\test_cart.py --no-header --no-summary -
===== test session starts =====
collecting ... collected 13 items

test_cart.py::test_add_item PASSED [ 7%]
test_cart.py::test_negative_price PASSED [ 15%]
test_cart.py::test_total PASSED [ 23%]
test_cart.py::test_apply_discount[0-10.0] PASSED [ 30%]
test_cart.py::test_apply_discount[50-5.0] PASSED [ 38%]
test_cart.py::test_apply_discount[100-0.0] PASSED [ 46%]
test_cart.py::test_invalid_discount[-10] PASSED [ 53%]
test_cart.py::test_invalid_discount[110] PASSED [ 61%]
test_cart.py::test_log_purchase PASSED [ 69%]
test_cart.py::test_valid_coupons[SAVE10-10] PASSED [ 76%]
test_cart.py::test_valid_coupons[HALF-50] PASSED [ 84%]
test_cart.py::test_invalid_coupon PASSED [ 92%]
test_cart.py::test_monkeypatch_coupons PASSED [100%]

===== 13 passed in 0.34s =====
Process finished with exit code 0

```

Задание 2 - Напишите тесты к реализованным функциям из лабораторной работы No1. Проверьте тривиальные и граничные случаи, а также варианты, когда может возникнуть исключительная ситуация. Если при реализации не использовались отдельные функции, необходимо провести рефакторинг кода.

Код программы:

```

test1_1.py
import pytest
from task_1 import find_median

```

```

# Тесты для функции find_median
@pytest.mark.parametrize("numbers,expected", [
    ([1, 2, 3], 2),          # Нечетное количество чисел
    ([1, 2, 3, 4], 2.5),     # Четное количество чисел
    ([5], 5),                # Один элемент
    ([1, 1, 1, 1, 1], 1),    # Все элементы одинаковые
    ([3, 1, 4, 1, 5, 9, 2, 6, 5, 3], 3.5), # Большой неотсортированный список
])

def test_find_median(numbers, expected):
    assert find_median(numbers.copy()) == expected # Используем копию, так как функция модифицирует
    список

def test_find_median_empty_list():
    with pytest.raises(IndexError):
        find_median([])

# Тест на типы данных
def test_find_median_invalid_types():
    with pytest.raises(TypeError):
        find_median(["a", "b", "c"]) # Строки вместо чисел

# Тест на граничные значения
def test_find_median_large_numbers():
    assert find_median([10000000000, -10000000000]) == 0

# Тест на отрицательные числа
def test_find_median_negative_numbers():
    assert find_median([-5, -3, -1, -2, -4]) == -3

# Тест на смешанные типы чисел (int и float)
def test_find_median_mixed_types():
    assert find_median([1, 2.5, 3, 4.5]) == 2.75

```

Изображение с результатами работы программы:

```

D:\python\python.exe "D:\pythonchik\PyCharm Community Edition 2024.3.5\plugins\python-ce\helpers\pycharm\_jb_pytest_r
Testing started at 17:38 ...
Launching pytest with arguments D:\sppp6\6\src\task2\test1_1.py --no-header --no-summary -q in D:\sppp6\6\src\task2

===== test session starts =====
collecting ... collected 10 items

test1_1.py::test_find_median[numbers0-2] PASSED [ 10%]
test1_1.py::test_find_median[numbers1-2.5] PASSED [ 20%]
test1_1.py::test_find_median[numbers2-5] PASSED [ 30%]
test1_1.py::test_find_median[numbers3-1] PASSED [ 40%]
test1_1.py::test_find_median[numbers4-3.5] PASSED [ 50%]
test1_1.py::test_find_median_empty_list PASSED [ 60%]
test1_1.py::test_find_median_invalid_types PASSED [ 70%]
test1_1.py::test_find_median_large_numbers PASSED [ 80%]
test1_1.py::test_find_median_negative_numbers PASSED [ 90%]
test1_1.py::test_find_median_mixed_types PASSED [100%]

===== 10 passed in 0.02s =====

Process finished with exit code 0

```

test1_2.py

```
import pytest
from task_1_2 import plus_one

# Тесты для функции plus_one
@pytest.mark.parametrize("digits,expected", [
    ([1, 2, 3], [1, 2, 4]),      # Обычный случай
    ([9], [1, 0]),               # Однозначное число 9
    ([9, 9, 9], [1, 0, 0, 0]),   # Все цифры 9
    ([1, 9, 9], [2, 0, 0]),      # Некоторые цифры 9
    ([0], [1]),                  # Ноль
    ([1, 0, 0], [1, 0, 1]),      # Число с нулями на конце
])
def test_plus_one(digits, expected):
    result = plus_one(digits.copy()) # Используем копию, так как функция модифицирует список
    assert result == expected

def test_plus_one_empty_list():
    with pytest.raises(IndexError):
        plus_one([])

# Тест на типы данных
def test_plus_one_invalid_types():
    with pytest.raises(TypeError):
        plus_one(["1", "2", "3"]) # Строки вместо чисел

# Тест на граничные значения
def test_plus_one_large_number():
    result = plus_one([9] * 1000) # Очень большое число из девяток
    assert len(result) == 1001
    assert result[0] == 1
    assert all(x == 0 for x in result[1:])

# Тест на отрицательные числа
def test_plus_one_negative_digits():
    with pytest.raises(ValueError):
        plus_one([-1, 2, 3]) # Отрицательные цифры недопустимы

# Тест на числа больше 9
def test_plus_one_large_digits():
    with pytest.raises(ValueError):
        plus_one([10, 2, 3]) # Цифры больше 9 недопустимы
```

Изображение с результатами работы программы:

