

Текст программы

```
import unittest
from operator import itemgetter

class Musician:
    """Музыкант"""
    def __init__(self, id, name):
        self.id = id
        self.name = name

class Orchestra:
    """Оркестр"""
    def __init__(self, id, name, salary, mus_id):
        self.id = id
        self.name = name
        self.salary = salary
        self.mus_id = mus_id

class MusOrc:
    """Связь многие-ко-многим"""
    def __init__(self, mus_id, orc_id):
        self.mus_id = mus_id
        self.orc_id = orc_id

# Данные
musicians = [
    Musician(1, "Моцарт"),
    Musician(2, "Шостакович"),
    Musician(3, "Глинка"),
]

orcs = [
    Orchestra(1, "Королевский симфонический оркестр", 50000, 1),
    Orchestra(2, "Гвардейский военный оркестр", 40000, 2),
    Orchestra(3, "Blue Note Band", 45000, 3),
```

```
Orchestra(4, "Русская балалайка", 30000, 3),  
Orchestra(5, "Дыхание весны", 35000, 1),  
Orchestra(6, "Филармонический оркестр Санкт-Петербурга", 60000, 2),  
]
```

```
mus_orc = [  
    MusOrc(1, 1),  
    MusOrc(2, 2),  
    MusOrc(3, 3),  
    MusOrc(1, 4),  
    MusOrc(2, 5),  
    MusOrc(3, 6),  
]
```

```
def first_task(one_to_many):  
    """A1. Список всех музыкантов и оркестров, отсортированный по  
    оркестрам."""
```

```
    # Сортируем по названию оркестра (x[0] - это название оркестра)  
    sorted_data = sorted(one_to_many, key=lambda x: x[0].lower())  
    return sorted_data
```

```
def second_task(orce, musicians):
```

```
    """A2. Список музыкантов с суммарной зарплатой оркестров,  
    отсортированный по суммарной зарплате."""
```

```
    musician_salary = {}  
    for musician in musicians:  
        musician_orchestras = [orc.salary for orc in orce if orc.mus_id == musician.id]  
        total_salary = sum(musician_orchestras)  
        musician_salary[musician.name] = total_salary  
    result = sorted(musician_salary.items(), key=itemgetter(1), reverse=True)  
    return result
```

```
def third_task(many_to_many):
```

```
    """A3. Список оркестров с 'оркестр' в названии и музыкантов в них."""  
    filtered_data = [(orc_name, musician) for orc_name, _, musician in  
many_to_many if 'оркестр' in orc_name.lower()]  
    return filtered_data
```

# Тесты

```
class TestTasks(unittest.TestCase):
```

```
    def setUp(self):
```

```
        """Подготовка данных для тестов."""
```

```
        # Связь один-ко-многим
```

```
        self.one_to_many = [(orc.name, orc.salary, musician.name)
                             for musician in musicians
                             for orc in orcs
                             if orc.mus_id == musician.id]
```

```
        # Связь многие-ко-многим
```

```
        many_to_many_temp = [(musician.name, mo.mus_id, mo.orc_id)
                               for musician in musicians
                               for mo in mus_orc
                               if musician.id == mo.mus_id]
        self.many_to_many = [(orc.name, orc.salary, musician_name)
                              for musician_name, mus_id, orc_id in many_to_many_temp
                              for orc in orcs if orc.id == orc_id]
```

```
    def test_first_task(self):
```

```
        """Тест для задания А1."""
```

```
        result = first_task(self.one_to_many)
```

```
        expected = [
```

```
            ('Blue Note Band', 45000, 'Глинка'),
            ('Гвардейский военный оркестр', 40000, 'Шостакович'),
            ('Дыхание весны', 35000, 'Моцарт'),
            ('Королевский симфонический оркестр', 50000, 'Моцарт'),
            ('Русская балалайка', 30000, 'Глинка'),
            ('Филармонический оркестр Санкт-Петербурга', 60000, 'Шостакович'),
```

```
        ]
```

```
        self.assertEqual(result, expected)
```

```
    def test_second_task(self):
```

```
        """Тест для задания А2."""
```

```
        result = second_task(orcs, musicians)
```

```
        expected = [
```

```
            ('Шостакович', 100000),
            ('Моцарт', 85000),
            ('Глинка', 75000),
```

```

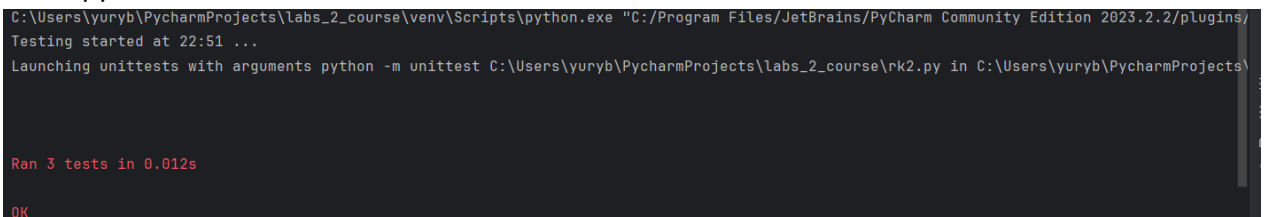
    ]
    self.assertEqual(result, expected)

def test_third_task(self):
    """Тест для задания А3."""
    result = third_task(self.many_to_many)
    expected = [
        ('Королевский симфонический оркестр', 'Моцарт'),
        ('Гвардейский военный оркестр', 'Шостакович'),
        ('Филармонический оркестр Санкт-Петербурга', 'Глинка'),
    ]
    self.assertEqual(result, expected)

if __name__ == "__main__":
    unittest.main()

```

## Вывод:



A screenshot of a terminal window with a dark background. The text is white and red. It shows the execution of a Python script using unittest. The output includes the file path, the start time, and the command used to launch the tests. At the bottom, it reports that 3 tests were run successfully in 0.012 seconds. An 'OK' button is visible in the bottom left corner.

```

C:\Users\yuryb\PycharmProjects\labs_2_course\venv\Scripts\python.exe "C:/Program Files/JetBrains/PyCharm Community Edition 2023.2.2/plugins/
Testing started at 22:51 ...
Launching unittests with arguments python -m unittest C:\Users\yuryb\PycharmProjects\labs_2_course\rk2.py in C:\Users\yuryb\PycharmProjects\

Ran 3 tests in 0.012s

OK

```