Московский государственный технический
университет им. Н.Э. Баумана

Факультет Информатика и системы управления
Кафедра ИУ5 «Системы обработки информации и управления»

Дисциплина «Парадигмы и конструкции языков программирования»

Отчет по рубежному контролю №2

«Модульное тестирование»

Вариант №5

Запрос А.

Выполнил:
студент группы ИУ5-31Б:
Есакова О. А.
Подпись и дата:

Проверил:
преподаватель каф.ИУ5
Гапанюк Ю. Е.
Подпись и дата:

Москва, 2025 г.

Файл main.py:

```
import requests

def main():
    print("Задача A1:")
    for item in requests.get_A1():
        print(item)
    print("\nЗадача A2:")
    for item in requests.get_A2():
        print(item)
    print("\nЗадача A3:")
    for key, value in requests.get_A3().items():
        print(f"{key}: {value}")

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

Файл data.py:

```
from classes import Orchestra, Musician, MusiciansOrchestras

orchestras = [
    #id оркестра, название оркестра
    Orchestra(1, 'симфонический оркестр'),
    Orchestra(2, 'джазовый оркестр'),
    Orchestra(3, 'камерный оркестр'),
    Orchestra(4, 'народных инструментов'),
    Orchestra(5, 'духовой оркестр'),]

musicians = [
    #id музыканта, имя, зарплата, id оркестра
    Musician(1, 'Алексей', 30000, 1),
    Musician(2, 'Николай', 35000, 2),
    Musician(3, 'Иван', 40000, 3),
    Musician(4, 'Анастасия', 27000, 1),
    Musician(5, 'Мария', 25000, 4),]
#список объектов класса MusiciansOrchestras со связью м:м
#один музыкант может играть в нескольких оркестрах, в одном оркестре могут
играть разные музыканты
musicians_orchestras = [
    #(оркестр-музыкант)
    MusiciansOrchestras(1, 1),
    MusiciansOrchestras(1, 4),
    MusiciansOrchestras(2, 2),
    MusiciansOrchestras(2, 4),
    MusiciansOrchestras(3, 3),
    MusiciansOrchestras(4, 5),
    MusiciansOrchestras(5, 5),]
```

Файл classes.py:

```
class Orchestra: #класс "оркестр"
    def __init__(self, id, name):
        self.id = id
        self.name = name

class Musician: #класс "музыкант"
```

```
    def __init__(self, id, name, salary, orchestra_id):
        self.id = id
        self.name = name
        self.salary = salary
        self.orchestra_id = orchestra_id

class MusiciansOrchestras: #для множественной связи м:м
    def __init__(self, orchestra_id, musician_id):
        self.orchestra_id = orchestra_id
        self.musician_id = musician_id
```

Файл test_requests1.py:

```
import unittest
from requests import get_A1, get_A2, get_A3

class TestRequests(unittest.TestCase):

    def test_A1(self):
        result = get_A1()
        self.assertIsInstance(result, list)
        self.assertTrue(result, "нужен не пустой список")
        self.assertIsInstance(result[0], tuple)
        self.assertEqual(len(result[0]), 3)

    def test_A2(self):
        result = get_A2()
        self.assertIsInstance(result, list)
        self.assertTrue(result, "нужен не пустой список")
        salaries = [item[1] for item in result]
        self.assertEqual(salaries, sorted(salaries, reverse=True))
        if len(salaries) > 1:
            self.assertGreaterEqual(salaries[0], salaries[-1])

    def test_A3(self):
        result = get_A3()
        self.assertIsInstance(result, dict)
        for key in result:
            self.assertIn('оркестр', key)
            self.assertIsInstance(result[key], list)
```

Файл requests.py:

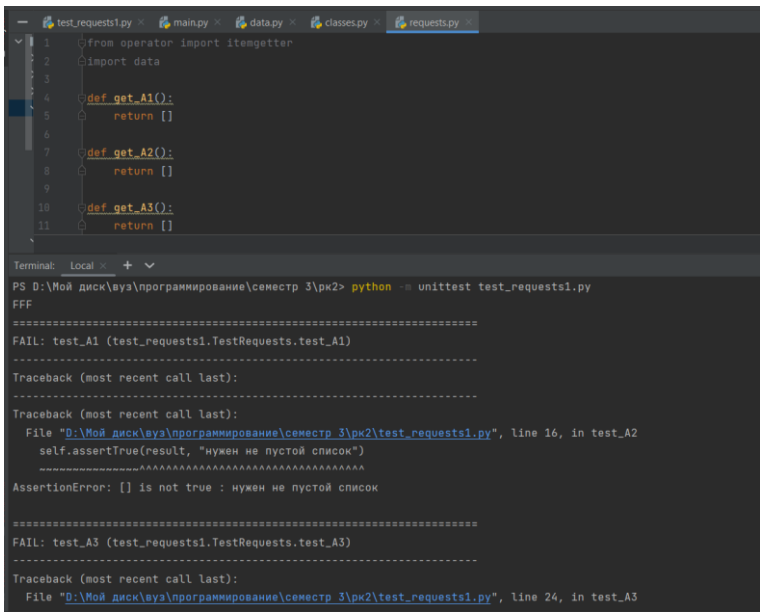
1. **Случай ошибки:**

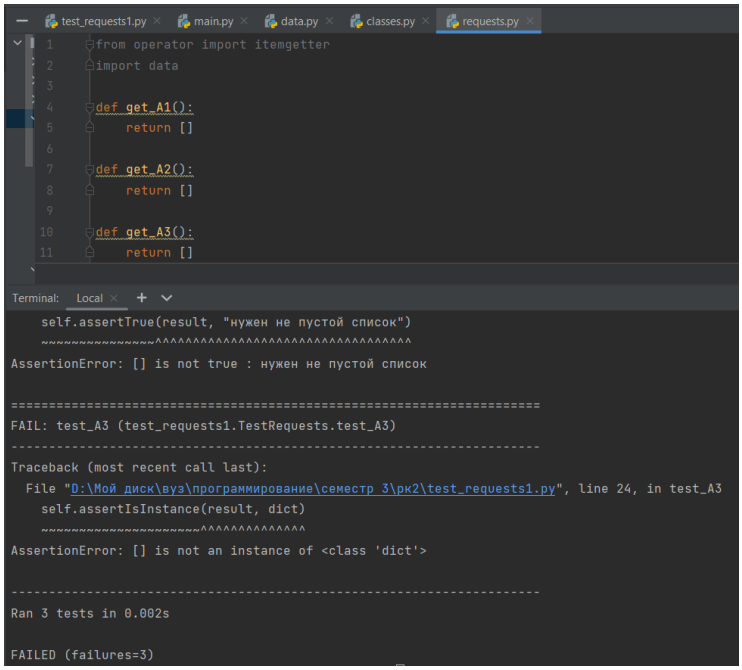
```
from operator import itemgetter
import data

