Рубежный контроль №2 по курсу ПиК ЯП.

Хорошаева Александра ИБМ3-34Б

26 вариант.

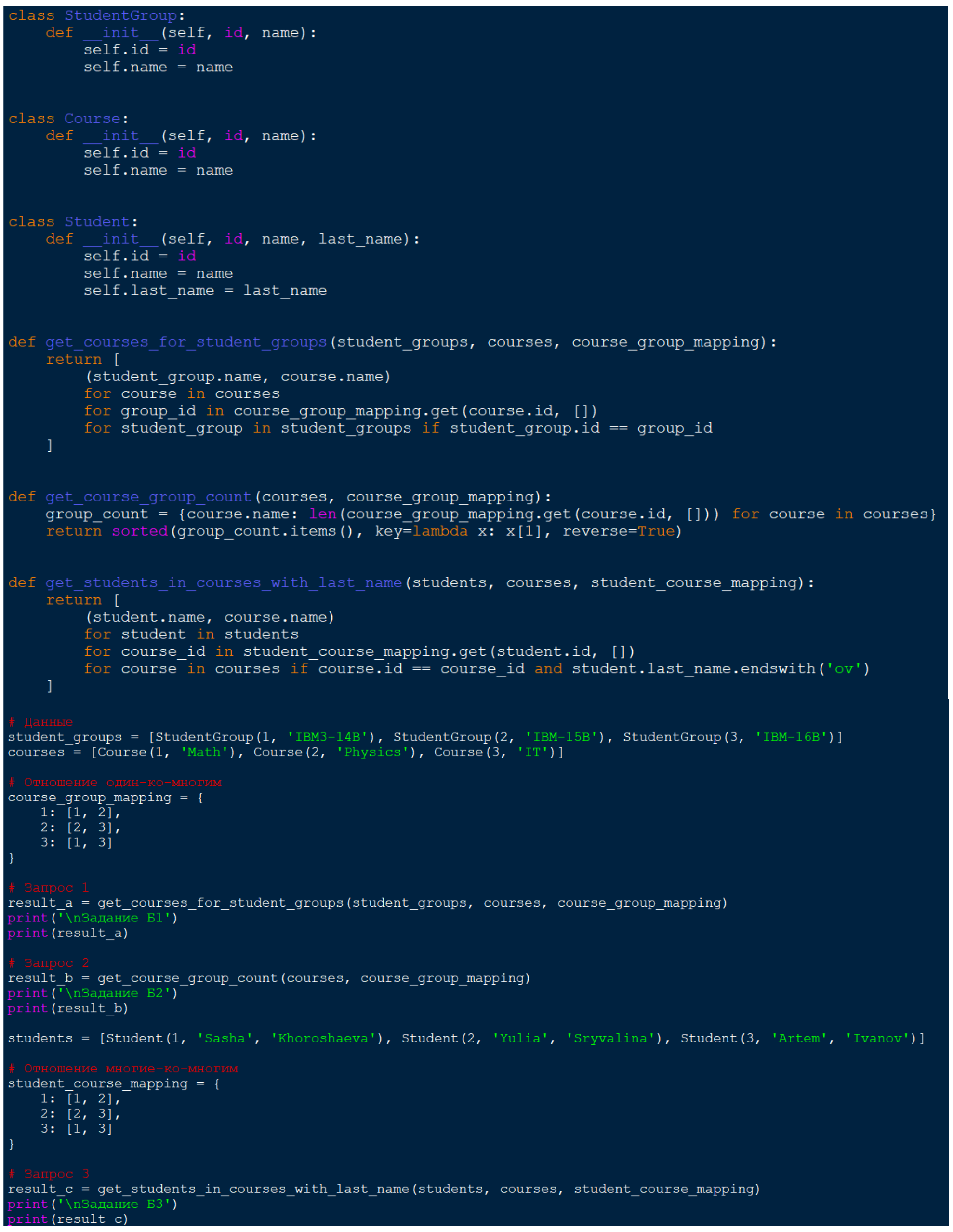
Условие задания.

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.

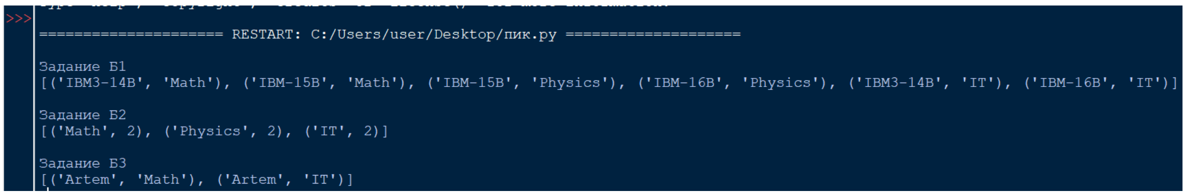
2) Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

1. Реализация рефакторинга текста программы из РК1.



