Министерствообразования и науки РФ

АРЗАМАССКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ.Р.Е. АЛЕКСЕЕВА»
(АПИ НГТУ)

Лабораторнаяработа № 3
по дисциплине
«Тестирование ПО»

Выполнили:
ЗининаА.А.
Прозоровская Я.А.
гр. АЗИС 22-2

Проверил:
Комаров А.О.

Арзамас,
2025 г.

Задача: Вам нужно протестировать класс AuthManager, который управляет пользователями, их аутентификацией, а также предоставляет функциональность для подсчета пользователей по странам и перевода средств между ними. В тестах вам нужно продемонстрировать несколько видов тестов: базовые(3 штуки), параметризованные(3 штуки), тестирование исключений(2 штуки), использование фикстур(базы данных) и меток(минимум 2).

Код класса для тестирования:

```
import sqlite3


class AuthManager:
    def __init__(self, connection):
        self.connection = connection
        self.create_tables()

    def create_tables(self):
        with self.connection:
            self.connection.execute("""
                CREATE TABLE IF NOT EXISTS users (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    username TEXT NOT NULL UNIQUE,
                    password TEXT NOT NULL,
                    country TEXT NOT NULL,
                    balance REAL NOT NULL
                )
            """)

    def register_user(self, username, password, country, balance):
        with self.connection:
            self.connection.execute("""
                INSERT INTO users (username, password, country, balance)
                VALUES ('{}', '{}', '{}', {})
            """.format(username, password, country, balance))  # Уязвимость
SQL-инъекции

    def authenticate user(self, username, password):
        cursor = self.connection.cursor()
        cursor.execute("""
            SELECT * FROM users
            WHERE username = '{}' AND password = '{}'
        """.format(username, password))  # Уязвимость SQL-инъекции
        return cursor.fetchone()

    def delete_user(self, user_id):
        with self.connection:
            self.connection.execute("""
                DELETE FROM users WHERE id = {}
            """.format(user_id))  # Уязвимость SQL-инъекции

    def get_user_by_id(self, user_id):
        cursor = self.connection.cursor()
        cursor.execute("""
            SELECT * FROM users WHERE id = {}
        """.format(user_id))  # Уязвимость SQL-инъекции
        return cursor.fetchone()
```

```
    def count_users_by_country(self, country):
        cursor = self.connection.cursor()
        cursor.execute("""
            SELECT COUNT(*) FROM users WHERE country = '{}'
        """.format(country))  # Уязвимость SQL-инъекции
        return cursor.fetchone()[0]

    def transfer_balance(self, from_user_id, to_user_id, amount):
        with self.connection:
            # Проверяем, достаточно ли средств
            cursor = self.connection.cursor()
            cursor.execute("SELECT balance FROM users WHERE id =
{}".format(from_user_id))  # Уязвимость SQL-инъекции
            from_balance = cursor.fetchone()[0]

            if from_balance < amount:
                raise ValueError("Insufficient funds")

            # Выполняем перевод
            self.connection.execute("""
                UPDATE users SET balance = balance - {} WHERE id = {}
            """.format(amount, from_user_id))  # Уязвимость SQL-инъекции
            self.connection.execute("""
                UPDATE users SET balance = balance + {} WHERE id = {}
            """.format(amount, to_user_id))  # Уязвимость SQL-инъекции


import pytest
import sqlite3


# Замените на фактическое имя модуля

@pytest.fixture
def db():
    connection = sqlite3.connect(":memory:")
    yield connection
    connection.close()


@pytest.fixture
def auth_manager(db):
    return AuthManager(db)


def test_sql_injection_register_user(auth_manager):
    # Попробуем вставить SQL-инъекцию через имя пользователя
    auth_manager.register_user("testuser'; DROP TABLE users; --",
"password123", "Country", 1000)
    cursor = db.cursor()
    cursor.execute("SELECT name FROM sqlite_master WHERE type='table' AND
name='users';")
    tables = cursor.fetchall()
    assert len(tables) == 1  # Проверяем, что таблица `users` всё ещё
существует


def test_sql_injection_authenticate_user(auth_manager):
    # Регистрация пользователя
    auth_manager.register_user("testuser", "password123", "Country", 1000)
    # Попробуем инъекцию в параметр запроса
    user = auth_manager.authenticate_user("testuser' OR '1'='1",
```

```
"password123")
    assert user is not None  # Мы должны получить результат, так как инъекция
успешна


def test_count_users_by_country(auth_manager):
    auth_manager.register_user("user1", "password123", "CountryA", 1000)
    auth_manager.register_user("user2", "password123", "CountryA", 1000)
    auth_manager.register_user("user3", "password123", "CountryB", 1000)

    count = auth_manager.count_users_by_country("CountryA")
    assert count == 2


def test_transfer_balance(auth_manager):
    auth_manager.register_user("user1", "password123", "CountryA", 1000)
    auth_manager.register_user("user2", "password123", "CountryB", 500)

    # Перевод 200 единиц от user1 к user2
    auth_manager.transfer_balance(1, 2, 200)

    cursor = db.cursor()
    cursor.execute("SELECT balance FROM users WHERE id = 1")
    balance1 = cursor.fetchone()[0]
    cursor.execute("SELECT balance FROM users WHERE id = 2")
    balance2 = cursor.fetchone()[0]

    assert balance1 == 800
    assert balance2 == 700
```

