Текст программы

```
class Oper:
 """Оператор"""
        def __init__(self, id, description, syntax, arity, plang_id):
                self.id = id
                self.description = description
                self.syntax = syntax
                self.arity = arity
                 self.plang_id = plang_id


class Plang:
 """Язык программирования"""

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


class OperPlang:
 """
 'Операторы языка' для реализации
 связи многие-ко-многим
 """

        def __init__(self, oper_id, plang_id):
                self.oper_id = oper_id
                 self.plang_id = plang_id


# Языки программирования
plangs = [
        Plang(1, "C++"),
         Plang(2, "C#"),
         Plang(3, "Pascal"),
         Plang(4, "Python"),
         Plang(5, "Java"),
]

# Операторы
opers = [
```

```
        Oper(1, "Array index", "[]", 2, 1),
        Oper(2, "Increment", "++", 1, 1),
        Oper(3, "Equality", "==", 2, 2),
        Oper(4, "Null coalescing", "??", 2, 2),
        Oper(5, "Assignment", ":=", 2, 3),
        Oper(6, "Exponentiation", "**", 2, 4),
        Oper(7, "Ternary operator", "?:", 3, 5),
]

opers_plangs = [
        OperPlang(1, 1),
        OperPlang(1, 2),
        OperPlang(1, 3),
        OperPlang(1, 4),
        OperPlang(1, 5),

        OperPlang(2, 1),
        OperPlang(2, 2),
        OperPlang(2, 5),

        OperPlang(3, 1),
        OperPlang(3, 2),
        OperPlang(3, 4),
        OperPlang(3, 5),

        OperPlang(4, 2),

        OperPlang(5, 3),

        OperPlang(6, 4),

        OperPlang(7, 1),
        OperPlang(7, 2),
        OperPlang(7, 5),
]


def main():
 """Основная функция"""

 # Соединение данных один-ко-многим
```

```
    one_to_many = [(op.description, op.syntax, op.arity, pl.name)
                   for pl in plangs
                   for op in opers
                   if op.plang_id == pl.id]

# Соединение данных многие-ко-многим
    many_to_many_temp = [(pl.name, op_pl.plang_id, op_pl.oper_id)
                         for pl in plangs
                         for op_pl in opers_plangs
                         if pl.id == op_pl.plang_id]

    many_to_many = [(op.description, op.syntax, op.arity, pl_name)
                   for pl_name, pl_id, op_id in many_to_many_temp
                   for op in opers if op.id == op_id]

# Д1 - список всех операторов, оканчивающихся на nt
     print('Задание Д1')
    res_1 = []
    for item in one_to_many:
        # Если две последних буквы названия оператора – nt
        if item[0][-2:] == "nt":
            res_1.append(item)
    print(res_1)

# Д2 - сиок всех языков программирования, отсортированных по средней
арности операторов
    print('Задание Д2')
    res_2 = []
     # Перебираем все языки программирования
    for pl in plangs:
        # Список операторов языка
        pl_ops = list(filter(lambda i: i[3] == pl.name, one_to_many))
        # Если список операторов не пустой
        if len(pl_ops) > 0:
            # арности операторов
            pl_ars = [item[2] for item in pl_ops]
            # Суммарная арность операторов языка
            pl_ars_sum = sum(pl_ars)
            # Средняя арность операторов языка
            res_2.append((pl.name, pl_ars_sum/len(pl_ars)))
```

```
        # Сортировка по средней арности
        print(sorted(res_2, key=lambda item:item[1], reverse=True))

 # Д3 - вывести список языков, начинающихся на C, и список операторов в
них
         print('\nЗадание Д3')
        res_3 = {}
        # Перебираем все языки программирования
        for pl in plangs:
                # Рассматриваем только языки, начинающиеся на C
                if 'C' == pl.name[0]:
                # Список операторов языка
                pl_ops = list(filter(lambda i: i[3] == pl.name, many_to_many))
                # Только описания операторов
                pl_ops_descriptions = [item[0] for item in pl_ops]
                # Добавляем результат в словарь
        # ключ - язык программирования, значение - список операторов
                res_3[pl.name] = pl_ops_descriptions

         print(res_3)


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

Результаты выполнения

Задание Д1

[('Increment', '++', 1, 'C++'), ('Assignment', ':=', 2, 'Pascal')]

Задание Д2

[('Java', 3.0), ('C#', 2.0), ('Pascal', 2.0), ('Python', 2.0), ('C++', 1.5)]

Задание Д3

{'C++': ['Array index', 'Increment', 'Equality', 'Ternary operator'], 'C#': ['Array index', 'Increment', 'Equality', 'Null coalescing', 'Ternary operator']}