# Рубежный контроль №1 по курсу ПиК ЯП.

## Хорошаева Александра ИБМ3-34Б

## 26 вариант.

Условие задания.

1. Необходимо создать два класса данных «Студенческая группа-Учебный курс» на языке Python, 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 классами «Студенческая группа-Учебный курс». При разработке запросов необходимо по возможности использовать функциональные возможности языка Python (list/dict comprehensions, функции высших порядков). Запросы: a) «Учебный курс» и «Студенческая группа» связаны соотношением один-ко-многим. Выведите список всех связанных студенческих групп и учебных курсов, отсортированный по студенческим группам, сортировка по учебным курсам произвольная. b) «Учебный курс» и «Студенческая группа» связаны соотношением один-ко-многим. Выведите список учебных курсов с количеством студенческих групп на каждом учебном курсе, отсортированный по количеству студенческих групп. c) «Учебный курс» и «Студенческая группа» связаны соотношением многие-ко-многим. Выведите список всех студентов, у которых фамилия заканчивается на «ов», и названия их учебных курсов. Для реализации запроса №2 введите в класс, находящийся на стороне связи «много», произвольный количественный признак.

Реализация на языке Python.

```
class StudentGroup:
    def __init__(self, id, name):
        self.id = id
        self.name = name
class Course:
    def __init__(self, id, name):
        self.id = id
        self.name = name
student_groups = [StudentGroup(1, 'IBM3-14B'), StudentGroup(2, 'IBM-15B'), StudentGroup(3, 'IBM-16B')]
courses = [Course(1, 'Math'), Course(2, 'Physics'), Course(3, 'IT')]
# Отношение один-ко-многим
course_group_mapping = {
    1: [1, 2],
    2: [2, 3],
    3: [1, 3]
}
# Запрос 1)
```

```
result_a = [(student_group.name, course.name) for course in courses for group_id in course_group_mapping.get(course.id, []) for student_group in student_groups if student_group.id == group_id]

print('\nЗадание Б1')

print(result_a)

# Запрос 2)

group_count = {course.name: len(course_group_mapping.get(course.id, [])) for course in courses}

result_b = sorted(group_count.items(), key=lambda x: x[1], reverse=True)

print('\nЗадание Б2')

print(result_b)

class Student:

    def __init__(self, id, name, last_name):

        self.id = id

        self.name = name

        self.last_name = last_name

students = [Student(1, 'Sasha', 'Khoroshaeva'), Student(2, 'Yulia', 'Sryvalina'), Student(3, 'Artem', 'Ivanov')]

# Отношение многие-ко-многим

student_course_mapping = {

    1: [1, 2],

    2: [2, 3],

    3: [1, 3]

}

# Добавим произвольный количественный признак в класс Student

Student.grade = {1: 4.5, 2: 3.9, 3: 4.1}

# Запрос 3)

result_c = [(student.name, course.name) for student in students for course_id in student_course_mapping.get(student.id, []) for course in courses if course.id == course_id and student.last_name.endswith('ov')]

print('\nЗадание Б3')

print(result_c)
```

Результат выполнения программы.

```
Задание Б1
[('IBM3-14B', 'Math'), ('IBM-15B', 'Math'), ('IBM-15B', 'Physics'), ('IBM-16B', 'Physics'), ('IBM3-14B', 'IT'), ('IBM-16B', 'IT')]

Задание Б2
[('Math', 2), ('Physics', 2), ('IT', 2)]

Задание Б3
[('Artem', 'Math'), ('Artem', 'IT')]
>>>
```