Шилина А.Ю. ИУ5-31Б;
ВАРИАНТ 25. Раздел. Документ;
Вариант А.

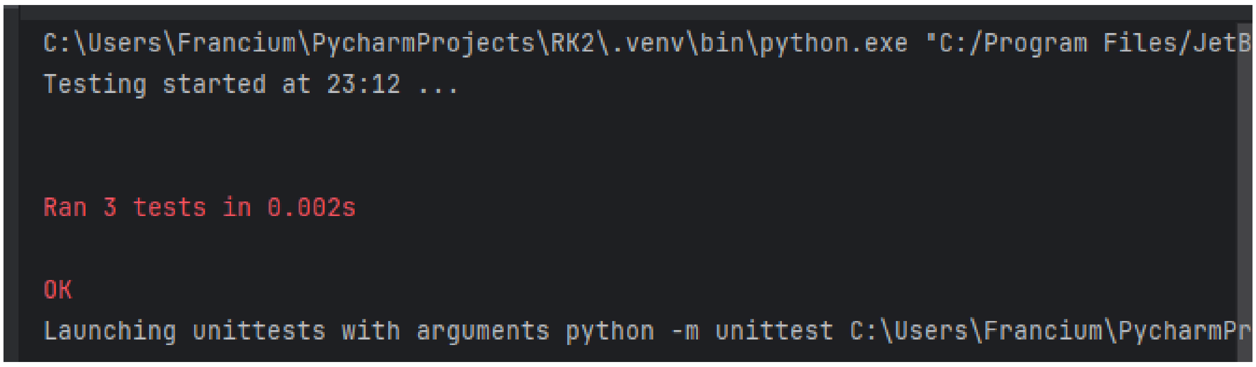
Код программы:

Main.py (рефакторинг + фикс ошибки)

```python
from collections import defaultdict

class Document:
    def __init__(self, id, name, pages, section_id):
        self.id = id
        self.name = name
        self.pages = pages
        self.section_id = section_id

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

class DocSection:
    def __init__(self, doc_id, section_id):
        self.doc_id = doc_id
        self.section_id = section_id

def one_to_many_mapping(documents, sections):
    result = [(doc.name, doc.pages, sec.name) for doc in documents for sec in
sections if doc.section_id == sec.id]
    result.sort(key=lambda x: x[2])
    return result


def many_to_many_mapping(documents, sections, docs_sections):
    """м ко м"""
    section_dict = {s.id: s.name for s in sections}
    result = defaultdict(set)

    for ds in docs_sections:
        for doc in documents:
            if doc.id == ds.doc_id and ds.section_id in section_dict:
                result[section_dict[ds.section_id]].add(doc.name)

    return {key: list(value) for key, value in result.items()}

def task_a2(one_to_many, sections):
    """сум страниц"""
    totals = defaultdict(int)

    for _, pages, section in one_to_many:
        totals[section] += pages

    return sorted(totals.items(), key=lambda x: -x[1])

def task_a3(many_to_many, sections):
    """разделы с 'раздел'(для проверки м ко м поиск по разделу, а не по
отделу, но и так и так работает правильно) и их документы"""
    result = {}
    for section, docs in many_to_many.items():
        if 'раздел' in section.lower():
            result[section] = docs
    return result
```

Файл test.py
```python
import unittest
from main import Document, Section, DocSection, one_to_many_mapping,
many_to_many_mapping, task_a2, task_a3

class TestDocumentProcessing(unittest.TestCase):
    def setUp(self):
        self.sections = [
            Section(1, 'Исторический раздел'),
            Section(2, 'Научный отдел'),
            Section(3, 'Литературный раздел'),
            Section(4, 'Философский отдел'),
            Section(5, 'Раздел математики'),
        ]

        self.documents = [
            Document(1, 'История Древнего Мира', 300, 1),
            Document(2, 'Квантовая механика', 250, 2),
            Document(3, 'Сборник стихов Лермонтова', 200, 3),
            Document(4, 'Метафизика Аристотеля', 150, 4),
            Document(5, 'Геометрия Евклида', 350, 5),
        ]

        self.docs_sections = [
            DocSection(1, 1),
            DocSection(2, 2),
            DocSection(3, 3),
            DocSection(4, 4),
            DocSection(5, 5),
            DocSection(1, 3),
            DocSection(2, 5),
        ]

    def test_one_to_many_mapping(self):
        one_to_many = one_to_many_mapping(self.documents, self.sections)
        expected = [
            ('История Древнего Мира', 300, 'Исторический раздел'),
            ('Сборник стихов Лермонтова', 200, 'Литературный раздел'),
            ('Квантовая механика', 250, 'Научный отдел'),
            ('Геометрия Евклида', 350, 'Раздел математики'),
            ('Метафизика Аристотеля', 150, 'Философский отдел'),
        ]
        self.assertEqual(one_to_many, expected)

    def test_task_a2(self):
        one_to_many = one_to_many_mapping(self.documents, self.sections)
        result = task_a2(one_to_many, self.sections)
        expected = [
            ('Раздел математики', 350),
            ('Исторический раздел', 300),
            ('Научный отдел', 250),
            ('Литературный раздел', 200),
            ('Философский отдел', 150),
        ]
        self.assertEqual(result, expected)

    def test_task_a3(self):
        many_to_many = many_to_many_mapping(self.documents, self.sections,
self.docs_sections)
        result = task_a3(many_to_many, self.sections)

        # Сортируем элементы внутри списков для корректного сравнения
        for key in result:
            result[key] = sorted(result[key])
        expected = {
            'Исторический раздел': ['История Древнего Мира'],
            'Литературный раздел': ['Сборник стихов Лермонтова', 'История
Древнего Мира'],
            'Раздел математики': ['Геометрия Евклида', 'Квантовая механика'],
        }

        for key in expected:
            expected[key] = sorted(expected[key])

        self.assertEqual(result, expected)


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

Результат:



```
C:\Users\Francium\PycharmProjects\RK2\.venv\bin\python.exe "C:/Program Files/JetB
Testing started at 23:12 ...



Ran 3 tests in 0.002s


OK
Launching unittests with arguments python -m unittest C:\Users\Francium\PycharmPr
```