

ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ МОЛДОВЫ  
ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАТИКИ  
ДЕПАРТАМЕНТ ИНФОРМАТИКИ

**Отчет**  
**по дисциплине „ПРОГРАММИРОВАНИЕ В PYTHON”**

Руководитель: Плешка Наталья, лектор

Выполнил: студент группы  
I2302 Богданов Юрий

Кишинев, 2024

## Вариант 1. Управление данными о книгах автора.

### Краткая постановка задачи:

Задача заключается в создании скрипта на языке Python для учета информации об авторах и их книгах. Программа должна предоставлять пользователю следующие возможности:

1. Добавление нового автора (его фамилии).
2. Добавление книги к существующему автору (книг может быть несколько).
3. Просмотр списка авторов и их книг.
4. Вывод количества книг у каждого автора.
5. Удаление автора и всех его книг из словаря.
6. Выход из программы.

Необходимо предусмотреть обработку ошибок, таких как ввод некорректных данных, и выводить обновленную информацию после каждой операции. Каждая из возможностей должна быть представлена в виде функции, а все функции следует хранить в отдельном модуле для лучшей организации кода.

### Логика реализованных алгоритмов:

Давайте рассмотрим логику каждого пункта из нашего меню:

Для начала необходимо добавить нового автора. Это можно сделать выбрав первую опцию нашего меню, с помощью функции **addAuthor**. Данная функция работает таким образом, что для начала она просит пользователя ввести фамилию автора, которого он просит добавить в словарь. Далее проверяется существует ли автор с такой же фамилией в словаре *infoAuthorsBooks*, если же он уже есть, то выводится сообщение об ошибке "Этот автор уже есть.". Если автора нет, создается новая запись в словаре с ключом - фамилией автора и пустым списком книг. После этого возвращается обновленный словарь *infoAuthorsBooks* , с добавленным автором.

Вторая опция нашего меню позволяет добавить книгу к существующему автору с помощью функции **addBook**. После вызова данной функции у пользователя запрашивается фамилия автора к которому он хочет добавить книгу. После ввода фамилии проверяется, существует ли автор с такой фамилией в словаре

*infoAuthorsBooks*. Если такого автора нет, выводится сообщение об ошибке: "Этого автора нет.". Если пользователь ввел фамилию автора, который есть в словаре, то теперь он может ввести название книги, которую хочет добавить к желаемому автору. Название книги добавляется в список книг данного автора в словаре *infoAuthorsBooks*. После этого возвращается обновленный словарь *infoAuthorsBooks*, с добавленной книгой.

Третья опция нашего меню позволяет пользователю просмотреть список авторов и их книг, с помощью функции **viewAuthors**. Она работает так, что для каждого автора в словаре *infoAuthorsBooks* выводится его фамилия и список его книг. Если же у автора нет добавленных книг, то выводится только его фамилия.

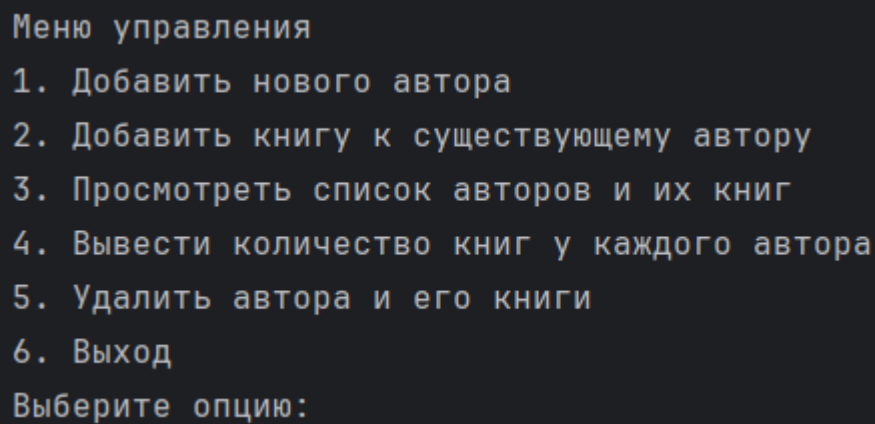
Четвертая опция меню позволяет пользователю вывести на экран количество книг у каждого автора, с помощью функции **viewBooksCount**. Она работает так, что для каждого автора в словаре *infoAuthorsBooks* выводится его фамилия и количество его книг. Если же у автора нет добавленных книг, то выводится только его фамилия с нулевым количеством книг.

