Постановка задачи

1) Необходимо создать два класса данных в соответствии с Вашим вариантом предметной области, которые связаны отношениями один-ко-многим и многие-ко-многим.

Пример классов данных для предметной области Сотрудник-Отдел:

1. Класс «Сотрудник», содержащий поля:
   - ID записи о сотруднике;
   - Фамилия сотрудника;
   - Зарплата (количественный признак);
   - ID записи об отделе. (для реализации связи один-ко-многим)
2. Класс «Отдел», содержащий поля:
   - ID записи об отделе;
   - Наименование отдела.
3. (Для реализации связи многие-ко-многим) Класс «Сотрудники отдела», содержащий поля:
   - ID записи о сотруднике;
   - ID записи об отделе.

2) Необходимо создать списки объектов классов, содержащих тестовые данные (3-5 записей), таким образом, чтобы первичные и вторичные ключи соответствующих записей были связаны по идентификаторам.

3) Необходимо разработать запросы в соответствии с Вашим вариантом. Запросы сформулированы в терминах классов «Сотрудник» и «Отдел», которые используются в примере. Вам нужно перенести эти требования в Ваш вариант предметной области. При разработке запросов необходимо по возможности использовать функциональные возможности языка Python (list/dict comprehensions, функции высших порядков).

Для реализации запроса №2 введите в класс, находящийся на стороне связи «много», произвольный количественный признак, например, «зарплата сотрудника».

Вариант В.

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

1	Студент	Группа

```
class Student:
    def __init__(self, student_id, surname, scholarship, group_id):
        self.student_id = student_id
        self.surname = surname
        self.scholarship = scholarship
        self.group_id = group_id

class Group:
    def __init__(self, group_id, group_name):
        self.group_id = group_id
        self.group_name = group_name

class GroupStudent:
    def __init__(self, student_id, group_id):
        self.student_id = student_id
        self.group_id = group_id

students = [
    Student(student_id=1, surname="Андреев", scholarship=1500, group_id=1),
    Student(student_id=2, surname="Борисов", scholarship=1200, group_id=1),
    Student(student_id=3, surname="Алексеева", scholarship=1800, group_id=2),
    Student(student_id=4, surname="Аркадьев", scholarship=1600, group_id=3),
    Student(student_id=5, surname="Васильева", scholarship=1100, group_id=2)
]

groups = [
    Group(group_id=1, group_name="Группа А"),
    Group(group_id=2, group_name="Группа Б"),
    Group(group_id=3, group_name="Группа В")
]

group_students = [
    GroupStudent(student_id=1, group_id=1),
    GroupStudent(student_id=2, group_id=1),
    GroupStudent(student_id=3, group_id=2),
    GroupStudent(student_id=4, group_id=3),
    GroupStudent(student_id=5, group_id=2)
]

result_1 = [
```

```
    (student.surname, next(group.group_name for group in groups if group.group_id == student.group_id))
    for student in students if student.surname.startswith("А")
]

group_min_scholarships = {
    group.group_name: min(student.scholarship for student in students if student.group_id == group.group_id)
    for group in groups
}
result_2 = sorted(group_min_scholarships.items(), key=lambda x: x[1])

result_3 = sorted([
    (next(student.surname for student in students if student.student_id == gs.student_id),
     next(group.group_name for group in groups if group.group_id == gs.group_id))
    for gs in group_students
], key=lambda x: x[0])

print("Task#1")
for a in result_1:
    print(a)
print("Task#2")
for a in result_2:
    print(a)
print("Task#3")
for a in result_3:
    print(a)
```

```
Task#1
('Андреев', 'Группа А')
('Алексеева', 'Группа Б')
('Аркадьев', 'Группа В')
Task#2
('Группа Б', 1100)
('Группа А', 1200)
('Группа В', 1600)
Task#3
('Алексеева', 'Группа Б')
('Андреев', 'Группа А')
('Аркадьев', 'Группа В')
('Борисов', 'Группа А')
('Васильева', 'Группа Б')
acti0n@MacBook-Alexey proga %
```