текстовый формат

```
from operator import itemgetter


class Emp:
    """Студенческая группа"""

    def __init__(self, id, name, number_of_subjects, number_of_credits) -> None:
        self.id = id
        self.name = name
        self.number_of_subjects = number_of_subjects
        self.number_of_credits = number_of_credits

    def __str__(self) -> str:
        return f"[{self.id}, {self.name}, {self.number_of_subjects}, {self.number_of_credits}]"


class Dep:
    """Учебный курс"""

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

    def __str__(self) -> str:
        return f"[{self.id}, {self.name}]"


class EmpDep:
    """
    'Студенческая группа - Учебный курс' для реализации
    связи многие-ко-многим
    """

    def __init__(self, dep_id, emp_id):
        self.dep_id = dep_id
        self.emp_id = emp_id


# Студенческая группа
deps = [
    Dep(1, 'First'),
    Dep(2, 'Second'),
    Dep(3, 'Third'),

    Dep(11, 'Fourth'),
    Dep(22, 'Fifth'),
    Dep(33, 'Sixth'),
```

```
]

# Учебный курс
emps = [
    Emp(1, 'IU7-12B', 2, 11),
    Emp(2, 'IU5-11B',  1, 11),
    Emp(3, 'IU6-21B', 4, 33),
    Emp(4, 'IU7-32B', 2, 33),
    Emp(5, 'IU7-54B', 8, 2),
]

emps_deps = [
    EmpDep(1, 1),
    EmpDep(1, 2),
    EmpDep(1, 3),
    EmpDep(3, 4),
    EmpDep(2, 5),

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


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

    # Соединение данных один-ко-многим
    one_to_many = [(e.name, e.number_of_subjects, d.name)
                   for d in deps
                   for e in emps
                   if e.id == d.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(d.name, ed.dep_id, ed.emp_id)
                         for d in deps
                         for ed in emps_deps
                         if d.id == ed.dep_id]

    many_to_many = [(e.name, e.number_of_subjects, name)
                    for name, _, emp_id in many_to_many_temp
                    for e in emps if e.id == emp_id]

    print('Задание A1')
    res_11 = sorted(one_to_many, key=itemgetter(2))
    print(res_11)

    print('\nЗадание A2')
    res_12_unsorted = []
    # Перебираем все компьютеры
```

```
    for d in deps:
        # Список микропроцессоров компьютера
        d_emps = list(filter(lambda i: i[2] == d.name, one_to_many))
        # Если в компьютере есть микропроцессоры
        if len(d_emps) > 0:
            # Кол-во ядер микропроцессоров компьютера
            d_sals = [sal for _, sal, _ in d_emps]
            # Сумма ядер микропроцессоров компьютера
            d_sals_sum = sum(d_sals)
            res_12_unsorted.append((d.name, d_sals_sum))

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

    print('\nЗадание А3')
    res_13 = {}
    # Перебираем все компьютеры
    for d in deps:
        if 'First' in d.name:
            # Список микропроцессоров компьютеров
            d_emps = list(filter(lambda i: i[2] == d.name, many_to_many))
            # Только название микропроцессора
            d_emps_names = [x for x, _, _ in d_emps]
            # Добавляем результат в словарь
            # ключ - компьютер, значение - список названий микропроцессоров
            res_13[d.name] = d_emps_names

    print(res_13)


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

#Результаты выполнения:

```
# Задание А1
# [('IU7-12B', 2, 'First'), ('IU5-11B', 1, 'Second'), ('IU6-21B', 4, 'Third')]

# Задание А2
# [('Third', 4), ('First', 2), ('Second', 1)]

# Задание А3
# {'First': ['IU7-12B', 'IU5-11B', 'IU6-21B']}
```