**Московский государственный технический**

**университет им. Н.Э. Баумана.**



Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по рубежному контролю № 2

|  |  |  |
| --- | --- | --- |
| Выполнила: |  | Проверил: |
| студентка группы ИУ5-32Б  Андреева Анастасия |  | преподаватель каф. ИУ5  Гапанюк Ю. Е. |
|  |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2021 г.

**Описание задания**

(Вариант запросов - Б, вариант предметной области – 1)

1. «Группа» и «Студент» связаны соотношением один-ко-многим. Выведите список всех связанных студентов и групп, отсортированный по студентам, сортировка по группам произвольная.
2. «Группа» и «Студент» связаны соотношением один-ко-многим. Выведите список групп с количеством студентов в каждом отделе, отсортированный по количеству студентов.
3. «Группа» и «Студент» связаны соотношением многие-ко-многим. Выведите список всех студентов, у которых фамилия заканчивается на «ов», и названия их групп.

**Условия рубежного контроля №2 по курсу БКИТ**

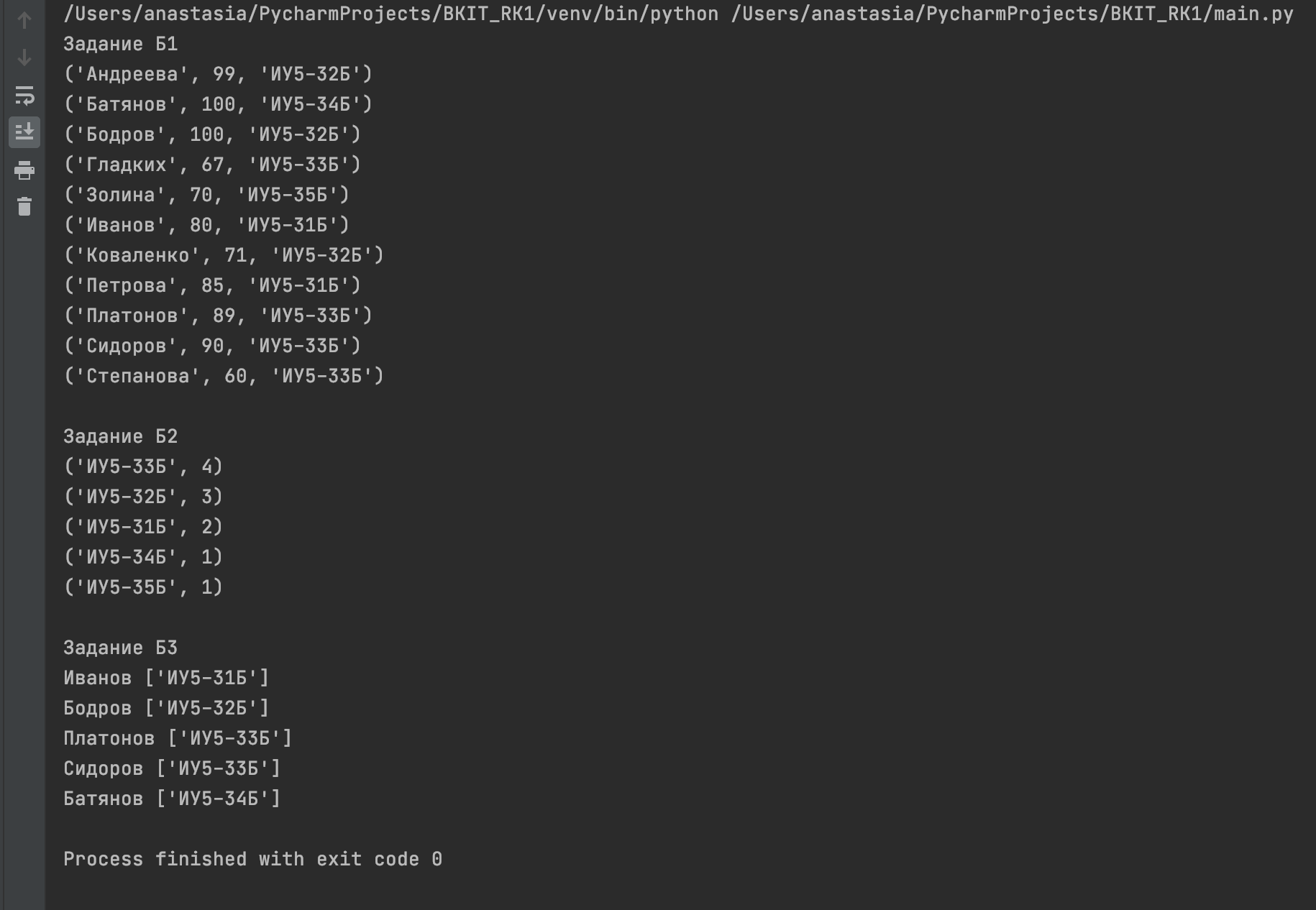
Рубежный контроль представляет собой разработку тестов на языке Python.

