Защищено:
Гапанюк Ю.Е.

"__"_____2025 г.

Демонстрация:
Гапанюк Ю.Е.

"__"_____2025 г.

Отчет по рубежному контролю № 1 по курсу Парадигмы и конструкции языков программирования ГУИМЦ

Тема работы: "Рубежный контроль представляет собой разработку программы на языке Python."

3
(количество листов)

ИСПОЛНИТЕЛЬ:

студент группы ИУ5Ц-53Б _____
(подпись)

Гаджиев А.Г.

"__"_____2025 г.

Москва, МГТУ - 2025 г.

1. Тема и задание для выполнения рубежного контроля.

Тема работы: Рубежный контроль представляет собой разработку программы на языке Python, которая выполняет следующие действия:

1) Необходимо создать два класса данных в соответствии с Вашим вариантом предметной области, которые связаны отношениями один-ко-многим и многие-ко-многим.

Пример классов данных для предметной области Сотрудник-Отдел:

1. Класс «Сотрудник», содержащий поля:
   - ID записи о сотруднике;
   - Фамилия сотрудника;
   - Зарплата (количественный признак);
   - ID записи об отделе. (для реализации связи один-ко-многим)
2. Класс «Отдел», содержащий поля:
   - ID записи об отделе;
   - Наименование отдела.
3. (Для реализации связи многие-ко-многим) Класс «Сотрудники отдела», содержащий поля:
   - ID записи о сотруднике;
   - ID записи об отделе.

2) Необходимо создать списки объектов классов, содержащих тестовые данные (3-5 записей), таким образом, чтобы первичные и вторичные ключи соответствующих записей были связаны по идентификаторам.

3) Необходимо разработать запросы в соответствии с Вашим вариантом. Запросы сформулированы в терминах классов «Сотрудник» и «Отдел», которые используются в примере. Вам нужно перенести эти требования в Ваш вариант предметной области. При разработке запросов необходимо по возможности использовать функциональные возможности языка Python (list/dict comprehensions, функции высших порядков).

Для реализации запроса №2 введите в класс, находящийся на стороне связи «много», произвольный количественный признак, например, «зарплата сотрудника».

Вариант Д.

1. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список всех сотрудников, у которых фамилия заканчивается на «ов», и названия их отделов.
2. «Отдел» и «Сотрудник» связаны соотношением один-ко-многим. Выведите список отделов со средней зарплатой сотрудников в каждом отделе, отсортированный по средней зарплате (отдельной функции вычисления среднего значения в Python нет, нужно использовать комбинацию функций вычисления суммы и количества значений)..
3. «Отдел» и «Сотрудник» связаны соотношением многие-ко-многим. Выведите список всех отделов, у которых название начинается с буквы «А», и список работающих в них сотрудников.

