# Отчёт

# По рк-2

# Дисциплина «Парадигмы и конструкции языков программирования»

Студент: Якубович Анна

Группа: ИБМЗ-23Б

Преподаватель: Гапанюк Ю.Е.

# Задание № 1:

## Задание:

Рубежный контроль представляет собой разработку тестов на языке Python.

1) Проведите рефакторинг текста программы рубежного контроля №1 таким образом,

чтобы он был пригоден для модульного тестирования.

2) Для текста программы рубежного контроля №1 создайте модульные тесты с

применением TDD - фреймворка (3 теста).

Вариант №26:
Предметная область: Студенческая группа и учебный курс

## Код программы:

```
students = [
    [1, 'Иванов', 4.5, 1],
    [2, 'Петров', 3.8, 2],
    [3, 'Сидоров', 4.0, 3],
]


courses = [
    [1, 'Математика'],
    [2, 'Программирование'],
    [3, 'Физика'],
]
def show_students():
    """Показать всех студентов"""
    print("\nВсе студенты:")
    for student in students:
        print(f"{student[1]} - GPA: {student[2]}, Курс: {student[3]}")
def show_courses():
    """Показать все курсы"""
    print("\nВсе курсы:")
```

```
    for course in courses:
        print(f"{course[0]}: {course[1]}")
def find_student(name):
    """Найти студента по имени"""
    for s in students:
        if s[1] == name:
            return s
    return None
def calculate_avg_gpa():
    """Посчитать средний GPA всех студентов"""
    total = 0
    for s in students:
        total += s[2]
    return total / len(students)
def main_program():
    print("=" * 30)
    print("ПРОСТАЯ ПРОГРАММА")
    print("=" * 30)

    show_students()
    show_courses()

    avg = calculate_avg_gpa()
def run_simple_tests():
    """Запускаем простые тесты"""
    print("\n" + "=" * 30)
    print("ТЕСТИРОВАНИЕ")
    print("=" * 30)

    # Тест 1: Проверяем данные
    print("\n[Тест 1] Проверяем данные...")
    assert len(students) > 0, "Должен быть хотя бы один студент"
    assert len(courses) > 0, "Должен быть хотя бы один курс"
    print("✅ Тест 1 пройден!")
```

```python
    # Тест 2: Проверяем поиск студента
    print("\n[Тест 2] Проверяем поиск студента...")
    ivanov = find_student('Иванов')
    assert ivanov is not None, "Студент Иванов должен существовать"
    assert ivanov[2] == 4.5, "GPA Иванова должен быть 4.5"
    print("✅Тест 2 пройден!")

    # Тест 3: Проверяем средний GPA
    print("\n[Тест 3] Проверяем средний GPA...")
    avg = calculate_avg_gpa()
    assert avg > 0, "Средний GPA должен быть больше 0"
    assert avg < 5, "Средний GPA должен быть меньше 5"
    print(f"✅Тест 3 пройден! Средний GPA: {avg:.2f}")

    # Тест 4: Проверяем несуществующего студента
    print("\n[Тест 4] Проверяем несуществующего студента...")
    nobody = find_student('Неизвестный')
    assert nobody is None, "Несуществующего студента быть не должно"
    print("✅Тест 4 пройден!")

    print("\n" + "=" * 30)
    print("ВСЕ ТЕСТЫ ПРОЙДЕНЫ УСПЕШНО!")
    print("=" * 30)
if __name__ == "__main__":
    print("Добро пожаловать!")
    print("\nЧто вы хотите сделать?")
    print("1. Запустить программу")
    print("2. Запустить тесты")
    print("3. Сделать всё")

    choice = input("\nВведите номер (1/2/3): ")

    if choice == "1":
        main_program()
    elif choice == "2":
```

```
        run_simple_tests()
    else:
        main_program()
        run_simple_tests()


    print("\nПрограмма завершена!")
```

Пример работы программы:

```
==============================
ПРОСТАЯ ПРОГРАММА
==============================

Все студенты:
Иванов - GPA: 4.5, Курс: 1
Петров - GPA: 3.8, Курс: 2
Сидоров - GPA: 4.0, Курс: 3

Все курсы:
1: Математика
2: Программирование
3: Физика

Средний GPA всех студентов: 4.10

==============================
ТЕСТИРОВАНИЕ
==============================

[Тест 1] Проверяем данные...
✅ Тест 1 пройден!

[Тест 2] Проверяем поиск студента...
✅ Тест 2 пройден!

[Тест 3] Проверяем средний GPA...
✅ Тест 3 пройден! Средний GPA: 4.10

[Тест 4] Проверяем несуществующего студента...
✅ Тест 4 пройден!

==============================
ВСЕ ТЕСТЫ ПРОЙДЕНЫ УСПЕШНО!
==============================

Программа завершена!
```