Московский государственный технический
университет им. Н. Э. Баумана

Факультет «Радиотехнический»
Кафедра ИУ5 «Информатика и вычислительная техника»

Курс «Разработка интернет-приложений»

Отчет по рубежному контролю №1

Выполнил:
студент группы РТ5-51
Алиев Т. М.
Подпись и дата:

Проверил:
преподаватель
Гапанюк Ю. Е.
Подпись и дата:

Москва, 2021 г.

Цель работы

Работа с классами в Python, организация и реализация запросов.

Задание

Рубежный контроль представляет собой разработку программы на языке Python, которая выполняет следующие действия:

1) Необходимо создать два класса данных, которые связаны отношениями один-ко-многим и многие-ко-многим.
   Классы:
   1. Класс «Компьютер», содержащий поля:
      - ID компьютера;
      - Марка компьютера;
      - Цена (количественный признак);
      - ID записи о кабинете. (для реализации связи один-ко-многим)
   2. Класс «Кабинет», содержащий поля:
      - ID кабинета;
      - Наименование кабинета.
   3. (Для реализации связи многие-ко-многим) Класс «Сотрудники отдела», содержащий поля:
      - ID записи о компьютере;
      - ID записи о кабинете.
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 следующие запросы:
   a. «Кабинеты» и «Компьютеры» связаны соотношением один-ко-многим. Выведите список всех кабинетов, у которых в названии присутствует слово «кабинет», и список находящихся в них компьютеров.
   b. «Кабинеты» и «Компьютеры» связаны соотношением один-ко-многим. Выведите список кабинетов со средней ценой компьютеров в каждом кабинете, отсортированный по средней цене.
   c. «Кабинеты» и «Компьютеры» связаны соотношением многие-комногим. Выведите список всех компьютеров, у которых марка начинается с буквы «А», и названия их кабинетов.

При разработке запросов необходимо по возможности использовать функциональные возможности языка Python (list/dict comprehensions, функции высших порядков).

Текст программы:

```
# используется для сортировки
from operator import itemgetter


class Student:
    """Студент"""

    def __init__(self, id, last_name, gpa, group_id):
        self.id = id
        self.last_name = last_name
        self.gpa = gpa  # gpa - средний балл
        self.group_id = group_id


class Group:
    """Группа"""

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


class StudentGroup:
    """
    'Студенты группы' для реализации
    связи многие-ко-многим
    """

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


# Группы
groups = [
    Group(1, 'РТ5-51Б'),
    Group(2, 'Группа ИУ5-51Б'),
    Group(3, 'Группа ИУ5-52Б'),
```

