Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и управление»

Кафедра «Системы обработки информации и управления»

Курс «Разработка интернет-приложений»

Отчет по рубежному контролю №1

Вариант Г7

Выполнила:
студентка группы ИУ5-54
Каримов А.Т.

Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Подпись и дата:

Москва, 2021 г.

Описание задания

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

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».

Текст программы

```
# используется для сортировки
from operator import itemgetter


class Proc:
    """Микропроцессор"""
    def __init__(self, id, ProcName, cost, comp_id):
        self.id = id
        self.ProcName = ProcName
        self.cost = cost
        self.comp_id = comp_id
```

```python
class Comp:
    """Компьютер"""
    def __init__(self, id, name):
        self.id = id
        self.name = name


class ProcComp:
    """
    'Программы компа' для реализации
    связи многие-ко-многим
    """
    def __init__(self, proc_id, comp_id):
        self.proc_id = proc_id
        self.comp_id = comp_id


# Компьютеры
comps = [
    Comp(1, 'PC Мама'),
    Comp(2, 'PC Папа'),
    Comp(3, 'PC Брат'),
    Comp(4, 'PC Личный'),
]

# Программы
microproc = [
    Proc(1, 'Inteli1', 2000, 1),
    Proc(2, 'Inteli2', 3300, 2),
    Proc(3, 'Inteli3', 5000, 3),
    Proc(4, 'Inteli4', 3000, 3),
    Proc(5, 'Inteli5', 1000, 4),
]

microproc_comps = [
    ProcComp(1, 1),
    ProcComp(2, 2),
    ProcComp(3, 3),
    ProcComp(4, 4),
    ProcComp(5, 4),
]


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

    # Соединение данных один-ко-многим
    one_to_many = [(s.ProcName, s.cost, k.name)
                   for k in comps
                   for s in microproc
                   if s.comp_id == k.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(k.name, sk.comp_id, sk.proc_id)
                         for k in comps
                         for sk in microproc_comps
                         if k.id == sk.comp_id]

    many_to_many = [(s.ProcName, s.cost, comp_name)
                    for comp_name, comp_id, proc_id in many_to_many_temp
                    for s in microproc if s.id == proc_id]

    print('Задание Г1')
    res_11 = [(s.ProcName, k.name)
              for k in comps
              for s in microproc
```

```
                if (s.comp_id == k.id) and (k.name == "PC Папа" or k.name == "PC
Мама")]
    print(res_11)


    print('\nЗадание Г2')
    res_12_unsorted = []

    for k in comps:

        k_microproc = list(filter(lambda i: i[2] == k.name, one_to_many))

        if len(k_microproc) > 0:

            k_costs = [cost for _, cost, _ in k_microproc]

            k_costs_max = max(k_costs)
            res_12_unsorted.append((k.name, k_costs_max))


    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    print(res_12)

    print('\nЗадание Г3')
    res_13 = sorted(many_to_many, key=itemgetter(2))
    print(res_13)



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

Анализ результатов

```
Задание Г1
[('Inteli1', 'PC Мама'), ('Inteli2', 'PC Папа')]

Задание Г2
[('PC Брат', 5000), ('PC Папа', 3300), ('PC Мама', 2000), ('PC Личный', 1000)]

Задание Г3
[('Inteli3', 5000, 'PC Брат'), ('Inteli4', 3000, 'PC Личный'), ('Inteli5', 1000, 'PC Личный'), ('Inteli1', 2000, 'PC Мама'), ('Inteli2', 3300, 'PC Папа')]

Process finished with exit code 0
```