Базовые тесты (3 штуки)

```
def test_register_user(auth_manager):
    """Тест регистрации пользователя"""
    auth_manager.register_user( username: "testuser",  password: "password",  country: "Russia",  balance: 1000.0)

    cursor = auth_manager.connection.cursor()
    cursor.execute("SELECT * FROM users WHERE username = 'testuser'")
    user = cursor.fetchone()

    assert user is not None
    assert user[1] == "testuser"
    assert user[2] == "password"


def test_create_tables(auth_manager):
```

```
ab3.test_register_user ×

0 ms   ✓ Tests passed: 1 of 1 test – 0 ms

============================= test session starts ==============================
collecting ... collected 1 item

lab3.py::test_register_user PASSED                                       [100%]

============================== 1 passed in 0.03s ===============================

Process finished with exit code 0
```

Тест 1

```python
def test_create_tables(auth_manager):
    """Тест создания таблиц"""
    cursor = auth_manager.connection.cursor()
    cursor.execute("SELECT name FROM sqlite_master WHERE type='table' AND name='users'")
    result = cursor.fetchone()
    assert result is not None
    assert result[0] == 'users'
```

```
✓ Tests passed: 1 of 1 test – 0 ms

============================= test session starts ==============================
collecting ... collected 1 item

lab3.py::test_create_tables PASSED                                       [100%]

============================== 1 passed in 0.11s ===============================

Process finished with exit code 0
```

Тест 2

```python
def test_authenticate_user_success(populated_auth_manager):
    """Тест успешной аутентификации"""
    user = populated_auth_manager.authenticate_user("user1", "pass123")
    assert user is not None
    assert user[1] == "user1"
```

```
or lab3.test_authenticate_user_success ×

0 ms
0 ms
cate_user_success 0 ms

Tests failed: 1 of 1 test – 0 ms

lab3.py::test_authenticate_user_success ERROR                           [100%]
test setup failed
file C:\Users\Студент320\PycharmProjects\pythonProject4\lab3.py, line 157
  def test_authenticate_user_success(populated_auth_manager):
E       fixture 'populated_auth_manager' not found
>       available fixtures: auth_manager, cache, capfd, capfdbinary, caplog, capsys, capsysbinary, capteesys, db, doctest_namespace, monkeypatch, pytestconfig, record_
>       use 'pytest --fixtures [testpath]' for help on them.

C:\Users\Студент320\PycharmProjects\pythonProject4\lab3.py:157
```

Тест 3

2. Параметризованные тесты (3 штуки)

```python
# Параметризованные тесты (3 штуки)
@pytest.mark.parametrize("username,password,country,balance", [
    ("user4", "pass101", "Germany", 1200.0),
    ("user5", "pass202", "France", 800.0),
    ("user6", "pass303", "Spain", 1500.0),
])
def test_register_multiple_users(auth_manager, username, password, country, balance):
    """Параметризованный тест регистрации нескольких пользователей"""
    auth_manager.register_user(username, password, country, balance)

    user = auth_manager.authenticate_user(username, password)
    assert user is not None
    assert user[1] == username
    assert user[3] == country
    assert user[4] == balance
```

