```python
# используется для сортировки
from operator import itemgetter

# Задания Д вариант 1

class Student:
    """Студент"""

    def __init__(self, id, fio, stipendiya, group_id):
        self.id = id
        self.fio = fio
        self.stipendiya = stipendiya
        self.group_id = group_id


class Group:
    """Группа"""

    def __init__(self, id, name):
        self.id = id
        self.name = name


class StudentClass:
    """
    'Студенты группы' для реализации
    связи многие-ко-многим
    """

    def __init__(self, group_id, student_id):
        self.group_id = group_id
        self.student_id = student_id

# Группы
groups = [
    Group(1, 'ИУ5'),
    Group(2, 'ИУ7'),
    Group(3, 'РК1'),
    Group(4, 'СМ5'),
    Group(5, 'МТ3'),
    Group(6, 'ФН4')
]

# Студенты
students = [
    Student(1, 'Афанасьев', 5000, 1),
    Student(2, 'Бенц', 1700, 2),
    Student(3, 'Большаков', 3500, 1),
    Student(4, 'Коновалов', 12000, 3),
    Student(5, 'Лаптев', 1700, 4),
    Student(6, 'Самойлов', 0, 5),
    Student(7, 'Трифонов', 5000, 5),
    Student(8, 'Шук', 1700, 6),
    Student(9, 'Папшев', 3500, 4),

]

students_classes = [
    StudentClass(1, 1),
    StudentClass(2, 2),
    StudentClass(3, 1),
    StudentClass(4, 3),
    StudentClass(5, 4),
    StudentClass(6, 5),
```

```
    StudentClass(7, 5),
    StudentClass(8, 6),
    StudentClass(9, 4),
]

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

    # Соединение данных один-ко-многим
    one_to_many = [(e.fio, e.stipendiya, d.name)
                   for d in groups
                   for e in students
                   if e.group_id == d.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(d.name, ed.group_id, ed.student_id)
                         for d in groups
                         for ed in students_classes
                         if d.id == ed.group_id]

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

    print('---Задание А1---')
    res_a1 = {}
    for s in students:
        if s.fio[len(s.fio) - 1] == 'в' and s.fio[len(s.fio) - 2] == 'о':
            print(s.fio, groups[(s.group_id) - 1].name)

    print('\n---Задание А2---')
    res_12_unsorted = []
    # Перебираем все группы
    for d in groups:
        # Список студентов в группе
        d_students = list(filter(lambda i: i[2] == d.name, one_to_many))
        # Если группа не пустая
        if len(d_students) > 0:
            # Последняя стипендия студента в группе
            d_last_stip = [last_stip for _, last_stip, _ in d_students]
            # Суммарная последняя стипендия студента в группе
            d_last_stip_sum = sum(d_last_stip)\
            # Средняя последняя стипендия студента в группе
            d_last_stip_av = d_last_stip_sum / len(d_last_stip)
            res_12_unsorted.append((d.name, d_last_stip_av))

    # Сортировка по средней последней стипендии
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    print(res_12)

    print('\n---Задание А3---')
    res_13 = {}
    # Перебираем все группы
    for c in groups:
        if c.name[len(c.name) - 3] == 'И':
            # Список студентов в группе
            d_students = list(filter(lambda i: i[2] == c.name, many_to_many))
            # Только фамилия студента
            d_student_names = [x for x, _, _ in d_students]
            # Добавляем результат в словарь
            # ключ - группа, значение - список фамилий
            res_13[c.name] = d_student_names

    print(res_13)
```

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

```
---Задание А1---
Большаков ИУ5
Коновалов РК1
Самойлов МТ3
Трифонов МТ3

---Задание А2---
[('РК1', 12000.0), ('ИУ5', 4250.0), ('СМ5', 2600.0), ('МТ3', 2500.0), ('ИУ7', 1700.0), ('ФН4', 1700.0)]

---Задание А3---
{'ИУ5': ['Афанасьев'], 'ИУ7': ['Бенц']}
```