Буйдина Кристина Анатольевна

ИУ5-33Б

РК1

```
#Вариант В.
#«Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия начинается с буквы «А», и названия их отделов.
#«Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов с минимальной зарплатой сотрудников в каждом отделе, отсортированный по минимальной зарплате.
#«Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по сотрудникам, сортировка по отделам произвольная.
#5 Музыкант Оркестр



# используется для сортировки
from operator import itemgetter

class Musician:
    """Музыкант"""

    def __init__(self, id, fio, sal, orch_id):
        self.id = id
        self.fio = fio
        self.sal = sal
        self.orch_id = orch_id


class Orchestra:
    """Оркестр"""

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


class MusOrch:
    """
    'Музыканты Оркестра' для реализации
    связи многие-ко-многим
    """

    def __init__(self, orch_id, mus_id):
        self.orch_id = orch_id
        self.mus_id = mus_id


# Отделы
orchestras = [
    Orchestra(1, 'деревянные духовые'),
    Orchestra(2, 'медные духовые'),
    Orchestra(3, 'струнные смычковые'),
    Orchestra(4, 'ударные'),

    Orchestra(11, 'деревянные духовые (другое)'),
    Orchestra(22, 'медные духовые (другое)'),
    Orchestra(33, 'струнные смычковые (другое)'),
    Orchestra(44, 'ударные (другое)'),
]
```

```python
# Сотрудники
musicians = [
    Musician(1, 'Артамонов', 25000, 1),
    Musician(2, 'Петров', 35000, 2),
    Musician(3, 'Иваненко', 45000, 3),
    Musician(4, 'Иванов', 35000, 3),
    Musician(5, 'Сушкин', 25000, 4),
]

musicians_orchestras = [
    MusOrch(1, 1),
    MusOrch(2, 2),
    MusOrch(3, 3),
    MusOrch(3, 4),
    MusOrch(4, 5),

    MusOrch(11, 1),
    MusOrch(22, 2),
    MusOrch(33, 3),
    MusOrch(33, 4),
    MusOrch(44, 5),
]


def main():
    """Основная функция"""

    # Соединение данных один-ко-многим
    one_to_many = [(e.fio, e.sal, d.name)
                   for d in orchestras
                   for e in musicians
                   if e.orch_id == d.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(d.name, ed.orch_id, ed.mus_id)
                         for d in orchestras
                         for ed in musicians_orchestras
                         if d.id == ed.orch_id]

    many_to_many = [(e.fio, e.sal, dep_name)
                    for dep_name, dep_id, emp_id in many_to_many_temp
                    for e in musicians if e.id == emp_id]

    print('Задание А1')

    res_11 = sorted([(fio, sal, name) for fio, sal, name in one_to_many if
fio.startswith('А')], key=itemgetter(2))
    for i in res_11:
        print(i)

    print('\nЗадание А2')
    res_12_unsorted = []
    # Перебираем все отделы оркестра
    for d in orchestras:
        # Список музыкантов отдела оркестра
        d_emps = list(filter(lambda i: i[2] == d.name, one_to_many))
        # Если отдел не пустой
        if len(d_emps) > 0:
            # Зарплаты музыкантов отдела оркестра
            d_sals = [sal for _, sal, _ in d_emps]
            # Суммарная зарплата музыкантов отдела оркестра
            d_sals_min = min(d_sals)
            res_12_unsorted.append((d.name, d_sals_min))
```

```
    # Сортировка по суммарной зарплате
    res_12 = sorted(res_12_unsorted, key=itemgetter(1),)
    print(res_12)

    print('\nЗадание А3')
    res_13 = {}
    sorted_many_to_many = sorted(many_to_many, key=itemgetter(0))

    # Перебираем все отделы оркестра
    for d in orchestras:
        # Список музыкантов отдела оркестра
        d_emps = list(filter(lambda i: i[2] == d.name, many_to_many))
        # Только ФИО музыкантов
        d_emps_names = [x for x, _, _ in d_emps]
        # Добавляем результат в словарь
        # ключ - отдел оркестра, значение - список фамилий
        res_13[d.name] = d_emps_names
    print(res_13)


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