**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 №1  
«Основные конструкции языка Python»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-32Б: |  | преподаватель каф. ИУ5 |
| Балабанов Алексей Олегович |  | Гапонюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2021 г.

**Постановка задачи:**

**Вариант №2**

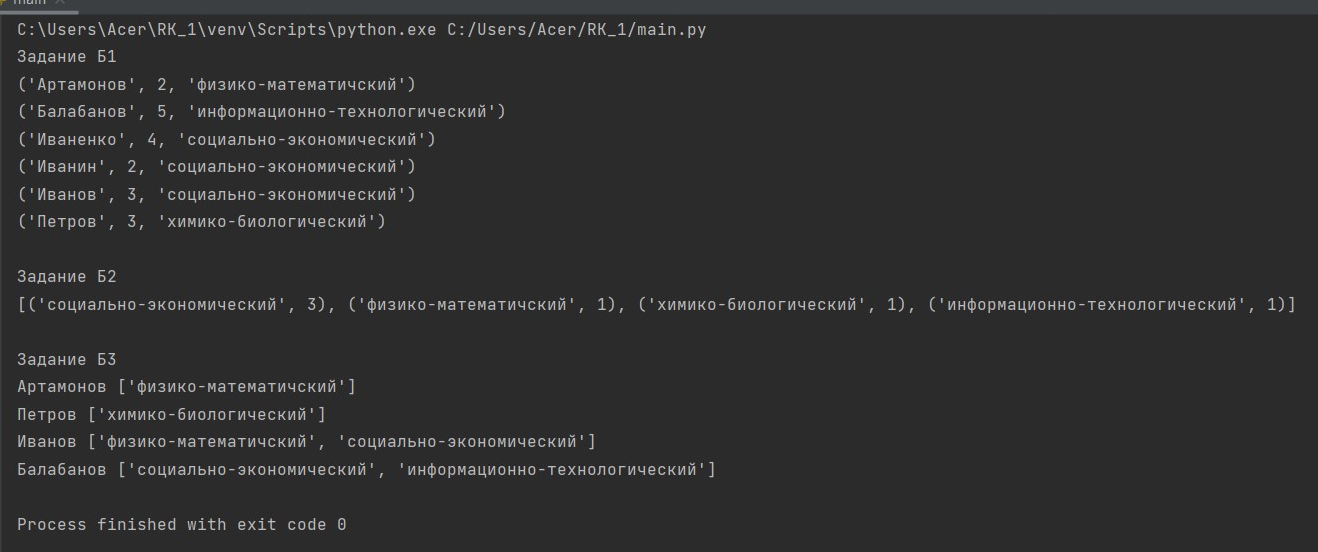
**Вариант Б.**

1. «Класс» и «Школьник» связаны соотношением один-ко-многим. Выведите список всех связанных сотрудников и отделов, отсортированный по сотрудникам, сортировка по отделам произвольная.
2. «Класс» и «Школьник» связаны соотношением один-ко-многим. Выведите список отделов с количеством сотрудников в каждом отделе, отсортированный по количеству сотрудников.
3. «Класс» и «Школьник» связаны соотношением многие-ко-многим. Выведите список всех сотрудников, у которых фамилия заканчивается на «ов», и названия их отделов.

**Текст кода:**

# используется для сортировки  
from operator import itemgetter  
  
  
class Student:  
 *"""Школьник"""* def \_\_init\_\_(self, id, fio, bal, st\_id):  
 self.id = id  
 self.fio = fio  
 self.bal = bal #Оценка по информатике  
 self.st\_id = st\_id  
  
  
class Class:  
 *"""Класс"""* def \_\_init\_\_(self, id, name):  
 self.id = id  
 self.name = name  
  
  
class StCl:  
 *"""  
 'Учащиеся класса' для реализации  
 связи многие-ко-многим  
 """* def \_\_init\_\_(self, st\_id, class\_id):  
 self.class\_id = class\_id  
 self.st\_id = st\_id  
  
  
# Классы  
clas = [  
 Class(1, 'физико-математичский'),  
 Class(2, 'химико-биологический'),  
 Class(3, 'социально-экономический'),  
 Class(4, 'информационно-технологический'),  
  
]  
  
# Школьники  
students = [  
 Student(1, 'Артамонов', 2, 1),  
 Student(2, 'Петров', 3, 2),  
 Student(3, 'Иваненко', 4, 3),  
 Student(4, 'Иванов', 3, 3),  
 Student(5, 'Иванин', 2, 3),  
 Student(6, 'Балабанов', 5, 4),  
]  
  
st\_cl = [  
 StCl(1, 1),  
 StCl(2, 2),  
 StCl(3, 3),  
 StCl(4, 3),  
 StCl(5, 3),  
 StCl(6, 4),  
  
 StCl(3, 2),  
 StCl(4, 1),  
 StCl(5, 4),  
 StCl(6, 3),  
]  
  
  
def main():  
 *"""Основная функция"""* # Соединение данных один-ко-многим  
 one\_to\_many = [(s.fio, s.bal, c.name)  
 for c in clas  
 for s in students  
 if s.st\_id == c.id]  
  
 # Соединение данных многие-ко-многим  
 many\_to\_many\_temp = [(c.name, cs.class\_id, cs.st\_id)  
 for c in clas  
 for cs in st\_cl  
 if c.id == cs.class\_id]  
  
 many\_to\_many = [(s.fio, s.bal, class\_name)  
 for class\_name, class\_id, st\_id in many\_to\_many\_temp  
 for s in students if s.id == st\_id]  
  
 print('Задание Б1')  
 res\_11 = sorted(one\_to\_many, key=itemgetter(0))  
 print(res\_11)  
  
 print('\nЗадание Б2')  
 res\_12\_unsorted = []  
 # Перебираем все классы  
 for c in clas:  
 # Список учеников класса  
 c\_student = list(filter(lambda i: i[2] == c.name, one\_to\_many))  
 # Если класс не пустой  
 if len(c\_student) > 0:  
 res\_12\_unsorted.append((c.name, len(c\_student)))  
  
 res\_12 = sorted(res\_12\_unsorted, key=itemgetter(1), reverse=True)  
 print(res\_12)  
  
 print('\nЗадание Б3')  
 res\_13 = {}  
 # Перебираем все отделы  
 for s in students:  
 if s.fio.endswith("ов"):  
 c\_students = list(filter(lambda i: i[0] == s.fio, many\_to\_many))  
 c\_students\_names = [x[2] for x in c\_students]  
 # Добавляем результат в словарь  
 # ключ - отдел, значение - список фамилий  
 res\_13[s.fio] = c\_students\_names  
  
 print(res\_13)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Тестирование:**

****

(Для более удобного отображения результата вывод Б1 и Б2 был изменён)