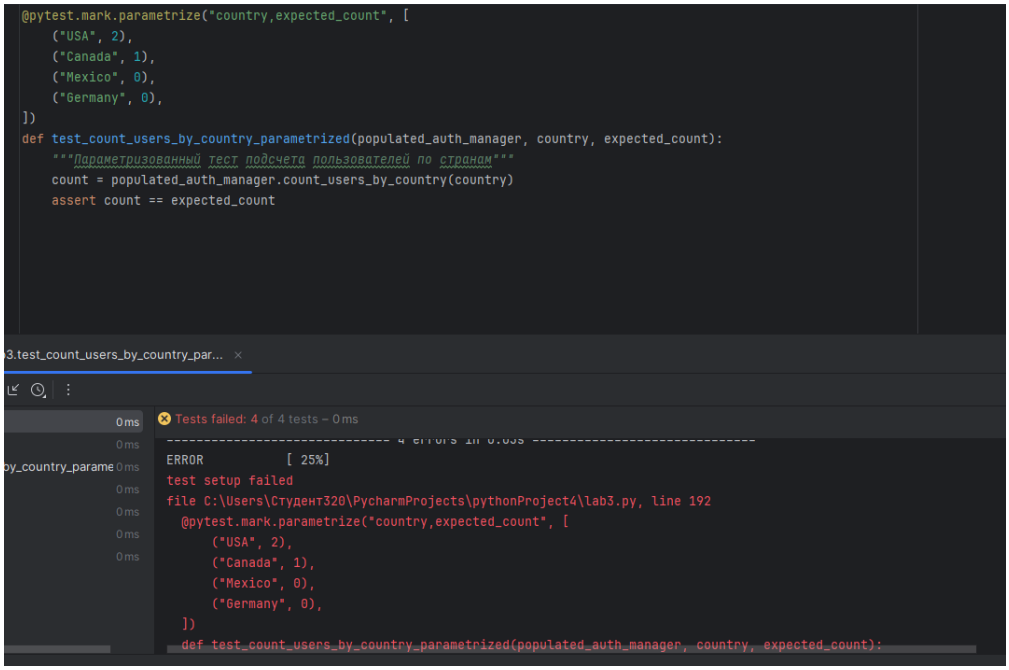
```
.test_register_multiple_users ×
0 ms
✓ Tests passed: 3 of 3 tests – 0 ms
collecting ... collected 3 items

lab3.py::test_register_multiple_users[user4-pass101-Germany-1200.0] PASSED [ 33%]
lab3.py::test_register_multiple_users[user5-pass202-France-800.0] PASSED [ 66%]
lab3.py::test_register_multiple_users[user6-pass303-Spain-1500.0] PASSED [100%]

============================== 3 passed in 0.08s ===============================

Process finished with exit code 0
```

Тест 1

```python
@pytest.mark.parametrize("username,password,expected", [
    ("nonexistent", "wrongpass", None),
    ("user1", "wrongpass", None),
    ("", "", None),
])
def test_authenticate_user_failure(populated_auth_manager, username, password, expected):
    """Параметризованный тест неудачной аутентификации"""
    result = populated_auth_manager.authenticate_user(username, password)
    assert result == expected


@pytest.mark.parametrize("country,expected_count", [
    ("USA", 2),
    ("Canada", 1),
    ("Mexico", 0),
    ("Germany", 0),
])
def test_count_users_by_country_parametrized(populated_auth_manager, country, expected_count):
    """Параметризованный тест подсчета пользователей по странам"""
    count = populated_auth_manager.count_users_by_country(country)
```