Пятая опция меню позволяет пользователю удалить автора и все его книги, с помощью функции **deleteAuthor**, которая работает по следующему принципу, пользователь вводит фамилию автора, которого хочет удалить.

Далее проверяется, существует ли автор с такой фамилией в словаре *infoAuthorsBooks*. Если такого автора нет, выводится сообщение об ошибке: "Этого автора нет.". Если пользователь ввел фамилию автора, который есть в словаре, то его запись удаляется из словаря *infoAuthorsBooks*. После этого возвращается обновленный словарь *infoAuthorsBooks*, без удалённого автора.

Наконец последняя шестая опция нашего меню позволяет пользователю выйти из программы. При выборе этой опции программа выводит сообщение "До свидания!". Выполняется *break*, который завершает цикл нашего меню *while True*, и программа завершается.

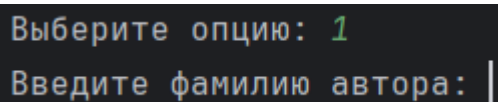
## Скрины интерфейсов для взаимодействия с пользователем и их назначение:



```
Меню управления
1. Добавить нового автора
2. Добавить книгу к существующему автору
3. Просмотреть список авторов и их книг
4. Вывести количество книг у каждого автора
5. Удалить автора и его книги
6. Выход
Выберите опцию:
```

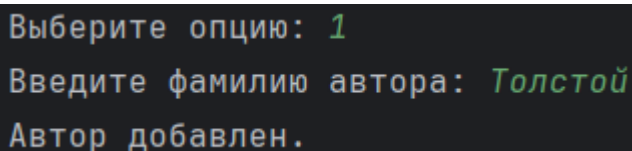
На данном скриншоте показано наше меню, которое видит пользователь и с которым он может взаимодействовать.

Начнём:



```
Выберите опцию: 1
Введите фамилию автора: |
```

На данном скриншоте показан выбор опции 1 для добавления нового автора. Например:



```
Выберите опцию: 1
Введите фамилию автора: Толстой
Автор добавлен.
```

Далее:

```
Выберите опцию: 2
Введите автора книги:
```

На данном скриншоте показан выбор опции 2 для добавления новой книги.

Например:

```
Выберите опцию: 2
Введите автора книги: Толстой
Введите название книги:
```

```
Выберите опцию: 2
Введите автора книги: Толстой
Введите название книги: Война и Мир
Книга добавлена.
```

Далее:

```
Выберите опцию: 3
Список авторов и их книг:
Толстой: Война и Мир
Пушкин: Евгений Онегин, Дубровский
```

На данном скриншоте показан выбор опции 3 для просмотра списка ранее добавленных авторов и их книг.

Далее:

```
Выберите опцию: 4
Количество книг у каждого автора:
Пушкин: 2
Толстой: 1
Достоевский: 0
```

На данном скриншоте показан выбор опции 4 для просмотра количества книг у ранее добавленных авторов.

Далее:

```
Выберите опцию: 5
Введите фамилию автора, которого хотите удалить: |
```

На данном скриншоте показан выбор опции 5 для удаления автора и его книг(если они есть).

Например:

```
Выберите опцию: 5
Введите фамилию автора, которого хотите удалить: Достоевский
Автор удален.
```

Далее:

```
Выберите опцию: 6
До свидания!
```

На данном скриншоте показан выбор опции 6 для выхода из программы.

### **Структуры данных, с которыми работали в приложении (использовались для хранения данных):**

Для реализации задачи создания скрипта на языке Python для учета информации об авторах и их книгах мы используем словари для хранения информации об авторах и их книгах.

