Текст программы

main.py

```python
from operator import itemgetter


class Book:
    """Книга"""

    def __init__(self, id, name_b, price, Shop_id):
        self.id = id
        self.name_b = name_b
        self.price = price
        self.Shop_id = Shop_id

class Shop:
    """Книжный магазин"""

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


class BookShop:
    """
    'Книга в книжном' для реализации
    связи многие-ко-многим
    """

    def __init__(self, Shop_id, Book_id):
        self.Shop_id = Shop_id
        self.Book_id = Book_id


# магазины
Shops = [
    Shop(1, 'Достаевский'),
    Shop(2, 'Читай город'),
    Shop(3, 'Республика'),
    Shop(11, 'Фаланастер'),
    Shop(22, 'Московский дом книги'),
    Shop(33, 'Ноты'),
]

# Сотрудники
Books = [
    Book(1, 'Герой нашего времен', 250, 1),
    Book(2, 'Мастер и маргарита', 350, 2),
    Book(3, 'Заводной апельсин', 450, 3),
    Book(4, 'Три товарища', 350, 3),
    Book(5, 'Портрет Дориана Грея', 250, 3),
]

Books_Shops = [
    BookShop(1, 1),
    BookShop(2, 2),
    BookShop(3, 3),
    BookShop(3, 4),
    BookShop(3, 5),
    BookShop(11, 1),
    BookShop(22, 2),
    BookShop(33, 3),
    BookShop(33, 4),
    BookShop(33, 5),
]

one_to_many = [(e.name_b, e.price, d.name)
                   for d in Shops
                   for e in Books
                   if e.Shop_id == d.id]

many_to_many_tBook = [(d.name, ed.Shop_id, ed.Book_id)
                    for d in Shops
                    for ed in Books_Shops
                    if d.id == ed.Shop_id]

many_to_many = [(e.name_b, e.price, Shop_name)
                    for Shop_name, Shop_id, Book_id in many_to_many_tBook
                    for e in Books if e.id == Book_id]

def B1(one_to_many):
    #print('Задание B1')#
    res_11 = sorted(one_to_many, key=itemgetter(2))
    return res_11
def B2(one_to_many):
    #print('\nЗадание B2')
    res_12_unsorted = []
    # Перебираем все магазины
    for d in Shops:
        num = 0
        # Список книг магазина
        d_Books = list(filter(lambda i: i[2] == d.name, one_to_many))
        if len(d_Books) > 0:
            num += 1
            res_12_unsorted.append((d.name, num))
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    return res_12

def B3(many_to_many_tBook):
    print('\nЗадание B3')
    res_13 = {}
    for d in Books:
        if 'и' in d.name_b:
            d_Shops = list(filter(lambda i: i[2] == d.id,
many_to_many_tBook))
            d_Shops_names = [x for x, _, _ in d_Shops]
            res_13[d.name_b] = d_Shops_names
    return res_13

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

    # Соединение данных один-ко-многим
    one_to_many = [(e.name_b, e.price, d.name)
                   for d in Shops
                   for e in Books
                   if e.Shop_id == d.id]

    # Соединение данных многие-ко-многим
    many_to_many_tBook = [(d.name, ed.Shop_id, ed.Book_id)
                          for d in Shops
                          for ed in Books_Shops
                          if d.id == ed.Shop_id]

    many_to_many = [(e.name_b, e.price, Shop_name)
```

```
                    for Shop_name, Shop_id, Book_id in many_to_many_tBook
                    for e in Books if e.id == Book_id]
    print('Задание B1')
    res_11 = sorted(one_to_many, key=itemgetter(2))
    print(res_11)

    print('\nЗадание B2')
    res_12_unsorted = []
    # Перебираем все магазины
    for d in Shops:
        num = 0
        # Список книг магазина
        d_Books = list(filter(lambda i: i[2] == d.name, one_to_many))
        if len(d_Books) > 0:
            num += 1
            res_12_unsorted.append((d.name, num))

    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    print(res_12)

    print('\nЗадание B3')
    res_13 = {}
    for d in Books:
        if 'и' in d.name_b:
            d_Shops = list(filter(lambda i: i[2] == d.id,
many_to_many_tBook))
            d_Shops_names = [x for x, _, _ in d_Shops]
            res_13[d.name_b] = d_Shops_names

    print(res_13)


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

test.py

