ИУ5-32Б
Афонин Иван

Рубежный контроль № 1.

Полученное задание:

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

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).

Вариант Б.

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по сотрудникам, сортировка по отделам произвольная.

2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов с количеством сотрудников в каждом отделе, отсортированный по количеству сотрудников.

3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех сотрудников, у которых фамилия заканчивается на «ов», и названия их отделов.

2	Школьник	Класс

Текст программы

```
from operator import itemgetter

class Student:
    def __init__(self, id, surname, group_id):
        self.id = id
        self.surname = surname
        self.group_id = group_id

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

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

students = [Student(1, 'Astakhov', 1), Student(2, 'Afonin', 2), Student(3,
'Borisova', 3), Student(4, 'Voloschuk', 1), Student(5, 'Zhukov', 2)]

groups = [Group(1, "IU5-31B"), Group(2, "IU5-32B"), Group(3, "IU5-33B")]

student_to_group = [StudentGroup(1, 1), StudentGroup(2, 2), StudentGroup(3, 3),
StudentGroup(4, 1), StudentGroup(5, 2)]

def first_task(students_list):
    result = sorted(students_list, key=itemgetter(0))
    return result

def second_task(students_list):
    temp = {}
    for student_surname, student_id, group_name in students_list:
        if group_name in temp:
            temp[group_name] += 1
        else:
            temp[group_name] = 1
    result = [(group_name, count) for group_name, count in temp.items()]
    result.sort(key=itemgetter(1), reverse=True)
    return result

def third_task(students_list, end):
    return [(surname, group_name) for surname, group_name in students_list if
surname.endswith(end)]

def main():
    one_to_many = [(student.surname, student.id, group.group_name)
                   for student in students
                   for group in groups
                   if student.group_id == group.id]
    many_to_many = [(student.surname, group.group_name)
                         for student in students
                         for group in groups
                         for student_group in student_to_group
                         if student.id == student_group.student_id and group.id
== student_group.group_id]

    print(first_task(one_to_many))
    print("\n")
    print(second_task(one_to_many))
    print("\n")
    print(third_task(many_to_many, "ov"))

if __name__ == "__main__":
```

```
main()
```

Результат выполнения программы

```
Задание №1
[('Afonin', 2, 'IU5-32B'), ('Astakhov', 1, 'IU5-31B'), ('Borisova', 3, 'IU5-33B'), ('Voloschuk', 4, 'IU5-31B'), ('Zhukov', 5, 'IU5-32B')]

Задание №2
[('IU5-31B', 2), ('IU5-32B', 2), ('IU5-33B', 1)]

Задание №3
[('Astakhov', 'IU5-31B'), ('Zhukov', 'IU5-32B')]
```