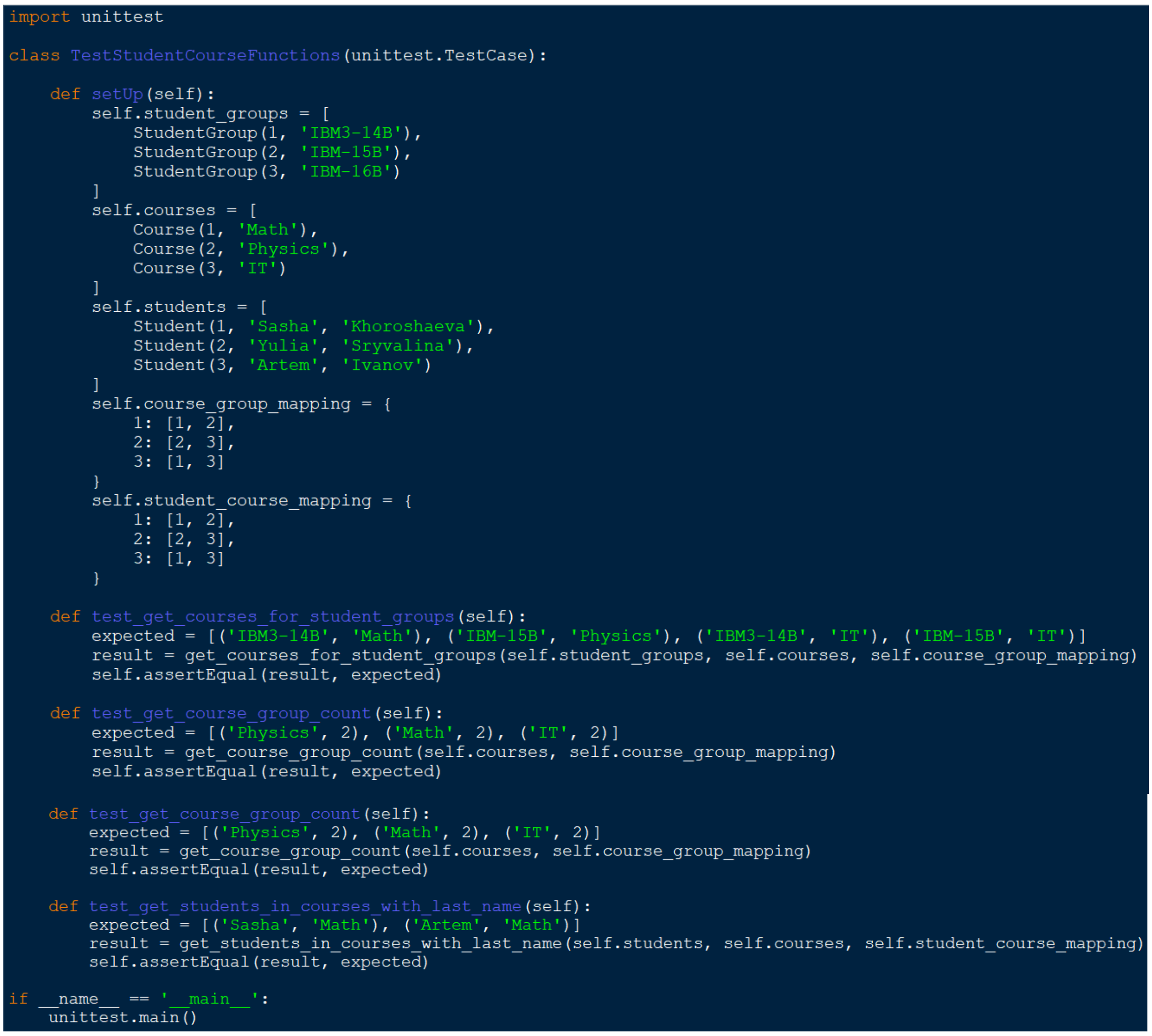
```python
class StudentGroup:
    def __init__(self, id, name):
        self.id = id
        self.name = name


class Course:
    def __init__(self, id, name):
        self.id = id
        self.name = name


class Student:
    def __init__(self, id, name, last_name):
        self.id = id
        self.name = name
        self.last_name = last_name


def get_courses_for_student_groups(student_groups, courses, course_group_mapping):
    return [
        (student_group.name, course.name)
        for course in courses
        for group_id in course_group_mapping.get(course.id, [])
        for student_group in student_groups if student_group.id == group_id
    ]


def get_course_group_count(courses, course_group_mapping):
    group_count = {course.name: len(course_group_mapping.get(course.id, [])) for course in courses}
    return sorted(group_count.items(), key=lambda x: x[1], reverse=True)


def get_students_in_courses_with_last_name(students, courses, student_course_mapping):
    return [
        (student.name, course.name)
        for student in students
        for course_id in student_course_mapping.get(student.id, [])
        for course in courses if course.id == course_id and student.last_name.endswith('ov')
    ]

# Данные
student_groups = [StudentGroup(1, 'IBM3-14B'), StudentGroup(2, 'IBM-15B'), StudentGroup(3, 'IBM-16B')]
courses = [Course(1, 'Math'), Course(2, 'Physics'), Course(3, 'IT')]

# Отношение один-ко-многим
course_group_mapping = {
    1: [1, 2],
    2: [2, 3],
    3: [1, 3]
}

# Запрос 1
result_a = get_courses_for_student_groups(student_groups, courses, course_group_mapping)
print('\nЗадание Б1')
print(result_a)

# Запрос 2
result_b = get_course_group_count(courses, course_group_mapping)
print('\nЗадание Б2')
print(result_b)

students = [Student(1, 'Sasha', 'Khoroshaeva'), Student(2, 'Yulia', 'Sryvalina'), Student(3, 'Artem', 'Ivanov')]

# Отношение многие-ко-многим
student_course_mapping = {
    1: [1, 2],
    2: [2, 3],
    3: [1, 3]
}

# Запрос 3
result_c = get_students_in_courses_with_last_name(students, courses, student_course_mapping)
print('\nЗадание Б3')
print(result_c)
```

