

Листинг кода:

```
from operator import itemgetter

class Dir:

    "Дирижёр"

    def __init__(self, id, fio, experience, orch_id):

        self.id = id

        self.fio = fio

        self.experience = experience

        self.orch_id = orch_id


class Orch:

    "Оркестр"

    def __init__(self, id, name):

        self.id = id

        self.name = name


class DirOrch:

    "Дирижёры оркестра (связь многие-ко-многим)"

    def __init__(self, dir_id, orch_id):

        self.dir_id = dir_id

        self.orch_id = orch_id


orchestras = [

    Orch(1, "Симфонический оркестр Москвы"),

    Orch(2, "Филармонический оркестр Санкт-Петербурга"),

    Orch(3, "Камерный оркестр России"),

    Orch(4, "Большой оркестр"),

    Orch(5, "Джазовый оркестр города")

]

conductors = [

    Dir(1, "Артамонов", 15, 1),
```

```

Dir(2, "Петров", 20, 2),
Dir(3, "Иваненко", 10, 3),
Dir(4, "Иванов", 12, 3),
Dir(5, "Иванин", 8, 3),
]
dirs_orchs = [
DirOrch(1, 1),
DirOrch(2, 2),
DirOrch(3, 3),
DirOrch(4, 3),
DirOrch(5, 3),
DirOrch(1, 4),
DirOrch(2, 5),
DirOrch(3, 5),
]
def main():
    "Основная функция"
    one_to_many = [(d.fio, d.experience, o.name)
for o in orchestras
for d in conductors
if d.orch_id == o.id]
    many_to_many_temp = [(o.name, do.orch_id, do.dir_id)
for o in orchestras
for do in dirs_orchs
if o.id == do.orch_id]
    many_to_many = [(d.fio, d.experience, orch_name)
for orch_name, orch_id, dir_id in many_to_many_temp
for d in conductors if d.id == dir_id]
    print("Задание A1")
    res_1 = sorted(one_to_many, key=itemgetter(2))
    for r in res_1:

```

```

print(r)

print("\nЗадание A2")

res_2_unsorted = []

for o in orchestras:

    d_list = list(filter(lambda x: x[2] == o.name, one_to_many))

    if len(d_list) > 0:

        exp_list = [exp for _, exp, _ in d_list]

        exp_sum = sum(exp_list)

        res_2_unsorted.append((o.name, exp_sum))

res_2 = sorted(res_2_unsorted, key=itemgetter(1), reverse=True)

for r in res_2:

    print(r)

print("\nЗадание A3")

res_3 = {}

for o in orchestras:

    if "оркестр" in o.name.lower():

        o_dirs = list(filter(lambda x: x[2] == o.name, many_to_many))

        o_dir_names = [x for x, _, _ in o_dirs]

        res_3[o.name] = o_dir_names

for k, v in res_3.items():

    print(f'{k}: {', '.join(v)}')

if __name__ == "__main__":

    main()

```

Результаты выполнения программы

Задание A1

Список всех связанных дирижёров и оркестров, отсортированный по оркестрам:

```
[('Петров', 8, 'Камерный оркестр'),  
( 'Артамонов', 5, 'Симфонический оркестр'),  
( 'Иваненко', 12, 'Эстрадный оркестр'),  
( 'Иванов', 9, 'Эстрадный оркестр'),  
( 'Сидоров', 4, 'Военный оркестр')]
```

Задание A2

Список оркестров с суммарным стажем дирижёров, отсортированный по суммарному стажу:

```
[('Эстрадный оркестр', 21),  
( 'Камерный оркестр', 8),  
( 'Симфонический оркестр', 5),  
( 'Военный оркестр', 4)]
```

Задание A3

Список оркестров, в названии которых есть слово "оркестр", и дирижёров, работающих в них:

```
{  
'Симфонический оркестр': ['Артамонов'],  
'Камерный оркестр': ['Петров'],  
'Эстрадный оркестр': ['Иваненко', 'Иванов'],  
'Военный оркестр': ['Сидоров'],  
'Оркестр народных инструментов': ['Артамонов', 'Иваненко']  
}
```

Пояснения к решению

1. Классы данных:

- Conductor — дирижёр (id, имя, стаж, id оркестра);
- Orchestra — оркестр (id, название);
- ConductorOrchestra — связь многие-ко-многим.

2. Тестовые данные:

- 5 оркестров и 5 дирижёров;
- связи реализованы корректно через идентификаторы.

3. Запросы варианта А:

- a. **A1:** Вывод всех связанных дирижёров и оркестров (один-ко-многим), сортировка по оркестрам.
- b. **A2:** Подсчёт суммарного стажа дирижёров в каждом оркестре, сортировка по убыванию.
- c. **A3:** Отбор оркестров, в названии которых есть слово "оркестр", и вывод дирижёров, в них работающих (многие-ко-многим).