Варианты предметной области

№ варианта: 25
Класс 1: Раздел
Класс 2: Документ

2. Листинг программы

```
# используется для сортировки
from operator import itemgetter

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

class Document:
    """Документ"""
    def __init__(self, id, title, author, section_id):
        self.id = id
        self.title = title
        self.author = author
        self.section_id = section_id

class DocumentSection:
    """Документы раздела для реализации связи многие-ко-многим"""
    def __init__(self, section_id, document_id):
        self.section_id = section_id
        self.document_id = document_id

# Разделы
sections = [
    Section(1, 'Административные документы'),
    Section(2, 'Аналитические материалы'),
    Section(3, 'Научные работы'),
    Section(4, 'Техническая документация'),
    Section(5, 'Отчетные документы'),
]

# Документы
documents = [
    Document(1, 'Приказ №1', 'Иванов', 1),
    Document(2, 'Анализ рынка', 'Петров', 2),
    Document(3, 'Исследование AI', 'Сидоров', 3),
    Document(4, 'Техническое задание', 'Кузнецов', 4),
    Document(5, 'Годовой отчет', 'Смирнов', 5),
    Document(6, 'Акт выполненных работ', 'Николаев', 1),
    Document(7, 'Статья по математике', 'Васильев', 3),
    Document(8, 'Анализ производительности', 'Федоров', 2),
]

# Связи многие-ко-многим
documents_sections = [
    DocumentSection(1, 1),
    DocumentSection(1, 6),
    DocumentSection(2, 2),
    DocumentSection(2, 8),
    DocumentSection(3, 3),
    DocumentSection(3, 7),
    DocumentSection(4, 4),
    DocumentSection(5, 5),
    # Дополнительные связи для демонстрации многие-ко-многим
    DocumentSection(1, 2),
```

```
    DocumentSection(3, 5),
]

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

    # Соединение данных один-ко-многим
    one_to_many = [(d.title, d.author, s.name)
                   for s in sections
                   for d in documents
                   if d.section_id == s.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(s.name, ds.section_id, ds.document_id)
                         for s in sections
                         for ds in documents_sections
                         if s.id == ds.section_id]

    many_to_many = [(d.title, d.author, section_name)
                    for section_name, section_id, document_id in many_to_many_temp
                    for d in documents if d.id == document_id]

    print('Задание Д1')
    # Список всех документов, у которых автор заканчивается на "ов", и названия их разделов
    res_1 = list(filter(lambda i: i[1].endswith('ов'), one_to_many))
    print(res_1)

    print('\nЗадание Д2')
    # Список разделов со средней длиной названия документов в каждом разделе,
    # отсортированный по средней длине
    res_2_unsorted = []
    # Перебираем все разделы
    for s in sections:
        # Список документов раздела
        s_docs = list(filter(lambda i: i[2] == s.name, one_to_many))
        # Если раздел не пустой
        if len(s_docs) > 0:
            # Длины названий документов раздела
            doc_title_lengths = [len(title) for title, _, _ in s_docs]
            # Средняя длина названия документов (сумма длин / количество)
            avg_title_length = sum(doc_title_lengths) / len(doc_title_lengths)
            res_2_unsorted.append((s.name, round(avg_title_length, 2)))

    # Сортировка по средней длине названия
    res_2 = sorted(res_2_unsorted, key=itemgetter(1))
    print(res_2)

    print('\nЗадание Д3')
    # Список всех разделов, у которых название начинается с буквы «А»,
    # и список документов в них
    res_3 = {}
    # Перебираем все разделы
    for s in sections:
        if s.name.startswith('А'):
            # Список документов раздела
            s_docs = list(filter(lambda i: i[2] == s.name, many_to_many))
            # Только названия документов
            s_docs_titles = [title for title, _, _ in s_docs]
            # Добавляем результат в словарь
```

```
        # ключ - раздел, значение - список названий документов
        res_3[s.name] = s_docs_titles

    print(res_3)

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

3. Результаты работы программы

```
PS C:\Users\Амид\Desktop\Лабы> & C:\Users\Амид\AppData\Local\Microsoft\WindowsApps\python3.11.exe c:/Users/Амид/Desktop/Лабы/Rk1.py
```

Задание Д1

[('Приказ №1', 'Иванов', 'Административные документы'), ('Анализ рынка', 'Петров', 'Аналитические материалы'), ('Анализ производительности', 'Федоров', 'Аналитические материалы'), ('Исследование AI', 'Сидоров', 'Научные работы'), ('Техническое задание', 'Кузнецов', 'Техническая документация'), ('Годовой отчет', 'Смирнов', 'Отчетные документы')]

Задание Д2

[('Отчетные документы', 13.0), ('Административные документы', 15.0), ('Научные работы', 17.5), ('Аналитические материалы', 18.5), ('Техническая документация', 19.0)]

Задание Д3

{'Административные документы': ['Приказ №1', 'Акт выполненных работ', 'Анализ рынка'], 'Аналитические материалы': ['Анализ рынка', 'Анализ производительности']}