



**Министерство науки и высшего образования
Российской Федерации Федеральное государственное
бюджетное образовательное учреждение высшего
образования**

**«Московский государственный технический
университет имени Н.Э. Баумана
(национальный исследовательский
университет)» (МГТУ им. Н.Э.
Баумана)**

**Факультет «Информатика и системы управления»
Кафедра «Системы обработки информации и управления»**

Рубежный контроль №2

Выполнил студент группы ИУ5-35Б

Сулайманов Р. Б.

Москва

2022 г.

Полученное 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 Д 17

17	Дирижер	Оркестр
----	---------	---------

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия заканчивается на «ов», и названия их отделов.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов со средней зарплатой сотрудников в каждом отделе, отсортированный по средней зарплате (отдельной функции вычисления среднего значения в Python нет, нужно использовать комбинацию функций вычисления суммы и количества значений).
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех отделов, у которых название начинается с буквы «Р», и список работающих в них сотрудников.

Текст программы

Программа разделена на несколько файлов.

- 1) Файл classes.py содержит классы для реализации задания:

```
class conductor:
    """дирижер"""

    def __init__(self, id, surname, salary, orch_id):
        self.id = id
        self.surname = surname
        self.salary = salary
        self.orch_id = orch_id

class orchestra:
    """оркестр"""

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

class condOrch:
    def __init__(self, id_cond, id_orch):
```

```
self.id_cond = id_cond
self.id_orch = id_orch
```

2) Файл connections.py содержит функции для создания связей один-ко-многим и многие-ко-многим:

```
def one_to_many(conductors, orchestras):
    return [(cond.surname, cond.salary, orch.name)
            for cond in conductors
            for orch in orchestras
            if cond.orch_id == orch.id]

def many_to_many(conductors, orchestras, condOrch):
    return [(cond.surname, cond.salary, orch.name)
            for oc in condOrch
            for cond in conductors
            for orch in orchestras
            if cond.id == oc.id_cond and orch.id == oc.id_orch]
```

3) Файл processing.py содержит функции для реализации требуемых заданий:

```
def task1(one_to_many):
    return [el for el in one_to_many if el[0].endswith('ов')]

def task2(one_to_many):
    orch = []
    salary = []
    for el in one_to_many:
        if el[2] not in orch:
            orch.append(el[2])
            salary.append((el[1], 1))
        else:
            idx = orch.index(el[2])
            salary[idx] = ((salary[idx][0] + el[1]) / (salary[idx][1] + 1), salary[idx][1] + 1)
    salary = [i[0] for i in salary]
    return sorted(list(zip(orch, salary)), key=lambda p: p[1], reverse=True)

def task3(many_to_many):
    orch_R = []
    cond_orch_R = []
    for el in many_to_many:
        if el[2].startswith('P'):
            if el[2] not in orch_R:
                orch_R.append(el[2])
                cond_orch_R.append((el[0],))
            else:
                cond_orch_R[orch_R.index(el[2])] += (el[0],)
    return list(zip(orch_R, cond_orch_R))
```