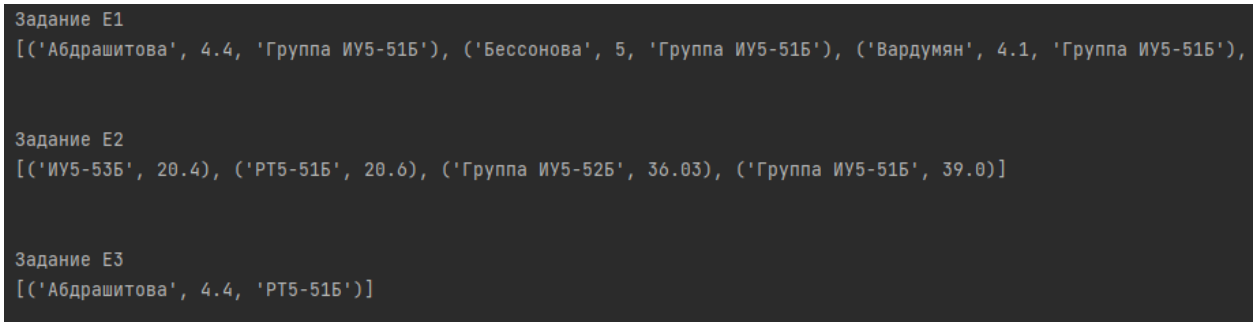
```
    Group(4, 'ИУ5-53Б'),
]

# Сотрудники
students = [
    Student(1, 'Абдрашитова', 4.4, 2),
    Student(2, 'Бессонова', 5, 2),
    Student(3, 'Вардумян', 4.1, 2),
    Student(4, 'Гальцев', 3.2, 2),
    Student(5, 'Дьячков', 3.6, 2),
    Student(6, 'Компаниец', 2.7, 2),
    Student(7, 'Мартынова', 3.7, 2),
    Student(8, 'Ноздрова', 5, 2),
    Student(9, 'Плотников', 4, 2),
    Student(10, 'Удодова', 3.3, 2),
    Student(11, 'Алексеев', 2.8, 3),
    Student(12, 'Бабин', 4.3, 3),
    Student(13, 'Ваганов', 2.3, 3),
    Student(14, 'Заузолков', 4.2, 3),
    Student(15, 'Корчевский', 2.43, 3),
    Student(16, 'Левин', 2.5, 3),
    Student(17, 'Никитина', 4.1, 3),
    Student(18, 'Олейников', 4.3, 3),
    Student(19, 'Тохсыров', 5, 3),
    Student(20, 'Федосеева', 4.1, 3),
    Student(21, 'Баданин', 4.5, 4),
    Student(22, 'Губанов', 4.8, 4),
    Student(23, 'Зонова', 2.3, 4),
    Student(24, 'Лялин', 4.6, 4),
    Student(25, 'Светашева', 4.2, 4),
    Student(26, 'Алиев', 5, 1),
    Student(27, 'Борисочкин', 4.1, 1),
    Student(28, 'Незаметдинов', 3, 1),
    Student(29, 'Робертс', 3.5, 1),
    Student(30, 'Фильчиков', 5, 1)
]

students_groups = [
    StudentGroup(1, 1),
    StudentGroup(2, 2),
```

```
    StudentGroup(3, 3),
    StudentGroup(3, 4),
    StudentGroup(3, 3),

    StudentGroup(11, 1),
    StudentGroup(22, 2),
    StudentGroup(30, 3),
    StudentGroup(30, 4),
    StudentGroup(30, 2),
]


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

    # Соединение данных один-ко-многим
    one_to_many = [(e.last_name, e.gpa, d.name)
                   for d in groups
                   for e in students
                   if e.group_id == d.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(d.name, ed.group_id, ed.student_id)
                         for d in groups
                         for ed in students_groups
                         if d.id == ed.group_id]

    many_to_many = [(e.last_name, e.gpa, group_name)
                    for group_name, group_id, student_id in many_to_many_temp
                    for e in students if e.id == student_id]

    # task 1
    print('Задание E1')
    res_1 = list(filter(lambda i: i[2].lower().find('группа') != -1, one_to_many))
    print(res_1)
    print("\n")

    # task 2
    print('Задание E2')
    res_12_unsorted = []
```

```
    for d in groups:
        count = 0
        d_students = list(filter(lambda i: i[2] == d.name, one_to_many))
        if len(d_students) > 0:
            d_gpa = [gpa for _, gpa, _ in d_students]
            count += 1
            d_gpa_sum = sum(d_gpa)
            aver_sum = round(d_gpa_sum / count, 2)


            res_12_unsorted.append((d.name, aver_sum))
    res_12 = sorted(res_12_unsorted, key=itemgetter(1))
    print(res_12)
    print("\n")

    # task 3
    a = ['a', 'a']  # одна буква латинская, другая русская
    print('Задание E3')
    res_3 = list(filter(lambda i: a.count(i[0][0].lower()) != 0, many_to_many))
    print(res_3)


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

Экранные формы с примерами выполнения программы:

Результат выполнения (вывод задания E1 не поместился полностью на скриншоте):



Вывод

Продемонстрированы возможности работы с классами в Python, организации и реализации запросов.