1) Проведите рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.

2) Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

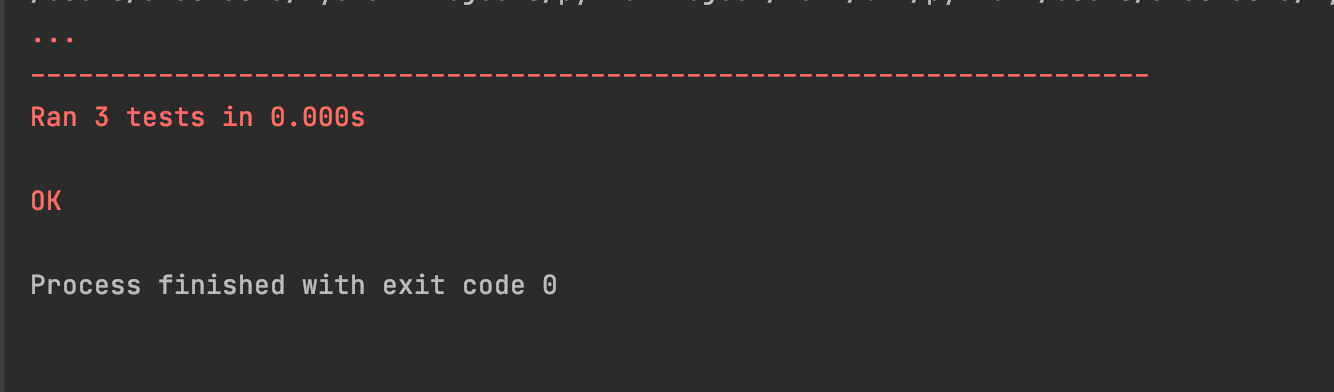
1. **Текст программы main.py**

from operator import itemgetter  
  
class Stud:  
 *"""Студент"""* def \_\_init\_\_(self, id, fio, scr, grp\_id ):  
 self.id = id  
 self.fio = fio  
 self.scr = scr  
 self.grp\_id = grp\_id  
  
class Grp:  
 *"""Группа"""* def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
  
class StudGrp:  
 *""" 'Студенты группы' для реализации связи многие-ко-многим"""* def \_\_init\_\_(self, stud\_id, grp\_id):  
 self.grp\_id = grp\_id  
 self.stud\_id = stud\_id  
  
# Группы  
groups = [  
 Grp(1, 'ИУ5-31Б'),  
 Grp(2, 'ИУ5-32Б'),  
 Grp(3, 'ИУ5-33Б'),  
 Grp(4, 'ИУ5-34Б'),  
 Grp(5, 'ИУ5-35Б')  
]  
  
# Студенты  
studs = [  
 Stud(1, 'Иванов', 80, 1),  
 Stud(2, 'Петрова', 85, 1),  
 Stud(3, 'Андреева', 99, 2),  
 Stud(4, 'Коваленко', 71, 2),  
 Stud(5, 'Бодров', 100, 2),  
 Stud(6, 'Степанова', 60, 3),  
 Stud(7, 'Платонов', 89, 3),  
 Stud(8, 'Сидоров', 90, 3),  
 Stud(9, 'Гладких', 67, 3),  
 Stud(10, 'Батянов', 100, 4),  
 Stud(11, 'Золина', 70, 5)  
]  
  
studs\_groups = [  
 StudGrp(1,1),  
 StudGrp(2,1),  
 StudGrp(3,2),  
 StudGrp(4,2),  
 StudGrp(5,2),  
 StudGrp(6,3),  
 StudGrp(7,3),  
 StudGrp(8,3),  
 StudGrp(9,3),  
 StudGrp(10,4),  
 StudGrp(11,5)  
]  
def b1(one\_to\_many):  
  