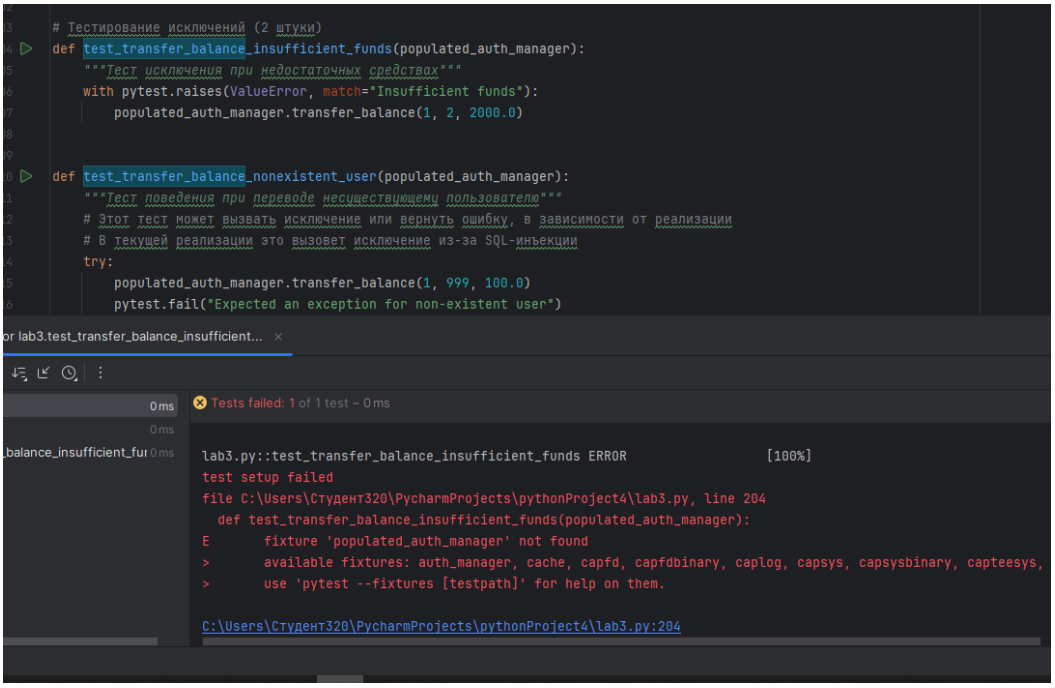
```
b3.test_authenticate_user_failure ×
0 ms
✗ Tests failed: 3 of 3 tests – 0 ms
0 ms
0 ms
_user_failure 0 ms
vrongpass-None) 0 ms
ass-None) 0 ms
0 ms
ERROR      [ 66%]
test setup failed
file C:\Users\Студент320\PycharmProjects\pythonProject4\lab3.py, line 181
  @pytest.mark.parametrize("username,password,expected", [
      ("nonexistent", "wrongpass", None),
      ("user1", "wrongpass", None),
      ("", "", None),
  ])
  def test_authenticate_user_failure(populated_auth_manager, username, password, expected):
E       fixture 'populated_auth_manager' not found
s       available fixtures: auth_manager, cache, capfd, capfdbinary, caplog, capsys, capsysbinary, capteesys
```