Главная структура данных - это словарь *infoAuthorsBooks*, для хранения информации об авторах и их книгах, где ключами являются фамилии авторов, а значениями - их книги.

Также присутствует вложенный список *infoAuthorsBooks[author]* ( в словаре *infoAuthorsBooks*), который используется для хранения списка книг каждого автора.

## Описание функционала мини-приложения, строками кода (кусками) и пояснениями:

Для начала рассмотрим содержимое файла Variant 1.py

```
import functions as func
```

```
infoAuthorsBooks = {}
```

Здесь мы импортируем модуль **functions** с псевдонимом **func** и инициализируем пустой словарь *infoAuthorsBooks*, который будет использоваться для хранения информации об авторах и их книгах.

```
while True:
```

```
    print("\nМеню управления")
    print("1. Добавить нового автора")
    print("2. Добавить книгу к существующему автору")
    print("3. Просмотреть список авторов и их книг")
    print("4. Вывести количество книг у каждого автора")
    print("5. Удалить автора и его книги")
    print("6. Выход")
```

Для функционала мини-приложения создаем меню с помощью цикла `while True`, который является бесконечным и будет выполняться до тех пор пока условие не будет ложным, но так как у нас здесь всегда условие истинно, цикл будет выполняться до тех пор пока пользователь не выберет 6 опцию и будет выполнен `break`, который завершит цикл и программа завершится.

```
choice = input("Выберите опцию: ")

if choice == "1":
    infoAuthorsBooks = func.addAuthor(infoAuthorsBooks)
elif choice == "2":
    infoAuthorsBooks = func.addBook(infoAuthorsBooks)
elif choice == "3":
    func.viewAuthors(infoAuthorsBooks)
elif choice == "4":
    func.viewBooksCount(infoAuthorsBooks)
elif choice == "5":
    infoAuthorsBooks = func.deleteAuthor(infoAuthorsBooks)
elif choice == "6":
    print("До свидания!")
    break
else:
    print("Вы выбрали несуществующую опцию! Выберите из существующих.")
```

Данный фрагмент кода используем для обработки выбора пользователя и вызова соответствующих функций.

**choice = input("Выберите опцию: ")**:

Запрашивает у пользователя ввод выбранной опции. Далее , используем оператор **if...elif...else**.

**if choice == "1":**  
    infoAuthorsBooks = func.addAuthor(infoAuthorsBooks)

В начале срабатывает **if** ,если значение переменной choice равно 1, то выполняется функция **addAuthor** из модуля functions, которая добавляет нового автора в словарь *infoAuthorsBooks* и возвращает обновленный словарь.



```
elif choice == "2":  
    infoAuthorsBooks = func.addBook(infoAuthorsBooks)  
    .  
    .  
    .  
elif choice == "5":  
    infoAuthorsBooks = func.deleteAuthor(infoAuthorsBooks)
```

Далее если пользователь не выбрал 1 опцию, то срабатывает оператор **elif** (с 2 по 5), программа проверяет следующее условие: равно ли оно 2, 3, 4 или 5. Если условие выполняется, соответствующая функция вызывается для выполнения соответствующего действия. Например, если **choise** равно 2, то вызывается функция **addBook**, которая добавляет книгу к существующему автору.

```
elif choice == "6":  
    print("До свидания!")  
    break  
else:  
    print("Вы выбрали несуществующую опцию! Выберите из существующих.")
```

В данном случае, если пользователь выберет опцию 6, то программа выполнит **break**, что приведет к выходу из цикла **while**, и **else**-блок не выполнится. Но, если ввод пользователя не соответствует ни одной из опций (т.е., не равен 1, 2, 3, 4, 5, или 6), цикл продолжит выполнение, и когда он завершится, программа выполнит **else**-блок, выводя сообщение "Вы выбрали несуществующую опцию! Выберите из существующих."