```
import unittest

from main import *

class TestMain(unittest.TestCase):
    Shops = [
        Shop(1, 'Достаевский'),
        Shop(2, 'Читай город'),
        Shop(3, 'Республика'),
        Shop(11, 'Фаланастер'),
        Shop(22, 'Московский дом книги'),
        Shop(33, 'Ноты'),
    ]
    Books = [
        Book(1, 'Герой нашего времен', 250, 1),
        Book(2, 'Мастер и маргарита', 350, 2),
        Book(3, 'Заводной апельсин', 450, 3),
        Book(4, 'Три товарища', 350, 3),
        Book(5, 'Портрет Дориана Грея', 250, 3),
    ]
    Books_Shops = [
        BookShop(1, 1),
        BookShop(2, 2),
        BookShop(3, 3),
        BookShop(3, 4),
        BookShop(3, 5),
        BookShop(11, 1),
        BookShop(22, 2),
        BookShop(33, 3),
        BookShop(33, 4),
        BookShop(33, 5),
    ]

    def test1(self):
        one_to_many = [(e.name_b, e.price, d.name)
                       for d in Shops
                       for e in Books
                       if e.Shop_id == d.id]
        self.assertEquals(B1(one_to_many), [('Герой нашего времен', 250,
'Достаевский'), ('Заводной апельсин', 450, 'Республика'), ('Три товарища',
350, 'Республика'), ('Портрет Дориана Грея', 250, 'Республика'), ('Мастер и
маргарита', 350, 'Читай город')])

    def test2(self):
        one_to_many = [(e.name_b, e.price, d.name)
                       for d in Shops
                       for e in Books
                       if e.Shop_id == d.id]
        self.assertEquals(B2(one_to_many), [('Достаевский', 1), ('Читай
город', 1), ('Республика', 1)])

    def test3(self):
        many_to_many_tBook = [(d.name, ed.Shop_id, ed.Book_id)
                              for d in Shops
                              for ed in Books_Shops
                              if d.id == ed.Shop_id]
        many_to_many = [(e.name_b, e.price, Shop_name)
                        for Shop_name, Shop_id, Book_id in many_to_many_tBook
                        for e in Books if e.id == Book_id]
        self.assertEquals(B3(many_to_many), {'Мастер и маргарита': ['Читай
город', 'Московский дом книги'], 'Заводной апельсин': ['Республика', 'Ноты'],
'Три товарища': ['Республика', 'Ноты'], 'Портрет Дориана Грея':
['Республика', 'Ноты']})

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

Результат выполнения

```
----------------------------------------------------------------------

Ran 3 tests in 0.001s


OK


Process finished with exit code 0
```

Пример неудачного прохождения тестов

```
======================================================================

FAIL: test2 (__main__.TestMain)

----------------------------------------------------------------------

Traceback (most recent call last):

  File "C:\Users\den20\PiKYAP_23-24\RK1\tests.py", line 46, in test2

    self.assertEquals(B2(one_to_many), [('Доставский', 1), ('Читай город', 1), ('Республика', 1)])

AssertionError: Lists differ: [('Достаевский', 1), ('Читай город', 1), ('Республика', 1)] != [('Доставский', 1), ('Читай город', 1), ('Республика', 1)]


First differing element 0:

('Достаевский', 1)

('Доставский', 1)


- [('Достаевский', 1), ('Читай город', 1), ('Республика', 1)]

?        -


+ [('Доставский', 1), ('Читай город', 1), ('Республика', 1)]


----------------------------------------------------------------------

Ran 3 tests in 0.002s


FAILED (failures=1)


Process finished with exit code 1
```