4) Файл testing.py реализует тестирование программы:

```
import unittest
from classes import conductor, orchestra, condOrch
from connections import one_to_many, many_to_many
from processing import task1, task2, task3

class RK_test(unittest.TestCase):
    def setUp(self):
        orchestras = [
            orchestra(1, 'Российский национальный оркестр'),
            orchestra(2, 'Ансамбль песни и пляски'),
            orchestra(3, 'Местные ребята')
        ]
        conductors = [
            conductor(1, 'Гончаренко', 15000, 1),
            conductor(2, 'Иванов', 38000, 2),
            conductor(3, 'Самоян', 12500, 3),
            conductor(4, 'Улепетов', 20000, 1),
            conductor(5, 'Бобер', 28500, 3)
        ]
        condOrchs = [
            condOrch(1, 1),
            condOrch(2, 1),
            condOrch(3, 2),
            condOrch(4, 2),
            condOrch(5, 1),
            condOrch(1, 2),
            condOrch(3, 3),
            condOrch(5, 3)
        ]
        self.one_to_many = one_to_many(conductors, orchestras)
        self.many_to_many = many_to_many(conductors, orchestras,
condOrchs)

    def test_task1(self):
        expected_result = [
            ('Иванов', 38000, 'Ансамбль песни и пляски'),
            ('Улепетов', 20000, 'Российский национальный оркестр')
        ]
        result = task1(self.one_to_many)
        self.assertEqual(result, expected_result)

    def test_task2(self):
        expected_result = [
            ('Ансамбль песни и пляски', 38000),
            ('Местные ребята', 20500),
            ('Российский национальный оркестр', 17500)
        ]
        result = task2(self.one_to_many)
        self.assertEqual(result, expected_result)

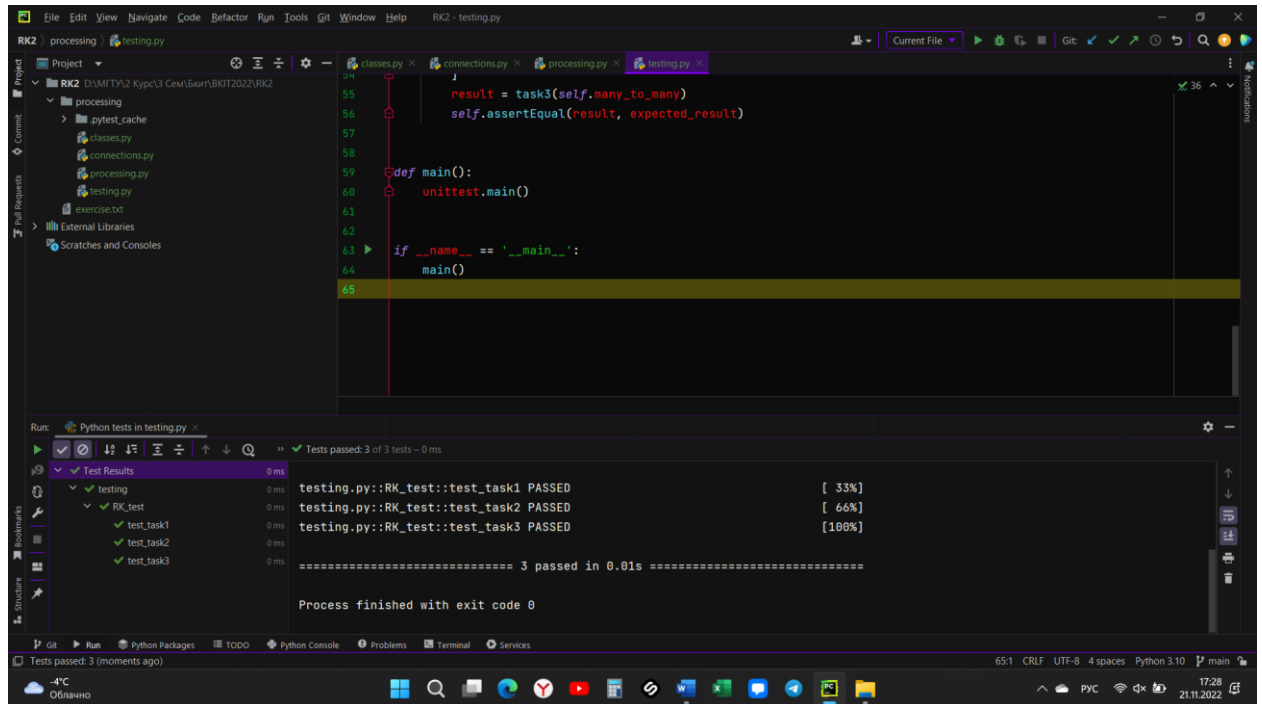
    def test_task3(self):
        expected_result = [
            ('Российский национальный оркестр', ('Гончаренко', 'Иванов',
'Бобер'))
        ]
        result = task3(self.many_to_many)
        self.assertEqual(result, expected_result)

def main():
```

```
unittest.main()

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

Работа программы



The screenshot shows an IDE window titled 'RK2 - testing.py'. The editor displays a Python script with the following code:

```
55     result = task3(self.many_to_many)
56     self.assertEqual(result, expected_result)
57
58
59 def main():
60     unittest.main()
61
62
63 if __name__ == '__main__':
64     main()
65
```

The 'Run' panel at the bottom shows the test results for 'testing.py::RK_test::test_task1', 'testing.py::RK_test::test_task2', and 'testing.py::RK_test::test_task3'. All three tests passed. The summary indicates '3 passed in 0.01s'.

Test Results:

Test Case	Duration	Result
testing.py::RK_test::test_task1	0 ms	PASSED
testing.py::RK_test::test_task2	0 ms	PASSED
testing.py::RK_test::test_task3	0 ms	PASSED

Summary: 3 passed in 0.01s

Process finished with exit code 0