Тест 2

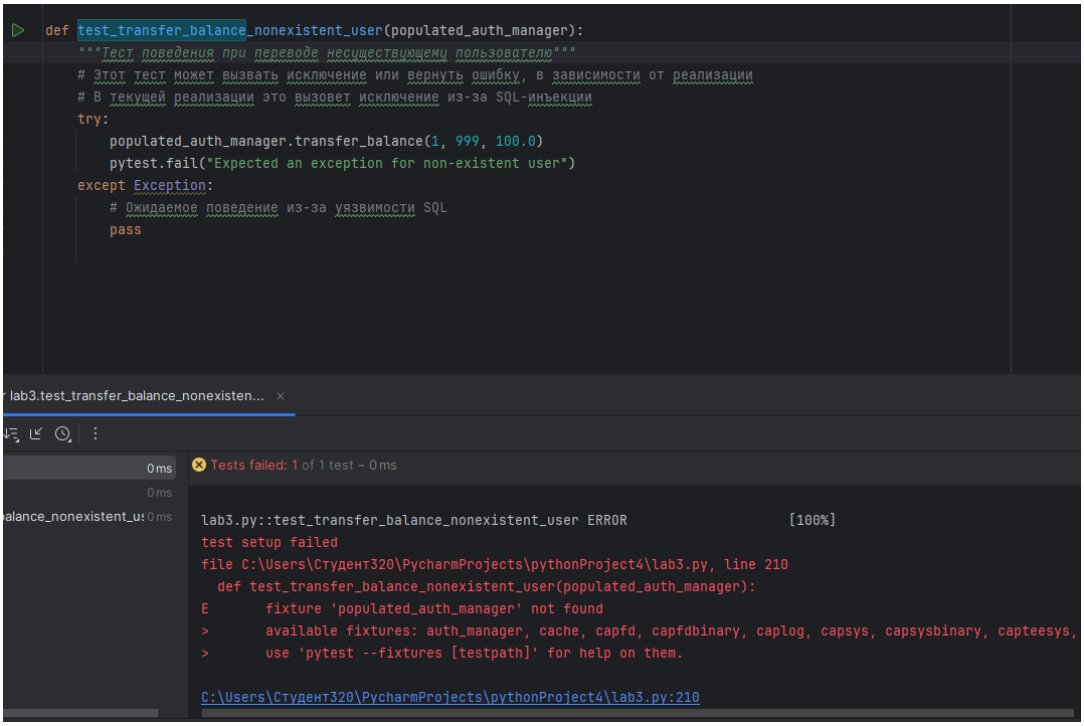


Тест 3

3. Тестирование исключений (2 штуки)

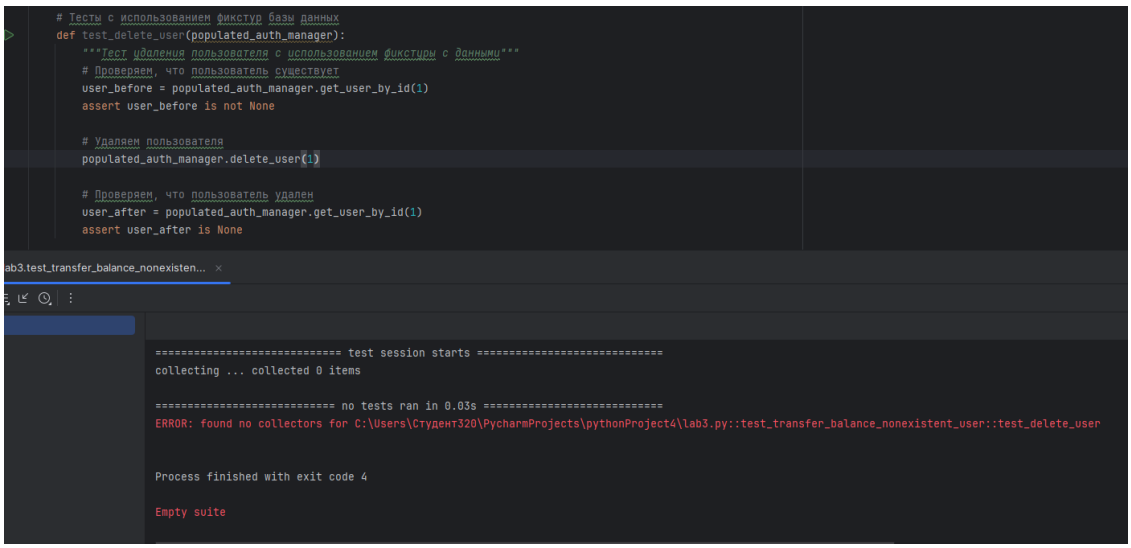


Тест 1



Тест 2

4. Тесты с использованием фикстур базы данных



Тест 1

```python
    def test_transfer_balance_success(populated_auth_manager):
        """Тест успешного перевода средств"""
        # Выполняем перевод
        populated_auth_manager.transfer_balance(1, 2, 200.0)

        # Проверяем балансы
        user1 = populated_auth_manager.get_user_by_id(1)
        user2 = populated_auth_manager.get_user_by_id(2)

        assert user1[4] == 800.0  # 1000 - 200
        assert user2[4] == 700.0  # 500 + 200
```

```
or lab3.test_transfer_balance_nonexisten...

============================= test session starts ==============================
collecting ... collected 0 items

============================ no tests ran in 0.01s =============================
ERROR: found no collectors for C:\Users\Студент320\PycharmProjects\pythonProject4\lab3.py::test_transfer_balance_nonexistent_user::test_transfer_balance_success


Process finished with exit code 4

Empty suite
```

