**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Разработка интернет приложений»

Рубежный контроль №1

Вариант №1Б

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-52Б |  | преподаватель каф. ИУ5 |
| Алексеев А. С. |  | Гапанюк Ю.Е. |
|  |  |  |

Москва, 2021 г.

Условия

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).

Код программы

# используется для сортировки

from operator import itemgetter

class Stud:

"""Студент"""

def \_\_init\_\_(self, id, fio, averper, gro\_id):

self.id = id

self.fio = fio

self.averper = averper

self.gro\_id = gro\_id

class Gro:

"""Группа"""

def \_\_init\_\_(self, id, name):

self.id = id

self.name = name

class StudGro:

"""

'Студент группы' для реализации

связи многие-ко-многим

"""

def \_\_init\_\_(self, gro\_id, stud\_id):

self.gro\_id = gro\_id

self.stud\_id = stud\_id

# Группы

gros = [

Gro(1, 'ИУ5 - 51Б'),

Gro(2, 'ИУ6 - 52Б'),

Gro(3, 'ИУ7 - 53Б'),

Gro(11, 'ИУ5'),

Gro(22, 'ИУ6'),

Gro(33, 'ИУ7'),

]

# Студенты

studs = [

Stud(1, 'Артамонов', 3, 1),

Stud(2, 'Петров', 4, 2),

Stud(3, 'Иваненко', 5, 3),

Stud(4, 'Иванов', 4, 3),

Stud(5, 'Иванин', 3, 3),

]

studs\_gros = [

StudGro(1, 1),

StudGro(2, 2),

StudGro(3, 3),

StudGro(3, 4),

StudGro(3, 5),

StudGro(11, 1),

StudGro(22, 2),

StudGro(33, 3),

StudGro(33, 4),

StudGro(33, 5),

]

def main():

"""Основная функция"""

# Соединение данных один-ко-многим

one\_to\_many = [(s.fio, s.averper, g.name)

for g in gros

for s in studs

if s.gro\_id == g.id]

# Соединение данных многие-ко-многим

many\_to\_many\_temp = [(g.name, sg.gro\_id, sg.stud\_id)

for g in gros

for sg in studs\_gros

if g.id == sg.gro\_id]

many\_to\_many = [(s.fio, s.averper, gro\_id)

for gro\_name, gro\_id, stud\_id in many\_to\_many\_temp

for s in studs if s.id == stud\_id]

print('Задание А1')

res\_11 = sorted(one\_to\_many, key=itemgetter(2))

print(res\_11)

print('\nЗадание А2')

res\_12\_unsorted = []

# Перебираем все группы

for g in gros:

# Список студентов группы

g\_studs = list(filter(lambda i: i[2] == g.name, one\_to\_many))

# Если группа не пустая

if len(g\_studs) > 0:

# Все средние оценки студентов группы

g\_averpers = [averper for \_, averper, \_ in g\_studs]

# Средняя оценка группы

g\_averper\_gro = sum(g\_averpers)/len(g\_studs)

res\_12\_unsorted.append((g.name, g\_averper\_gro))

# Сортировка по средней оценке группы

res\_12 = sorted(res\_12\_unsorted, key=itemgetter(1), reverse=True)

print(res\_12)

print('\nЗадание А3')

res\_13 = {}

# Перебираем все группы

for g in gros:

if 'ИУ5' in g.name:

# Список студентов группы

g\_studs = list(filter(lambda i: i[2] == g.name, many\_to\_many))

# Только ФИО студентов

g\_studs\_names = [x for x, \_, \_ in g\_studs]

# Добавляем результат в словарь

# ключ - группа, значение - список фамилий

res\_13[g.name] = g\_studs\_names

print(res\_13)

wait = input("PRESS ENTER TO CONTINUE.")

if \_\_name\_\_ == '\_\_main\_\_':

main()

Результаты программы