Результат выполнения программы.



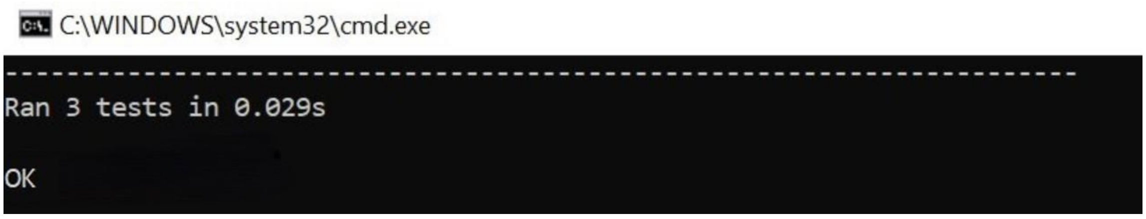
```
>>>
==================== RESTART: C:/Users/user/Desktop/пик.py ====================

Задание Б1
[('IBM3-14B', 'Math'), ('IBM-15B', 'Math'), ('IBM-15B', 'Physics'), ('IBM-16B', 'Physics'), ('IBM3-14B', 'IT'), ('IBM-16B', 'IT')]

Задание Б2
[('Math', 2), ('Physics', 2), ('IT', 2)]

Задание Б3
[('Artem', 'Math'), ('Artem', 'IT')]
```

2. Создание модульных тестов.



```python
import unittest

class TestStudentCourseFunctions(unittest.TestCase):

    def setUp(self):
        self.student_groups = [
            StudentGroup(1, 'IBM3-14B'),
            StudentGroup(2, 'IBM-15B'),
            StudentGroup(3, 'IBM-16B')
        ]
        self.courses = [
            Course(1, 'Math'),
            Course(2, 'Physics'),
            Course(3, 'IT')
        ]
        self.students = [
            Student(1, 'Sasha', 'Khoroshaeva'),
            Student(2, 'Yulia', 'Sryvalina'),
            Student(3, 'Artem', 'Ivanov')
        ]
        self.course_group_mapping = {
            1: [1, 2],
            2: [2, 3],
            3: [1, 3]
        }
        self.student_course_mapping = {
            1: [1, 2],
            2: [2, 3],
            3: [1, 3]
        }

    def test_get_courses_for_student_groups(self):
        expected = [('IBM3-14B', 'Math'), ('IBM-15B', 'Physics'), ('IBM3-14B', 'IT'), ('IBM-15B', 'IT')]
        result = get_courses_for_student_groups(self.student_groups, self.courses, self.course_group_mapping)
        self.assertEqual(result, expected)

    def test_get_course_group_count(self):
        expected = [('Physics', 2), ('Math', 2), ('IT', 2)]
        result = get_course_group_count(self.courses, self.course_group_mapping)
        self.assertEqual(result, expected)

    def test_get_course_group_count(self):
        expected = [('Physics', 2), ('Math', 2), ('IT', 2)]
        result = get_course_group_count(self.courses, self.course_group_mapping)
        self.assertEqual(result, expected)

    def test_get_students_in_courses_with_last_name(self):
        expected = [('Sasha', 'Math'), ('Artem', 'Math')]
        result = get_students_in_courses_with_last_name(self.students, self.courses, self.student_course_mapping)
        self.assertEqual(result, expected)

if __name__ == '__main__':
    unittest.main()
```

Результат проверки.



```
C:\WINDOWS\system32\cmd.exe
----------------------------------------------------------------------
Ran 3 tests in 0.029s

OK
```