Мы рассмотрели содержимое файла Variant 1.py теперь перейдем к рассмотрению функций из модуля **functions.py**.

```
def addAuthor(infoAuthorsBooks):
    author = input("Введите фамилию автора: ")
    if author in infoAuthorsBooks:
        print("Этот автор уже есть.")
    else:
        infoAuthorsBooks[author] = []
        print("Автор добавлен.")
    return infoAuthorsBooks
```

Для начала функция запрашивает у пользователя ввод фамилии нового автора и сохраняет ее в переменную *author*. С помощью **if...in** проверяется, присутствует ли уже фамилия автора в словаре *infoAuthorsBooks*. Если да, то выводится сообщение о том, что этот автор уже есть. Если фамилия автора не найдена в словаре, то создается новая запись с ключом, равным фамилии автора, и соответствующим пустым списком для книг этого автора. Затем выводится сообщение о том, что автор успешно добавлен. И после этого обновленный словарь *infoAuthorsBooks* возвращается из функции после выполнения всех операций.

```
def addBook(infoAuthorsBooks):
    author = input("Введите автора книги: ")
    if author not in infoAuthorsBooks:
        print("Этого автора нет.")
    else:
        books = input("Введите название книги: ")
        infoAuthorsBooks[author].append(books)
        print("Книга добавлена.")
    return infoAuthorsBooks
```

Эта функция является аналогичной предыдущей, только в отличии от предыдущей она использует **not...in** проверяя, если автора нет, то выводит этого автора нет, а также если пользователь ввел правильную фамилию которая есть, то у него запрашивается ввод названия книги, которая сохраняется в переменную **books**. Затем название книги добавляется в список книг этого автора в словаре *infoAuthorsBooks*, и выводится сообщение о том, что книга успешно добавлена. И после этого

обновленный словарь *infoAuthorsBooks* возвращается из функции после выполнения всех операций.

```
def viewAuthors(infoAuthorsBooks):  
    print("Список авторов и их книг:")  
    for author, books in infoAuthorsBooks.items():  
        print(author + ": " + ", ".join(books))
```

Выводится заголовок, который указывает, что будет выведен список авторов и их книг. С помощью метода *items()* интегрируем по элементам словаря *infoAuthorsBooks*. Переменная **author** принимает ключ (имя автора), а переменная **books** принимает значение (книги этого автора). После этого выводится имя автора, за которым следует двоеточие, а затем список его книг. *",".join(books)* объединяет названия книг в списке *books* с помощью запятых и пробелов.

```
def viewBooksCount(infoAuthorsBooks):  
    print("Количество книг у каждого автора:")  
    for author, books in infoAuthorsBooks.items():  
        print(author + ": " + str(len(books)))
```

Функция работает примерно так же как и предыдущая, но отличие в том, что здесь уже выводится количество книг, которое определяется с помощью функции *len(books)*, которая возвращает количество книг. Поскольку *len()* возвращает целое число, его результат преобразуется в строку с помощью *str()*, чтобы он мог быть выведен с помощью *print()*.

```
def deleteAuthor(infoAuthorsBooks):  
    author = input("Введите фамилию автора, которого хотите удалить: ")  
    if author in infoAuthorsBooks:  
        del infoAuthorsBooks[author]  
        print("Автор удален.")  
    else:  
        print("Этого автора нет.")  
    return infoAuthorsBooks
```

Для начала у пользователя запрашивается ввод фамилии автора, которого он хочет удалить, и сохраняется в переменной *author*. С помощью *if...in* проверяется, присутствует ли уже фамилия автора в словаре *infoAuthorsBooks*. Если автор присутствует в словаре, он

удаляется из него с помощью оператора `del`, и выводится сообщение о том, что автор успешно удален. Если же автора нет, выводится сообщение о том, что этого автора нет. Наконец, обновленный словарь *infoAuthorsBooks* возвращается из функции после выполнения всех операций.

Мы рассмотрели все присутствующие функции в модуле `functions.py`.

### **Выводы – как показался язык Python в решении задачи.**

После выполнения поставленной задачи я могу сделать выводы, что использование Python показалось мне очень эффективным и удобным для решения этой задачи благодаря своей простоте, выразительности и мощным средствам работы с коллекциями данных, такими как словари и списки. Использование функций и модулей способствует повышению читаемости кода.