Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования» Отчет по РК №2

Вариант запросов: B

Вариант предметной области: 25

|  |  |
| --- | --- |
| Выполнил: | Проверил: |
| студент группы ИУ5-33Б | преподаватель каф. ИУ5 |
| Швецов Даниил | Гапанюк Ю. Е. |

Москва, 2023 г.

# Вариант запросов B. Предметная область 25.

1. «Раздел» и «Документ» связаны соотношением один-ко-

многим. Выведите список всех библиотек, которые начинаются с буквы «А», и названия их языков программирования.

1. «Раздел» и «Документ» связаны соотношением один-ко-

многим. Выведите список языков программирования с минимальный размером библиотеки в каждом языке программирования, отсортированный по

минимальному размеру.

1. «Раздел» и «Документ» связаны соотношением многие-ко-

многим. Выведите список всех связанных библиотек и языков программирования, отсортированный по библиотекам, сортировка по языкам программирования произвольная.

# Задание

1. Проведите рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.
2. Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

# Файл main.py

# используется для сортировки

from operator import itemgetter

from itertools import groupby

class Doc:

    """Документ"""

    def \_\_init\_\_(self, id, name):

        self.id = id

        self.name = name

class Sec:

    """Раздел"""

    def \_\_init\_\_(self, id, name, leng, doc):

        self.id = id

        self.name = name

        self.leng = leng

        self.doc = doc

class SecDoc:

    """

    'Сотрудники отдела' для реализации

    связи многие-ко-многим

    """

    def \_\_init\_\_(self, sec\_id, doc\_id):

        self.doc\_id = doc\_id

        self.sec\_id = sec\_id

def OneToMany(Sec, Doc):

    return [(e.name, e.leng, d.name)

        for d in Doc

        for e in Sec

        if e.doc == d.id]

def ManyToMany(Secs, Doc, SecDoc):

    many\_to\_many\_temp = [(d.name, ed.doc\_id, ed.sec\_id)

        for d in Doc

        for ed in SecDoc

        if d.id==ed.doc\_id]

    many\_to\_many = [(e.name, e.leng, doc\_name)

        for doc\_name, \_, dep\_id in many\_to\_many\_temp

        for e in Secs if e.id==dep\_id]

    return many\_to\_many

# Документы

docs = [

    Doc(1, 'Договор от 21.10.21'),

    Doc(2, 'Соглашение от 29.09.21'),

    Doc(3, 'Устав от 11.05.19'),

    Doc(6, 'Договор от 13.07.22'),

    Doc(7, 'Соглашение от 31.03.22'),

    Doc(8, 'Устав от 01.01.99'),

]

# Разделы

secs = [

    Sec(1, 'Оглавление', 1, 1),

    Sec(2, 'Автоматизация', 10, 1),

    Sec(3, 'Стоимость', 2, 2),

    Sec(4, 'Ансамбль', 5, 2),

    Sec(5, 'Условия', 7, 2),

    Sec(6, 'Предметы и виды деятельности', 3, 3),

    Sec(7, 'Участники', 11, 3),

    Sec(8, 'Уставной Капитал', 1, 8),

    Sec(9, 'преамбула', 5, 7),

    Sec(10, 'реквизиты', 6, 1),

]

secs\_docs = [

    SecDoc(1,1),

    SecDoc(2,1),

    SecDoc(3,2),

    SecDoc(4,2),

    SecDoc(5,2),

    SecDoc(6,3),

    SecDoc(7,3),

    SecDoc(8,8),

    SecDoc(9,7),

    SecDoc(10,1),

    SecDoc(11,1),

    SecDoc(22,2),

    SecDoc(33,3),

    SecDoc(33,4),

    SecDoc(33,5),

]

def get\_first(data):

    return list(filter(lambda x: x[0].lower()[0] == 'а', data))

def get\_second(data):

    grouped = groupby(data, lambda x: x[2])

    data = []

    for i in grouped:

        data.append(min(i[1], key = lambda x: x[1]))

    return sorted(data, key = lambda x: x[1])

def get\_third(data):

    return sorted(data, key= lambda x: x[0])

def main():

    """Основная функция"""

    # Соединение данных один-ко-многим

    one\_to\_many = OneToMany(secs, docs)

    # Соединение данных многие-ко-многим

    many\_to\_many = ManyToMany(secs, docs, secs\_docs)

    print('Задание B1')

    print("\n".join(map(lambda x: f"Документ : {x[2]}, Раздел : {x[0]}, Длинной {x[1]}", get\_first(one\_to\_many))))

    print("\nЗадание В2")

    print("\n".join("Документ: {2}, Раздел минимальной длины: {0}, Длина {1}".format(\*i) for i in get\_second(one\_to\_many)))

    print("\nЗадание В3")

    print("\n".join(map(lambda x: f"Документ: {x[2]}, Раздел: {x[0]}, длина: {x[1]}", get\_third(many\_to\_many))))

if \_\_name\_\_ == '\_\_main\_\_':

    main()

# Файл tests.py

import unittest

from main import \*

class TestJoinOneToMany(unittest.TestCase):

    def test\_one\_to\_many(self):

        result = OneToMany(secs, docs)

        self.assertTrue(len(result) == 10)

class TestJoinManyToMAny(unittest.TestCase):

    def test\_many\_to\_many(self):

        result = ManyToMany(secs, docs, secs\_docs)

        self.assertTrue(len(result) == 10)

class TestResults(unittest.TestCase):

    def test\_get\_a1\_result(self):

        data = [("Секция 1", 5, "Договор 1"), ("А Секция 2", 5, "Договор 1"), ("Секция 3", 5, "Договор 2")]

        result = get\_first(data)

        self.assertEqual(result, [("А Секция 2", 5, "Договор 1")])  # Проверка корректности результата

    def test\_get\_a1\_empty\_result(self):

        data = []  # пустой список

        result = get\_first(data)

        self.assertEqual(result, [])  # ожидаемый результат - пустой список

    def test\_get\_a2\_result(self):

        data = [("Секция 1", 5, "Договор 1"), ("А Секция 2", 6, "Договор 1"), ("Секция 3", 5, "Договор 2")]

        result = get\_second(data)

        self.assertEqual(result, [("Секция 1", 5, "Договор 1"), ("Секция 3", 5, "Договор 2")])  # Проверка корректности результата

    def test\_get\_a3\_result(self):

        data = [("Секция 1", 5, "Договор 1"),

                ("А Секция 2", 6, "Договор 1"),

                ("Секция 3", 5, "Договор 2")

        ]

        result = sorted(data, key=lambda x: x[0])  # ожидаемый результат - отсортированный по имени библиотеки

        self.assertEqual(result, get\_third(data))

if \_\_name\_\_ == '\_\_main\_\_':

# Результат выполнения

**Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание**