ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ МОЛДОВЫ

ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАТИКИ

ДЕПАРТАМЕНТ ИНФОРМАТИКИ

**MAMALIGA ARTUR**

**Отчет**

по дисциплине „ПРОГРАММИРОВАНИЕ В PYTHON”

Руководитель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Плешка Наталья, лектор

(подпись)

Автор: Mamaliga

(подпись)

Кишинев, 2024

**Задача:**

Написать скрипт на Python для управления информацией об авторах и их книгах. Скрипт должен использовать словарь для хранения данных, где ключами будут имена авторов, а значениями – списки их книг. Пользователю предоставляется меню с возможностями:  
Скрипт должен предусматривать обработку ошибок, таких как ввод некорректных данных, и выводить обновленную информацию после каждой операции. Все функции должны быть реализованы в отдельном модуле.  
  
**Логика реализованных алгоритмов:**

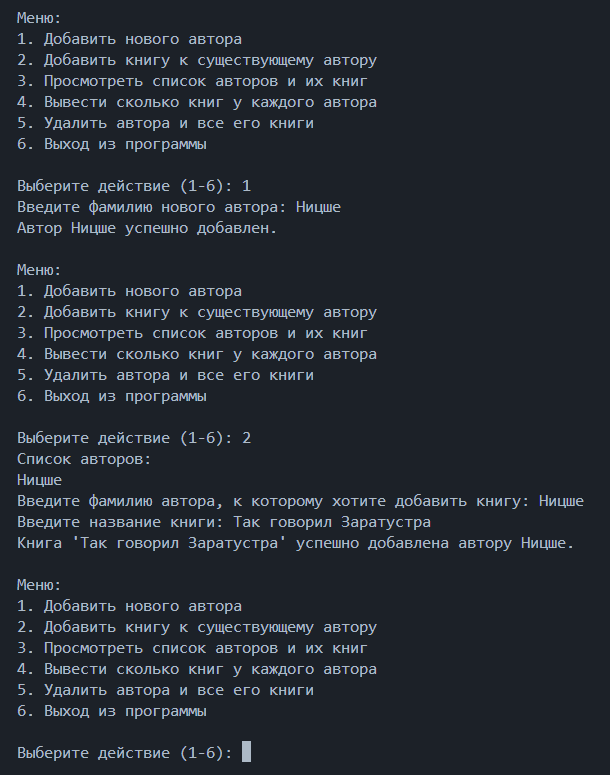
Использование словаря для хранения информации об авторах и списка для хранения информации о их книгах.

Меню создано при помощи бесконечного цикла и условной конструкции которая сверяет число введенное пользователем отождествляющее выбор, впоследствие решая какую функцию вызвать. Выход из программы реализован оператором brake.

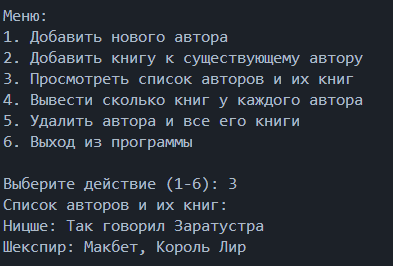
Каждая операция вынесена в отдельную функцию.

1. Операция добавить автора сначала проверяет на то введена ли фамилия автора (не оставленно ли поле ввода пустым), проверяет есть ли уже такой автор в словаре, и если есть то выводит то, что он уже есть и выходит из функции. Если же поле не пустое и такого автора еще нет в словаре то добавит его в словарь и выведет сообщение об успешном добавлении.
2. Операция добавить книгу к существующему автору еще в меню выведет функцию которая выведет фамилии всех добавленных авторов, после спросит у пользователя фамилию автора и название книге, после вызовет функцию добавления книги. Функция добавить книгу проверит условия корректности данных введенных пользователем, а именно не пусто ли поле фамилии автора, есть ли данный автор в словаре, не пусто ли поле книги, есть ли уже такая книга. Если все условия соблюдены добавиться новая книга
3. Операция просмотреть список авторов и их книг при помощи цикла фор выведет каждого автора и его книги.
4. Операция вывести сколько книг у каждого автора циклом фор выведет фамилию каждого автора и длину списка его книг.
5. Операция удалить автора и все его книги проверит сначала введено ли поле автора, есть ли введенный автор в словаре и если есть удалит вхождение его ключа в словаре и вместе с этим список.

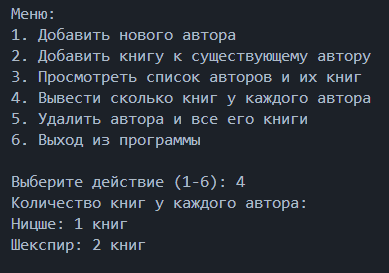
**Скриншоты взаимодействия с пользователем:**



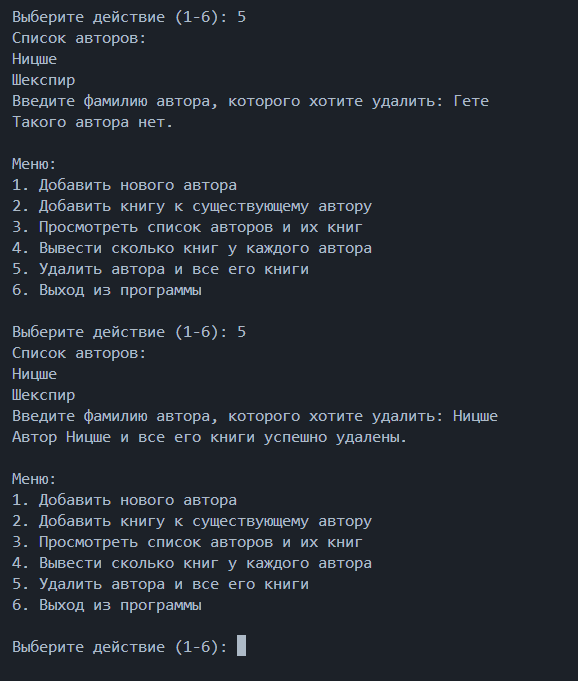
Тут пользователь создает в словаре вхождение с ключем Ницше и значением пустой список. После добавляется книга в этот список по ключу Ницше.



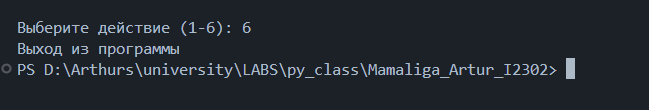
Тут выбрано просмотреть список авторов и их книг, что вывело всех авторов и книги им добавленные.



Тут я вывел список авторов и количество их книг.



Тут я пробую сначала удалить автора, которого нет в словаре, но программа мне этого не позволяет, а после я удаляю из словаря Ницше.



Данным образом реализован выход из программы.

**Структуры данных и описание функционала:**

В данном задании были использованны словарь и список для хранения информации о писателях и их книгах.

author\_dict = dict()

Так как словарь для хранения авторов и их книг один и неразрывно связан с функционалом. Я имею в виду, что функции могут работать только с конкретным словарем, а не впринципе с любым, так как нам важно то чтобы структура словаря сохранялась и не было ключей типа int к примеру, из этого следуюет то, что я решил разместить словарь внутри модуля с функциями, ибо только с ним они работать и будут, хотя можно и было каждый раз передавать функции словарь, но как я выше объяснил я посчитал это нецелесообразным.

Словарь инициализировал функцией dict(), хотя по сути это вызов конструктора словаря, я мог использовать синтаксический сахар в виде author\_dict = {} но как по мне использование конструктора, а не литерала более явно передает суть действий.

\*словарь - это структура данных, которая используется для хранения пар ключ-значение. Каждый ключ уникален и связан с определенным значением.

def add\_author(author):

    if not author:

        print("Вы не ввели фамилию автора, попробуйте снова")

    elif author in author\_dict:

        print("Такой автор уже существует.")

    else:

        author\_dict[author] = list() # добавление нового ключа в словарь со значением пустой список

        print(f"Автор {author} успешно добавлен.")

