Рубежного контроль №2 по курсу БКИТ.
Вариант Д25.

1. Класс «Раздел», содержащий поля:
   - ID записи о разделе;
   - Оглавление раздела;
   - Количество страниц (количественный признак);
   - ID записи о документе. (для реализации связи один-ко-многим)
2. Класс «Документ», содержащий поля:
   - ID записи о документе;
   - Наименование документа.
3. (Для реализации связи многие-ко-многим) Класс «Раздел документа», содержащий поля:
   - ID записи о разделе;
   - ID записи о документе.

1. «Документ» и «Раздел» связаны соотношением один-ко-многим. Выведите список оглавления разделов, которые заканчиваются на «ов», и названия их документа.

2. «Документ» и «Раздел» связаны соотношением один-ко-многим. Выведите список документов по среднему количеству страниц раздела в каждом документе, отсортированный по среднему количеству страниц.
3. «Документ» и «Раздел» связаны соотношением многие-ко-многим. Выведите список всех документов, у которых название начинается с буквы «А», и список разделов в нем.

Код программы.

Измененый код перввого рк для модульного тестирования.

```
from operator import itemgetter
class Sec:
    """Раздел"""
    def __init__(self, id, contents, pages, doc_id):
        self.id = id
        self.cont = contents
        self.pages = pages
        self.doc_id = doc_id
class Doc:
    """Документ"""
    def __init__(self, id, name):
        self.id = id
        self.name = name
class SecDoc:
    """Класс ^Раздел документа^(Для реализации связи многие-ко-многим) """
    def __init__(self, sec_id, doc_id):
        self.doc_id = doc_id
```

```
        self.sec_id = sec_id
# Документы
docs = [
    Doc(1, 'Доход'),
    Doc(2, 'Аппартаменты'),
    Doc(3, 'Города')
]

# Разделы
secs = [
    Sec(1, 'Деньги', 1000, 1),
    Sec(2, 'Здания', 1200, 2),
    Sec(3, 'Азов', 500, 3),
    Sec(4, 'Псков', 450, 3),
    Sec(5, 'Проценты', 600, 1),
    Sec(6, 'Сады', 550, 2)
]

secs_docs = [
    SecDoc(1, 1),
    SecDoc(5, 1),
    SecDoc(2, 2),
    SecDoc(6, 2),
    SecDoc(3, 3),
    SecDoc(4, 3)
]

# Соединения данных один ко многим
one_to_many = [(s.cont, s.pages, d.name)
    for d in docs
    for s in secs
    if s.doc_id == d.id]

    # Соединение данных многие ко многим
many_to_many_temp = [(d.name, sd.doc_id, sd.sec_id)
    for d in docs
    for sd in secs_docs
    if d.id == sd.doc_id]
many_to_many = [(s.cont, s.pages, doc_name)
    for doc_name, doc_id, sec_id in many_to_many_temp
    for s in secs
    if s.id == sec_id]

def list_avg(lst):
    if not(len(lst)):
        return 0
    else:
        return sum(lst)/len(lst)

def task_1():
    return [s for s in one_to_many if s[0][-1:] == 'в' and s[0][-2] == 'о']

def task_2():
    res_12_unsorted = []
    for d in docs:
        # Список разделов в документе
        d_secs = [s[1] for s in one_to_many if s[2] == d.name]
        res_12_unsorted.append((d.name, list_avg(d_secs)))
    return sorted(res_12_unsorted, key=itemgetter(1), reverse=True)

def task_3():
    names = (d.name for d in docs if d.name[:1] == 'А')
    return [(name, [s[0] for s in many_to_many if s[2] == name]) for name in
names]
```

Модульное тестирование.

```
import unittest
from main import *

task_1_result = [
    ('Азов', 500, 'Города'), ('Псков', 450, 'Города')
]

task_2_result = [
    ('Аппартаменты', 875), ('Доход', 800), ('Города', 475)
]

task_3_result = [
    ('Аппартаменты', ['Здания', 'Сады'])
]

class RK_test(unittest.TestCase):
    def test_task_1(self):
        self.assertEqual(task_1_result, task_1())

    def test_task_2(self):
        self.assertEqual(task_2_result, task_2())

    def test_task_3(self):
        self.assertEqual(task_3_result, task_3())
```

Результаты тестирования

```
Ran 3 tests in 0.003s

OK

Process finished with exit code 0
```