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
«Разработка программ на языке Python.»

Выполнил:
студент группы РТ5-51
Бушуев В.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:

1. «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.
2. «Кабинеты» и «Компьютеры» связаны соотношением один-ко-многим. Выведите список кабинетов со средней ценой компьютеров в каждом кабинете, отсортированный по средней цене.
3. «Кабинеты» и «Компьютеры» связаны соотношением многие-ко-многим. 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:

```
import functools as ft
import fnmatch


class Computer:
    def __init__(self, ID, MARK, PRICE, RM_ID):
        self.id = ID
        self.mark = MARK
        self.pr = PRICE
        self.rm_id = RM_ID


class Room:
    def __init__(self, ID, NAME):
        self.id = ID
        self.name = NAME


class CompRoom:
    def __init__(self, CMP_ID, RM_ID):
        self.cmp_id = CMP_ID
        self.rm_id = RM_ID


def room_print(one_many):
    if ('кабинет' in one_many[2].lower()):
        print("%-25s" % one_many[2])
```

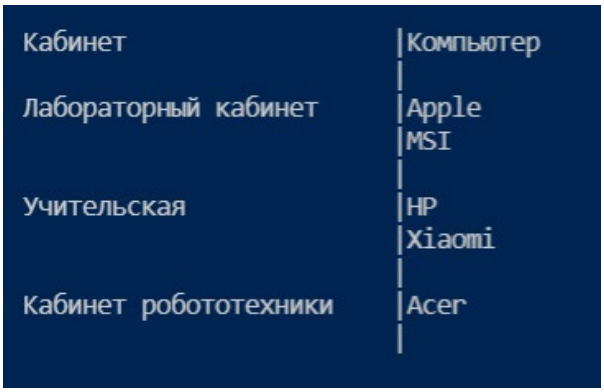
```
    return one_many


def main():
    computers = [Computer(1, "Apple", 70000, 1),
                 Computer(2, "Acer", 40000, 3),
                 Computer(3, "HP", 50000, 2),
                 Computer(4, "MSI", 60000, 1),
                 Computer(5, "Xiaomi", 55000, 2)]

    rooms = [Room(1, "Лабораторный кабинет"),
             Room(2, "Учительская"),
             Room(3, "Кабинет робототехники")]

    comp_room = [CompRoom(1, 1),
                 CompRoom(1, 3),
                 CompRoom(2, 2),
                 CompRoom(3, 2),
                 CompRoom(3, 3),
                 CompRoom(4, 1),
                 CompRoom(5, 1),
                 CompRoom(5, 3)
                 ]

    one_many = [(comp.mark, comp.pr, room.name)
                for comp in computers
                for room in rooms
                if comp.rm_id == room.id]
```

```
comps_table = {comp.mark: [table.rm_id for table in comp_room
                           if (comp.id == table.cmp_id)]
               for comp in computers
               }

a3 = {mark: [rm.name for rm in rooms if (rm.id in comps_table[mark])]
      for mark in comps_table

      }

class A12:
    def __init__(self, room_name, comps, sum_pr) -> None:
        self.rm = room_name
        self.computers = comps
        self.avg_pr = sum_pr / len(self.computers)

a12 = [A12(rm.name, [comp for comp, pr, room in one_many if room == rm.name], ft.reduce(
    lambda acc, elem: acc + elem, [pr for comp, pr, room in one_many if room == rm.name], 0))for rm in rooms]

print("\n\n%-25s|%-15s\n%25s|" % ("Кабинет", "Компьютер", ""), end="")

for room in a12:
    print("\n%-25s|" % (room.rm), end='')
    for comp in room.computers:
        print("%-15s\n%25s|" % (comp, ""), end="")
```

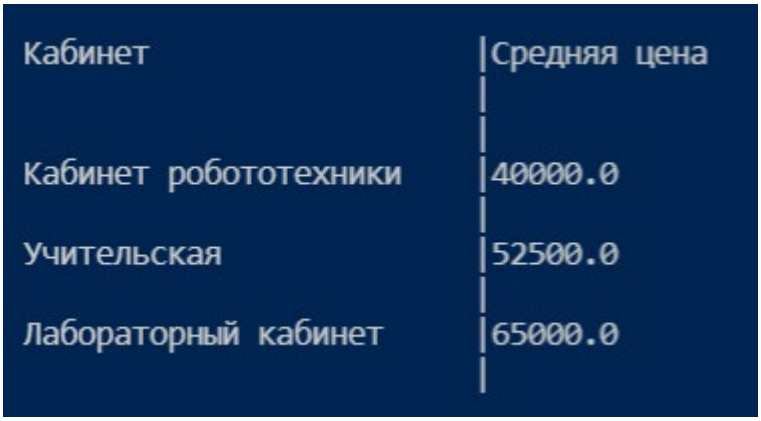
```
    print("\n\n\n\n%-25s|%-15s\n%25s|\n%25s|" %
          ("Кабинет", "Средняя цена", "", ""))
    for room in sorted(a12, key=lambda x: x.avg_pr):
        print("%-25s|{0}\n%-25s|".format(room.avg_pr)    %    (room.rm,
""))

    print("\n\n\nКомпьютер / Его кабинеты:\n")
    for key in a3:
        if fnmatch.fnmatch(key, "A*"):
            print(key + ": " + ", ".join(a3[key]))
    print("\n\n")


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

Экранные формы с примерами выполнения программы:

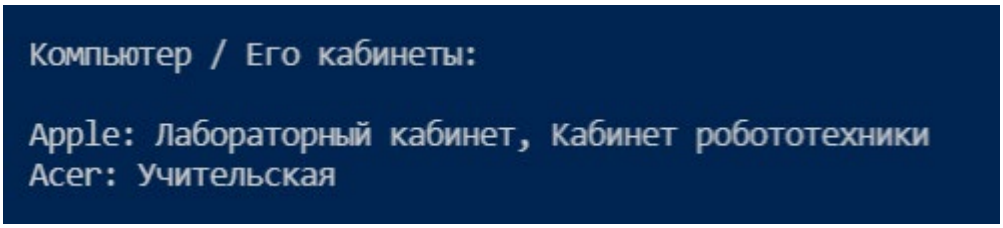
1.



2.



3.



Вывод

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