



**Министерство науки и высшего образования Российской
Федерации Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

**Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»**

Лабораторная работа №2
по дисциплине «Электротехника» на тему:
«Источники тока и напряжения»

Выполнил:
студент группы ИУ5-35Б
Акулова А.А.

Преподаватель:
Ю.Е. Гапанюк

2021 г.

Текст программы

```
# используется для сортировки
from operator import itemgetter

class Student:
    """Студент"""
    def __init__(self, id, fio, debt_quantity, group_id):
        self.id = id
        self.fio = fio
        self.debt_quantity = debt_quantity
        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, group_id, student_id):
        self.student_id = student_id
        self.group_id = group_id

# Студенты
students = [
    Student(1, 'Акулова', 0, 1),
    Student(2, 'Александров', 1, 1),
    Student(3, 'Беленьков', 4, 2),
    Student(4, 'Ким', 7, 3),
    Student(5, 'Норков', 5, 3),
]

# Группы
groups = [
    Group(1, 'ИУ5-35'),
    Group(2, 'Аэрокосмическая группа #1'),
    Group(3, 'Группа по созданию черного ящика "ГАС Контур")',

    Group(11, 'ИУ5-31'),
    Group(22, 'аэрокосмическая группа #2'),
    Group(33, 'Амбициозная группа по разработке АРМ ЛПР'),
]

students_groups = [
    StudentGroup(1, 1),
    StudentGroup(1, 2),
    StudentGroup(2, 3),
    StudentGroup(3, 4),
    StudentGroup(3, 5),

    StudentGroup(11, 1),
    StudentGroup(11, 2),
    StudentGroup(22, 3),
    StudentGroup(33, 4),
    StudentGroup(33, 5),
]
```

```

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

    # Соединение данных один-ко-многим
    one_to_many = [(s.fio, s.debt_quantity, 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.group_id, sg.student_id)
                           for g in groups
                           for sg in students_groups
                           if g.id == sg.group_id]

    many_to_many = [(s.fio, s.debt_quantity, group_name)
                     for group_name, group_id, student_id in many_to_many_temp
                     for s in students if s.id == student_id]

    print('Задание A1')
    res_11 = []
    for s in students:
        # Перебираем всех студентов
        if s.fio.endswith('ов'):
            # Список студентов группы
            g_students = list(filter(lambda i: i[0]==s.fio, one_to_many))
            res_11.append(g_students)
    print(res_11)

    print('\nЗадание A2')
    res_12_unsorted = []
    # Перебираем все отделы
    for g in groups:
        # Список студентов группы
        g_students = list(filter(lambda i: i[2]==g.name, one_to_many))
        # Если группа не пустая
        if len(g_students) > 0:
            # Долги студентов группы
            g_debts = [debt for _, debt, _ in g_students]
            # Средний долг студентов группы
            g_debts_avg = sum(g_debts)/len(g_debts)
            res_12_unsorted.append((g.name, g_debts_avg))

    # Сортировка по среднему долгу
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    print(res_12)

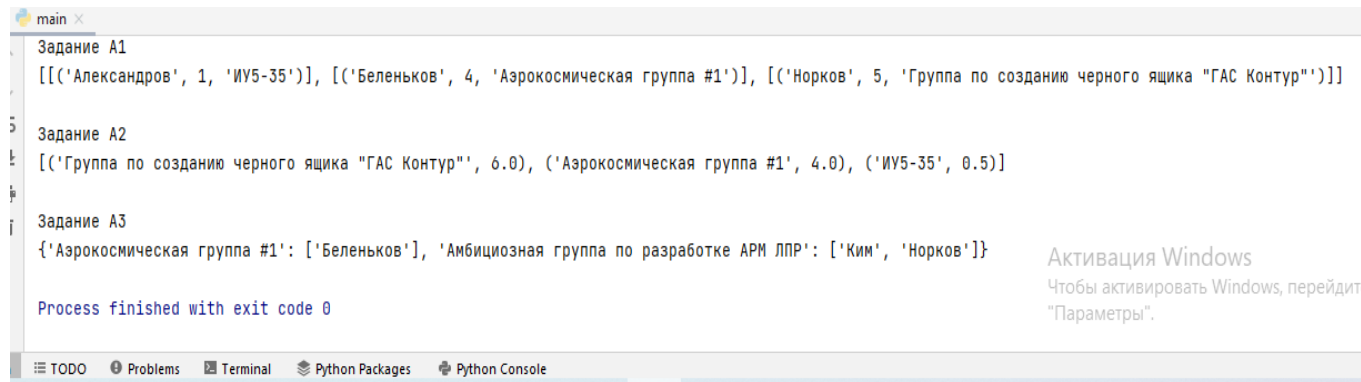
    print('\nЗадание A3')
    res_13 = {}
    # Перебираем все группы
    for g in groups:
        if g.name.startswith('А'): # регистрозависимая операция
            # Список студентов группы
            g_students = list(filter(lambda i: i[2]==g.name, many_to_many))
            # Только ФИО студентов
            g_students_names = [fio for fio, , _ in g_students]
            # Добавляем результат в словарь
            # ключ - группа, значение - список фамилий
            res_13[g.name] = g_students_names

    print(res_13)

```

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

Результаты прпрограммы



The screenshot shows a terminal window titled 'main' with the following output:

```
Задание A1  
[['Александров', 1, 'ИУ5-35']], [['Беленьков', 4, 'Аэрокосмическая группа #1']], [['Норков', 5, 'Группа по созданию черного ящика "ГАС Контур"']]  
  
Задание A2  
[['Группа по созданию черного ящика "ГАС Контур"', 6.0), ('Аэрокосмическая группа #1', 4.0), ('ИУ5-35', 0.5)]  
  
Задание A3  
{'Аэрокосмическая группа #1': ['Беленьков'], 'Амбициозная группа по разработке АРМ ЛПР': ['Ким', 'Норков']}  
  
Process finished with exit code 0
```

On the right side of the terminal, there is a Windows activation watermark:

Активация Windows
Чтобы активировать Windows, перейдите на [страницу](#) "Параметры".

The bottom of the IDE shows tabs for 'TODO', 'Problems', 'Terminal', 'Python Packages', and 'Python Console'.