def get_A1():
    return []

def get_A2():
    return []

def get_A3():
    return []
```





2. **Нет ошибки:**

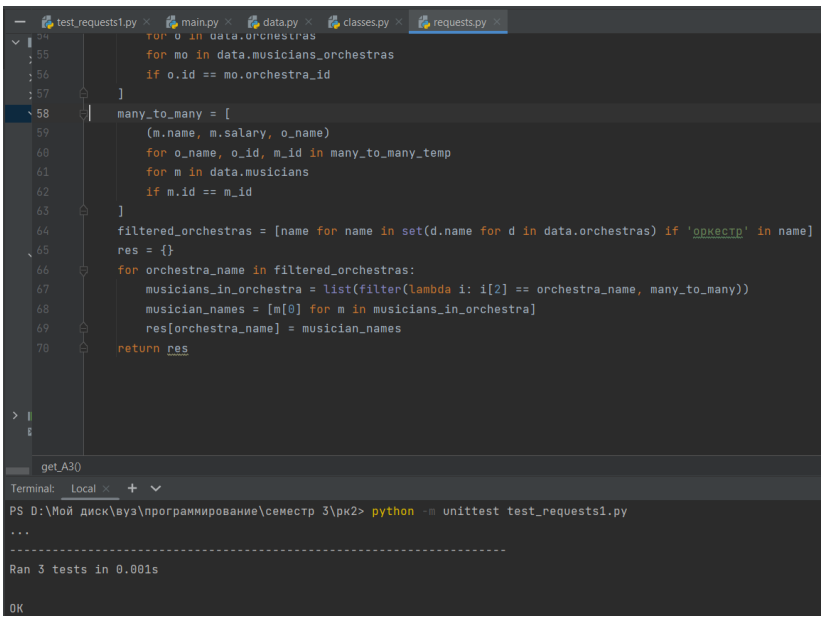
```
from operator import itemgetter
import data

def get_A1():
    one_to_many = [(m.name, m.salary, o.name)
                   for o in data.orchestras
                   for m in data.musicians
                   if m.orchestra_id == o.id]
    sorted_result = sorted(one_to_many, key=itemgetter(2))
    return sorted_result

def get_A2():
```

```
    one_to_many = [(m.name, m.salary, o.name)
                   for o in data.orchestras
                   for m in data.musicians
                   if m.orchestra_id == o.id]
    res = []
    for o in data.orchestras:
        orch_musicians = list(filter(lambda i: i[2] == o.name, one_to_many))
        if len(orch_musicians) > 0:
            total_salary = sum(m[1] for m in orch_musicians)
            res.append((o.name, total_salary))
    sorted_res = sorted(res, key=itemgetter(1), reverse=True)
    return sorted_res

def get_A3():
    many_to_many_temp = [
        (o.name, mo.orchestra_id, mo.musician_id)
        for o in data.orchestras
        for mo in data.musicians_orchestras
        if o.id == mo.orchestra_id
    ]
    many_to_many = [
        (m.name, m.salary, o_name)
        for o_name, o_id, m_id in many_to_many_temp
        for m in data.musicians
        if m.id == m_id
    ]
    filtered_orchestras = [name for name in set(d.name for d in data.orchestras) if 'оркестр' in
name]
    res = {}
    for orchestra_name in filtered_orchestras:
        musicians_in_orchestra = list(filter(lambda i: i[2] == orchestra_name, many_to_many))
        musician_names = [m[0] for m in musicians_in_orchestra]
        res[orchestra_name] = musician_names
    return res
```


test_requests1.py
main.py
data.py
classes.py
requests.py
for o in data.orchestras
for mo in data.musicians_orchestras
if o.id == mo.orchestra_id
]
many_to_many = [
(m.name, m.salary, o_name)
for o_name, o_id, m_id in many_to_many_temp
for m in data.musicians
if m.id == m_id
]
filtered_orchestras = [name for name in set(d.name for d in data.orchestras) if 'оркестр' in name]
res = {}
for orchestra_name in filtered_orchestras:
musicians_in_orchestra = list(filter(lambda i: i[2] == orchestra_name, many_to_many))
musician_names = [m[0] for m in musicians_in_orchestra]
res[orchestra_name] = musician_names
return res
get_A3()
Terminal:
Local
PS D:\Мой диск\вуз\программирование\семестр 3\рк2> python -m unittest test_requests1.py
...
Ran 3 tests in 0.001s
OK