```
D:\python\python.exe "D:/pythonchik/PyCharm Community Edition 2024.3.5/plugins/python-ce/helpers/pycharm/_jb_pytest_runner.py"
Testing started at 17:41 ...
Launching pytest with arguments D:\sppp6\6\src\task2\test1_2.py --no-header --no-summary -q in D:\sppp6\6\src\task2

===== test session starts =====
collecting ... collected 11 items

test1_2.py::test_plus_one[digits0-expected0] PASSED [ 9%]
test1_2.py::test_plus_one[digits1-expected1] PASSED [ 18%]
test1_2.py::test_plus_one[digits2-expected2] PASSED [ 27%]
test1_2.py::test_plus_one[digits3-expected3] PASSED [ 36%]
test1_2.py::test_plus_one[digits4-expected4] PASSED [ 45%]
test1_2.py::test_plus_one[digits5-expected5] PASSED [ 54%]
test1_2.py::test_plus_one_empty_list PASSED [ 63%]
test1_2.py::test_plus_one_invalid_types PASSED [ 72%]
test1_2.py::test_plus_one_large_number PASSED [ 81%]
test1_2.py::test_plus_one_negative_digits PASSED [ 90%]
test1_2.py::test_plus_one_large_digits PASSED [100%]

===== 11 passed in 0.02s =====

Process finished with exit code 0
```

Задание 3.

Написать тесты к методу, а затем реализовать сам метод по заданной спецификации.

7) Напишите метод `String substringBetween(String str, String open, String close)` выделяющий подстроку относительно открывающей и закрывающей строки.

Спецификация метода:

`substringBetween (None , None, None) = TypeError`

`substringBetween (None, * , *) = None`

`substringBetween (* , None, *) = None`

`substringBetween (* , * , None) = None`

`substringBetween ("", "", "") = ""`

`substringBetween ("", "", "[") = None`

`substringBetween ("", "[", "]") = None`

`substringBetween (" yabcz ", "", "") = ""`

`substringBetween (" yabcz ", "y", "z") = " abc"`

`substringBetween (" yabczyabcz ", "y", "z") = " abc "`

`substringBetween ("wx[b]yz", "[", "]") = "b"`

Код программы:

test_substring_between.py

```
import pytest
```

```
from substring_between import substringBetween
```

```
def test_substringBetween_all_none():
```

```
    """Test: substringBetween(None, None, None) = TypeError"""
```

```
    with pytest.raises(TypeError):
```

```
        substringBetween(None, None, None)
```

```
def test_substringBetween_str_none():
```

```
    """Test: substringBetween(None, *, *) = None"""
```

```
    assert substringBetween(None, "a", "b") is None
```

```

def test_substringBetween_open_none():
    """Test: substringBetween(*, None, *) = None"""
    assert substringBetween("abc", None, "b") is None

def test_substringBetween_close_none():
    """Test: substringBetween(*, *, None) = None"""
    assert substringBetween("abc", "a", None) is None

def test_substringBetween_empty_strings():
    """Test: substringBetween("", "", "") = """""
    assert substringBetween("", "", "") == ""

def test_substringBetween_empty_with_close():
    """Test: substringBetween("", "", ']') = None"""
    assert substringBetween("", "", "]") is None

def test_substringBetween_empty_with_brackets():
    """Test: substringBetween("", '[', ']') = None"""
    assert substringBetween("", "[", "]") is None

def test_substringBetween_empty_delimiters():
    """Test: substringBetween(' yabcz ', ' ', ' ') = """""
    assert substringBetween(" yabcz ", "", "") == ""

def test_substringBetween_simple():
    """Test: substringBetween(' yabcz ', 'y', 'z') = ' abc'"""
    assert substringBetween(" yabcz ", "y", "z") == "abc"

def test_substringBetween_multiple():
    """Test: substringBetween(' yabczyabcz ', 'y', 'z') = ' abc'"""
    assert substringBetween(" yabczyabcz ", "y", "z") == "abc"

def test_substringBetween_brackets():
    """Test: substringBetween('wx[b]yz', '[', ']') = 'b'"""
    assert substringBetween("wx[b]yz", "[", "]") == "b"

```

Рисунок с результатами работы программы:

```

D:\python\python.exe "D:\pythonchik\PyCharm Community Edition 2024.3.5\plugins\python-ce\helpers\pycharm\_jb_pytest_runner.py" --path D:
Testing started at 17:44 ...
Launching pytest with arguments D:\sppp6\6\src\task3\test_substring_between.py --no-header --no-summary -q in D:\sppp6\6\src\task3

===== test session starts =====
collecting ... collected 11 items

test_substring_between.py::test_substringBetween_all_none PASSED [ 9%]
test_substring_between.py::test_substringBetween_str_none PASSED [ 18%]
test_substring_between.py::test_substringBetween_open_none PASSED [ 27%]
test_substring_between.py::test_substringBetween_close_none PASSED [ 36%]
test_substring_between.py::test_substringBetween_empty_strings PASSED [ 45%]
test_substring_between.py::test_substringBetween_empty_with_close PASSED [ 54%]
test_substring_between.py::test_substringBetween_empty_with_brackets PASSED [ 63%]
test_substring_between.py::test_substringBetween_empty_delimiters PASSED [ 72%]
test_substring_between.py::test_substringBetween_simple PASSED [ 81%]
test_substring_between.py::test_substringBetween_multiple PASSED [ 90%]
test_substring_between.py::test_substringBetween_brackets PASSED [100%]

===== 11 passed in 0.02s =====

Process finished with exit code 0

```

Вывод: освоил приемы тестирования кода на примере использования пакета pytest

