МГТУ им. Н.Э. Баумана

Отчёт по РК №2
по курсу «Парадигмы и конструкции языков программирования»
Вариант 24.

Преподаватель
Гапанюк Ю. Е.

Студент группы ИУ5-31Б
Яковенко С.А.

2023 г.

Текст программы

main.py

```
from operator import itemgetter

class Chapter:
    """Глава"""
    def __init__(self, chapter_id, name, pages, book_id):
        self.chapter_id = chapter_id
        self.name = name
        self.pages = pages
        self.book_id = book_id

class Book:
    """Книга"""
    def __init__(self, id, title):
        self.title = title
        self.id = id

class Chap_of_book:
    """Связь главы и книги"""
    def __init__(self, book_id, chapter_id):
        self.book_id = book_id
        self.chapter_id = chapter_id

def a1_solution(one_to_many):
    return sorted(one_to_many, key=itemgetter(2))

def a2_solution(one_to_many, books):
    res_12_unsorted = []
    for b in books:
        b_chps = list(filter(lambda i: i[2] == b.title, one_to_many))
        if len(b_chps) > 0:
            b_pages = [pages for _, pages, _ in b_chps]
            b_pages_sum = sum(b_pages)
            res_12_unsorted.append((b.title, b_pages_sum))
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    return res_12

def a3_solution(many_to_many, books):
    res_13 = {}
    for b in books:
        if 'Гарри Поттер' in b.title:
            b_chps = list(filter(lambda i: i[2] == b.title, many_to_many))
            b_chps_names = [x for x, _, _ in b_chps]
            res_13[b.title] = b_chps_names
    return res_13

def main():
    """Основная функция"""
    books = [
        Book(1, 'Город костей'),
        Book(2, 'Гарри Поттер от издательства N'),
        Book(3, 'Полианна'),
        Book(22, 'Гарри Поттер от издательства M'),
        Book(33, 'Гарри Поттер от издательства P'),
    ]

    chapters = [
        Chapter(1, 'Хранитель ключей', 21, 2),
        Chapter(2, 'Запретный лес', 17, 2),
        Chapter(3, 'Тайны и ложь', 5, 1),
```

```
        Chapter(4, 'Сумеречный охотник', 10, 1),
        Chapter(5, 'Мисс Полли', 6, 3)
    ]

    chap_books = [
        Chap_of_book(1, 3),
        Chap_of_book(1, 4),
        Chap_of_book(2, 1),
        Chap_of_book(2, 2),
        Chap_of_book(3, 5),
        Chap_of_book(22, 1),
        Chap_of_book(33, 2),
    ]

    one_to_many = [(ch.name, ch.pages, b.title)
                   for b in books
                   for ch in chapters
                   if ch.book_id == b.id]

    many_to_many_temp = [(b.title, chb.book_id, chb.chapter_id)
                         for b in books
                         for chb in chap_books
                         if b.id == chb.book_id]

    many_to_many = [(ch.name, ch.pages, book_title)
                    for book_title, book_id, chapter_id in many_to_many_temp
                    for ch in chapters if ch.chapter_id == chapter_id]

    print('Задание А1')
    print(a1_solution(one_to_many))

    print('\nЗадание А2')
    print(a2_solution(one_to_many, books))

    print('\nЗадание А3')
    print(a3_solution(many_to_many, books))
```

tests.py

```
import unittest
from main import *
class Test_Program(unittest.TestCase):
    # Глобальные переменные
    books = [
        Book(1, 'Город костей'),
        Book(2, 'Гарри Поттер от издательства N'),
        Book(3, 'Полианна'),
        Book(22, 'Гарри Поттер от издательства M'),
        Book(33, 'Гарри Поттер от издательства P'),
    ]

    chapters = [
        Chapter(1, 'Хранитель ключей', 21, 2),
        Chapter(2, 'Запретный лес', 17, 2),
        Chapter(3, 'Тайны и ложь', 5, 1),
        Chapter(4, 'Сумеречный охотник', 10, 1),
        Chapter(5, 'Мисс Полли', 6, 3)
    ]

    chap_books = [
        Chap_of_book(1, 3),
        Chap_of_book(1, 4),
```

```
        Chap_of_book(2, 1),
        Chap_of_book(2, 2),
        Chap_of_book(3, 5),

        Chap_of_book(22, 1),
        Chap_of_book(33, 2),
    ]

    def test_A1(self):
        one_to_many = [(ch.name, ch.pages, b.title)
                       for b in self.books
                       for ch in self.chapters
                       if ch.book_id == b.id]

        self.assertEqual(a1_solution(one_to_many),
                         [('Хранитель ключей', 21, 'Гарри Поттер от
издательства N'),
                          ('Запретный лес', 17, 'Гарри Поттер от издательства
N'),
                          ('Тайны и ложь', 5, 'Город костей'),
                          ('Сумеречный охотник', 10, 'Город костей'),
                          ('Мисс Полли', 6, 'Полианна')])

    def test_A2(self):
        one_to_many = [(ch.name, ch.pages, b.title)
                       for b in self.books
                       for ch in self.chapters
                       if ch.book_id == b.id]

        self.assertEqual(a2_solution(one_to_many, self.books),
                         [('Гарри Поттер от издательства N', 38),
                          ('Город костей', 15),
                          ('Полианна', 6)])

    def test_A3(self):
        many_to_many_temp = [(b.title, chb.book_id, chb.chapter_id)
                             for chb in self.chap_books
                             for b in self.books
                             if b.id == chb.book_id]

        many_to_many = [(ch.name, ch.pages, book_title)
                        for book_title, book_id, chapter_id in
many_to_many_temp
                        for ch in self.chapters if ch.chapter_id ==
chapter_id]

        self.assertDictEqual(a3_solution(many_to_many, self.books),{'Гарри
Поттер от издательства N': ['Хранитель ключей', 'Запретный лес'],
                                                    'Гарри Поттер от
издательства M': ['Хранитель ключей'],
                                                    'Гарри Поттер от
издательства P': ['Запретный лес']})


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

Результат выполнения

```
Ran 3 tests in 0.004s

OK

Process finished with exit code 0
```