здесь я создаю новое вхождение в словарь с ключем типа string который является фамилией автора. В качестве значения я передаю список созданный функцией list().

\*список - это упорядоченная коллекция элементов, которая может содержать объекты любого типа, включая другие списки.

def list\_authors():

    print("Список авторов:")

    for author in author\_dict.keys():

        print(author)

функция, которая нужна для того чтобы вывести фамилии всех авторов чтобы пользователю было удобно смотря на то что вывела эта функция вводить фамилию автора когда программа от него это потребует.

def add\_book(author, book):

    # проверка условий для работы корректной работы функции.

    if not author:

        print("Вы не ввели фамилию автора.")

    elif author not in author\_dict.keys():

        print("Такого автора нет.")

    elif not book:

        print("Вы не ввели название книги.")

    elif book in author\_dict[author]:

        print("Такая книга уже есть в списке у этого автора.")

    else:

        author\_dict[author].append(book)

        print(f"Книга '{book}' успешно добавлена автору {author}.")

функция добавления книги, сперва проверяет корректность введенных пользователем данных, а далее добавит с помощью метода append книгу в список принадлежащий заданному автору. Выведется же результат операции при помощи форматирования строки. В последней строке а именно функции print – f говорит о том что строка должна быть отформатирована, а переменные в фигурных скобках будут заменены их значениями.

def list\_authors\_books():

    print("Список авторов и их книг:")

    for author, books in author\_dict.items():

        print(f"{author}: {', '.join(books)}")

def count\_books():

    print("Количество книг у каждого автора:")

    for author, books in author\_dict.items():

        print(f"{author}: {len(books)} книг")

первая функция выведет всех авторов и их книги, используя цикл фор в поля author и book соответственно будут помещены ключ и значение (вместо author и book можно было бы использовать слова key и value). Благодаря методу join который возвращает в виде строки соедененные строкой элементы какого то множества, я вывожу все книги через запятую.

Функция подсчета книг выведет длину списка для каждого элемента словаря, благодаря функции len() я получаю длину списка.

def delete\_author(author):

    if not author:

        print("Вы не ввели фамилию автора.")

    elif author not in author\_dict:

        print("Такого автора нет.")

    else:

        del author\_dict[author]

        print(f"Автор {author} и все его книги успешно удалены.")

Данная функция проверяет корректность ввода, а после просто удаляет элемент словаря с ключем переданным в аргументе функции

from func import \*

while True:

    print("\nМеню:")

    print("1. Добавить нового автора")

    print("2. Добавить книгу к существующему автору")

    print("3. Просмотреть список авторов и их книг")

    print("4. Вывести сколько книг у каждого автора")

    print("5. Удалить автора и все его книги")

    print("6. Выход из программы\n")

    choice = input("Выберите действие (1-6): ")

    if choice == '1':

        author = input("Введите фамилию нового автора: ")

        add\_author(author)

    elif choice == '2':

        list\_authors()

        author = input("Введите фамилию автора, к которому хотите добавить книгу: ")

        book = input("Введите название книги: ")

        add\_book(author, book)

    elif choice == '3':

        list\_authors\_books()

    elif choice == '4':

        count\_books()

    elif choice == '5':

        list\_authors()

        author = input("Введите фамилию автора, которого хотите удалить: ")

        delete\_author(author)

    elif choice == '6':

        print("Выход из программы")

        break

    else:

        print("Некорректный ввод. Пожалуйста, выберите действие из меню.")

В первой строке я импортирую все из модуля func чтобы не использовать для вызова функций слово func, к примеру func.add\_book()

Далее используя бесконечный цикл я вывожу меню и сверяю введенное пользователем число с соответствующим символом который соответствует каждому пункту меню.  
Оператор break прервет цикл и программа прекратиться.

**Вывод:**  
Язык Python показался удобным и эффективным в решении поставленной задачи. Встроенные структуры данных, такие как словари и списки, позволяют легко управлять информацией. Использование функций делает код более читаемым и модульным. Python также обладает широким спектром инструментов для обработки пользовательского ввода и вывода, что делает разработку приложений с интерфейсом более удобной.