Тест 2

5. Тесты с метками (минимум 2)

```python
# Тесты с метками (минимум 2)
@pytest.mark.slow
def test_performance_large_number_of_users(auth_manager):
    """Тест производительности с большим количеством пользователей (медленный тест)"""
    for i in range(1000):
        auth_manager.register_user( username: f"user{i}", password: f"pass{i}", country: f"Country{i % 10}", i * 10)

    count = auth_manager.count_users_by_country("Country0")
    assert count == 100  # 1000 пользователей / 10 стран
```

```
lab3.test_performance_large_number_...

23 ms  Tests passed: 1 of 1 test – 23 ms

Testing started at 11:09 ...
Launching pytest with arguments lab3.py::test_performance_large_number_of_users --no-header --no-summary -q in C:\Users\Студент320\PycharmProjects\pythonProject4

============================= test session starts ==============================
collecting ... collected 1 item

lab3.py::test_performance_large_number_of_users PASSED                   [100%]

======================== 1 passed, 2 warnings in 0.10s =========================

Process finished with exit code 0
```

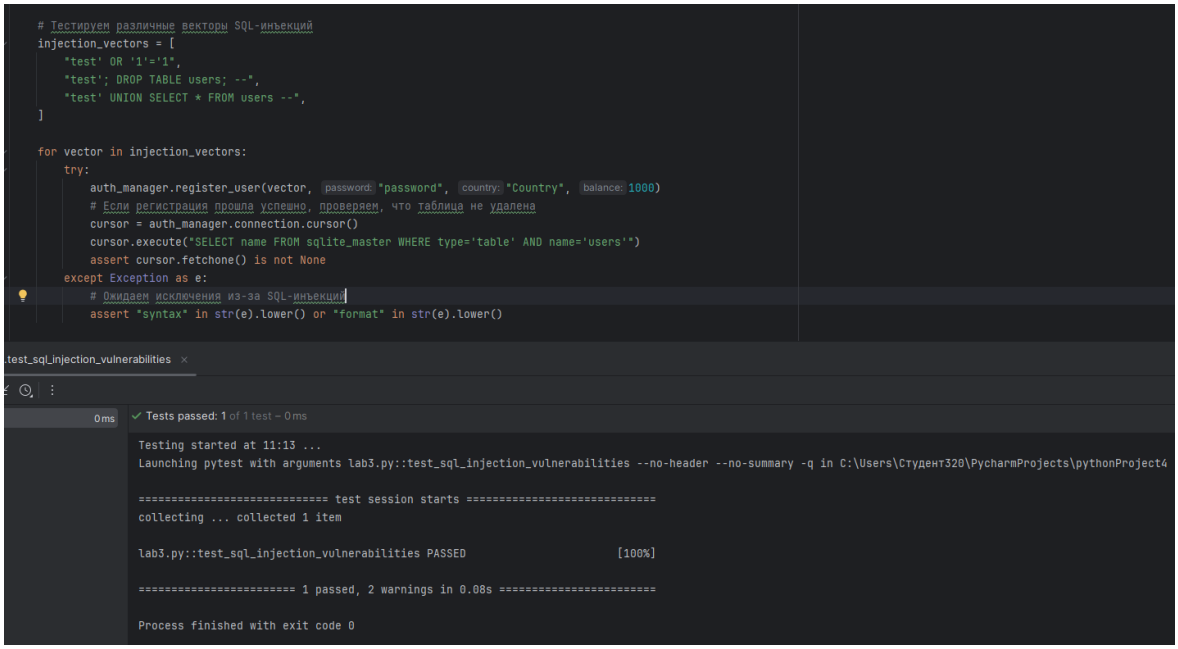
Тест 1

```python
@pytest.mark.security
def test_sql_injection_vulnerabilities(auth_manager):
    """Тест уязвимостей SQL-инъекции (тест безопасности)"""
```

```
lab3.test_sql_injection_vulnerabilities

0 ms  Tests passed: 1 of 1 test – 0 ms

Testing started at 11:11 ...
Launching pytest with arguments lab3.py::test_sql_injection_vulnerabilities --no-header --no-summary -q in C:\Users\Студент320\PycharmProjects\pythonProject4

============================= test session starts ==============================
collecting ... collected 1 item

lab3.py::test_sql_injection_vulnerabilities PASSED                       [100%]

======================== 1 passed, 2 warnings in 0.02s =========================

Process finished with exit code 0
```

Тест 2

6. Тестируем различные векторы SQL-инъекций



Тест 1