

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. Баумана

Факультет “Информатика и системы управления”
Кафедра “Системы обработки информации и управления”



Дисциплина “Парадигмы и конструкции языков программирования”

Отчет по рубежному контролю № 1
Вариант Д1

ИМ

Выполнил:
Студент группы ИУ5-36Б
Германенко И.М.
Преподаватель:
Гапанюк Ю.Е.

Москва 2025

ЛИСТИНГ КОДА

```
from operator import itemgetter
```

```
class Student:
```

```
    """Студент"""
```

```
    def __init__(self, id, fio, scholarship, group_id):
```

```
        self.id = id
```

```
        self.fio = fio
```

```
        self.scholarship = scholarship # Количественный признак для запроса Д2
```

```
        self.group_id = group_id
```

```
class Group:
```

```
    """Группа"""
```

```
    def __init__(self, id, name):
```

```
        self.id = id
```

```
        self.name = name
```

```
class StudentGroup:
```

```
    """'Студенты группы' для реализации связи многие-ко-многим"""
```

```
    def __init__(self, student_id, group_id):
```

```
        self.student_id = student_id
```

```
        self.group_id = group_id
```

```
groups = [
```

```
    Group(1, 'ИУ5-31Б'),
```

```
    Group(2, 'ИУ5-32Б'),
```

```
Group(3, 'AK4-01Б'), # Название начинается на 'А'  
Group(11, 'АЭ7-11М'), # Название начинается на 'А'  
]
```

```
students = [  
    Student(1, 'Петров', 2500, 1),  
    Student(2, 'Сидоров', 3000, 2),  
    Student(3, 'Иванов', 3500, 2),  
    Student(4, 'Смирнов', 2500, 3),  
    Student(5, 'Кузнецов', 4000, 3),  
]
```

```
# 'Студенты группы' для связи многие-ко-многим
```

```
# Допустим, некоторые студенты посещают дополнительные курсы в других  
группах
```

```
students_groups = [  
    StudentGroup(1, 1),  
    StudentGroup(2, 2),  
    StudentGroup(3, 2),  
    StudentGroup(4, 3),  
    StudentGroup(5, 3),  
    # Студенты на доп. курсах в группах, начинающихся на 'А'  
    StudentGroup(1, 11), # Петров в АЭ7-11М  
    StudentGroup(2, 11), # Сидоров в АЭ7-11М  
    StudentGroup(4, 11), # Смирнов в АЭ7-11М  
]
```

```
def main():
```

```
    """Основная функция"""
```

```
    # Соединение данных один-ко-многим
```

```
    one_to_many = [(s.fio, s.scholarship, g.name)
```

```
                    for g in groups
```

```
                    for s in students
```

```
                    if s.group_id == g.id]
```

```
    # Соединение данных многие-ко-многим
```

```
    many_to_many_temp = [(g.name, sg.student_id)
```

```
                        for g in groups
```

```
                        for sg in students_groups
```

```
                        if g.id == sg.group_id]
```

```
    many_to_many = [(s.fio, s.scholarship, group_name)
```

```
                    for group_name, student_id in many_to_many_temp
```

```
                    for s in students if s.id == student_id]
```

```
    print('Задание Д1')
```

```
    # Вывод списка всех студентов, у которых фамилия заканчивается на «ов»,
```

```
    # и названия их групп. (связь один-ко-многим)
```

```
res_d1 = [(fio, group_name)
           for fio, scholarship, group_name in one_to_many
           if fio.endswith('ов')]
print(res_d1)
```

```
print('\nЗадание Д2')
```

```
# Вывод списка групп со средней стипендией студентов в каждой группе,
# отсортированный по средней стипендии. (связь один-ко-многим)
res_d2_unsorted = []
```

```
for g in groups:
```

```
    g_students = list(filter(lambda i: i[2] == g.name, one_to_many))
```

```
    # Если группа не пустая
```

```
    if len(g_students) > 0:
```

```
        g_scholarships = [scholarship for _, scholarship, _ in g_students]
```

```
        g_scholarship_avg = sum(g_scholarships) / len(g_scholarships)
```

```
        res_d2_unsorted.append((g.name, g_scholarship_avg))
```

```
# Сортировка по средней стипендии
```

```
res_d2 = sorted(res_d2_unsorted, key=itemgetter(1))
```

```
print(res_d2)
```

```

print('\nЗадание Д3')

# Вывод списка всех групп, у которых название начинается с буквы «А»,
# и списка обучающихся в них студентов. (связь многие-ко-многим)
res_d3 = {}

for g in groups:
    if g.name.startswith('A'):

        g_students = list(filter(lambda i: i[2] == g.name, many_to_many))

        g_students_fio = [fio for fio, _, _ in g_students]
        res_d3[g.name] = g_students_fio
print(res_d3)

if __name__ == '__main__':
    main()

```

Результат выполнения программы

Задание Д1

```

[('Петров', 'ИУ5-31Б'), ('Сидоров', 'ИУ5-32Б'), ('Иванов', 'ИУ5-32Б'), ('Смирнов',
'АК4-01Б'), ('Кузнецов', 'АК4-01Б')]

```

Задание Д2

```

[('ИУ5-31Б', 2500.0), ('ИУ5-32Б', 3250.0), ('АК4-01Б', 3250.0)]

```

Задание Д3

```

{'АК4-01Б': ['Смирнов', 'Кузнецов'], 'АЭ7-11М': ['Петров', 'Сидоров',
'Смирнов']}

```