 res1 = sorted(one\_to\_many, key=itemgetter(0))  
 return res1  
  
def b2(one\_to\_many):  
 res2\_temp = []  
 # Перебираем все группы  
 for g in groups:  
 # Список студентов группы  
 g\_studs = list(filter(lambda i: i[2] == g.name, one\_to\_many))  
 if len(g\_studs) > 0:  
 res2\_temp.append((g.name, len(g\_studs)))  
 # Сортировка по количеству студентов в группе  
 res2 = sorted(res2\_temp, key=itemgetter(1), reverse=True)  
 return res2  
  
def b3(many\_to\_many):  
 res3 = {}  
 for s in studs:  
 if s.fio.endswith('ов'):  
 # Список студентов с фамилией на "ов"  
 s\_groups = list(filter(lambda i: i[0] == s.fio, many\_to\_many))  
 # Получаем их имена  
 s\_groups\_names = [x[2] for x in s\_groups]  
 res3[s.fio] = s\_groups\_names  
 return res3  
  
def main():  
 # Соединение данных один-ко-многим  
 one\_to\_many = [(s.fio, s.scr, g.name)  
 for s in studs  
 for g in groups  
 if s.grp\_id==g.id]  
 # Соединение данных многие-ко-многим  
 many\_to\_many\_temp = [(g.name, sg.grp\_id, sg.stud\_id)  
 for g in groups  
 for sg in studs\_groups  
 if g.id==sg.grp\_id]  
  
 many\_to\_many = [(s.fio, s.scr, grp\_name)  
 for grp\_name, grp\_id, stud\_id in many\_to\_many\_temp  
 for s in studs if s.id == stud\_id]  
  
 print('Задание Б1')  
 print(b1(one\_to\_many))  
  
 print('\nЗадание Б2')  
  
 print(b2(one\_to\_many))  
  
 print('\nЗадание Б3')  
  
 print(b3(many\_to\_many))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Результат выполнения программы main.py**

1. **Текст программы test\_tdd.py**

import unittest  
from main import \*  
  
class test\_cases(unittest.TestCase):  
 def test\_b1(self):  
 one\_to\_many = [(s.fio, s.scr, g.name)  
 for s in studs  
 for g in groups  
 if s.grp\_id == g.id]  
  
  
  
 self.assertEqual(b1(one\_to\_many), [('Андреева', 99, 'ИУ5-32Б'),  
 ('Батянов', 100, 'ИУ5-34Б'),  
 ('Бодров', 100, 'ИУ5-32Б'),  
 ('Гладких', 67, 'ИУ5-33Б'),  
 ('Золина', 70, 'ИУ5-35Б'),  
 ('Иванов', 80, 'ИУ5-31Б'),  
 ('Коваленко', 71, 'ИУ5-32Б'),  
 ('Петрова', 85, 'ИУ5-31Б'),  
 ('Платонов', 89, 'ИУ5-33Б'),  
 ('Сидоров', 90, 'ИУ5-33Б'),  
 ('Степанова', 60, 'ИУ5-33Б')])  
 def test\_b2(self):  
 one\_to\_many = [(s.fio, s.scr, g.name)  
 for s in studs  
 for g in groups  
 if s.grp\_id == g.id]  
 self.assertEqual(b2(one\_to\_many), [('ИУ5-33Б', 4),  
 ('ИУ5-32Б', 3),  
 ('ИУ5-31Б', 2),  
 ('ИУ5-34Б', 1),  
 ('ИУ5-35Б', 1)])  
  
 def test\_b3(self):  
 many\_to\_many\_temp = [(g.name, sg.grp\_id, sg.stud\_id)  
 for g in groups  
 for sg in studs\_groups  
 if g.id == sg.grp\_id]  
  
 many\_to\_many = [(s.fio, s.scr, grp\_name)  
 for grp\_name, grp\_id, stud\_id in many\_to\_many\_temp  
 for s in studs if s.id == stud\_id]  
 self.assertEqual(b3(many\_to\_many),{'Иванов': ['ИУ5-31Б'],  
 'Бодров': ['ИУ5-32Б'],  
 'Платонов': ['ИУ5-33Б'],  
 'Сидоров': ['ИУ5-33Б'],  
 'Батянов': ['ИУ5-34Б']})  
  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

**Результат выполнения программы test\_tdd.py**