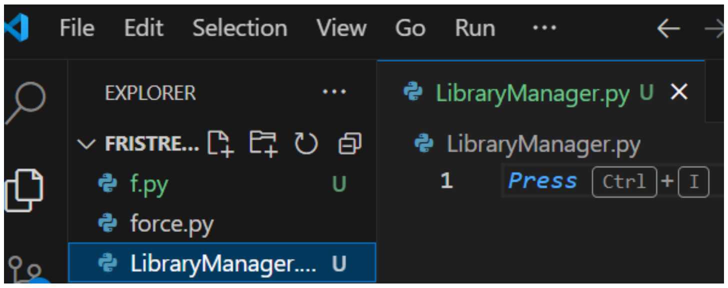
Overview
In this worksheet, we will implement the concepts of abstract data types. This Python code simulates a basic library management system. It initializes empty lists, a set, and a dictionary to store information about books. Users can add books to the library, search for a book by title, display all books in the library, and remove a book from the library.

Pre-requisites:

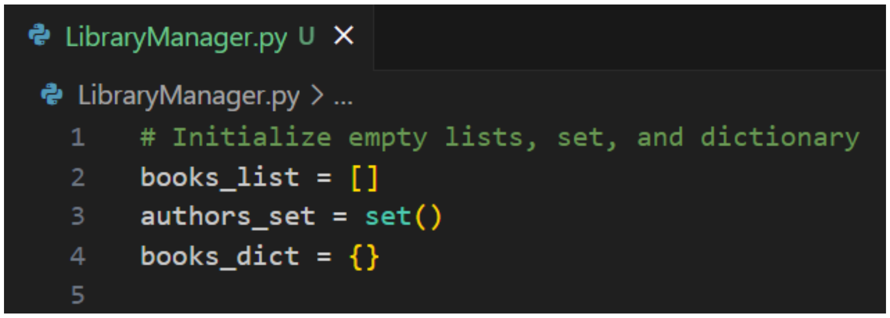
1. Github account
2. Git
3. VSCode
4. Python
5. Basic understanding of Python syntax.
6. Basic understanding of abstract data types and its operations
7. Familiarity with git commands

Instructions:

1. **Open the cloned folder in VSCode**.
    a. Create a python file called "LibraryManager.py"



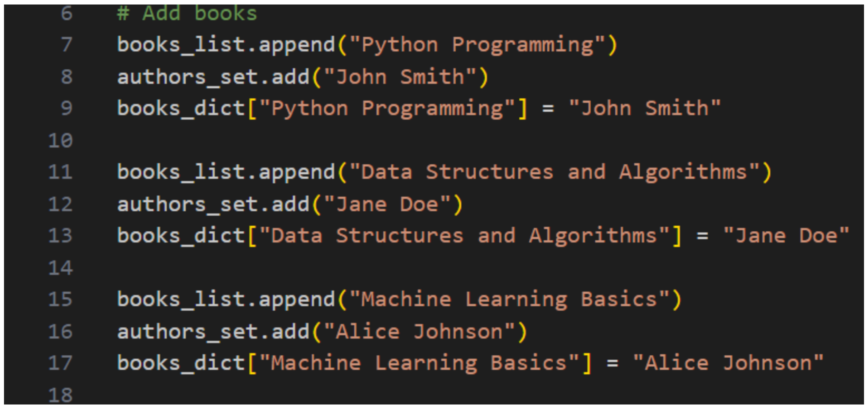
2. **Write a program**:
    a. The code starts by initializing three empty data structures: books_list, authors_set, and books_dict. These will store the titles, authors, and a mapping of titles to authors, respectively. In a file called LibraryManager.py, add the following code.



```python
# Initialize empty lists, set, and dictionary
books_list = []
authors_set = set()
books_dict = {}

```

    b. Three books are added to the library using append for the list, add for the set, and direct assignment for the dictionary.



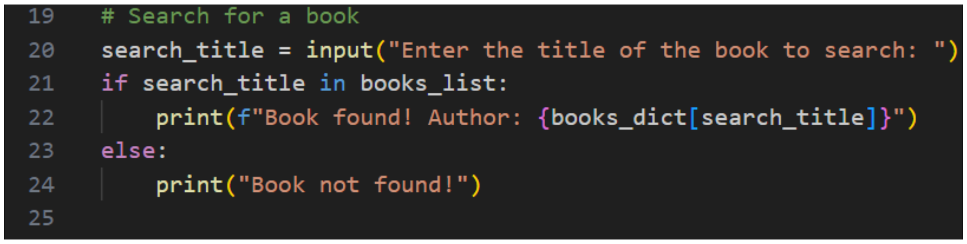
```python
# Add books
books_list.append("Python Programming")
authors_set.add("John Smith")
books_dict["Python Programming"] = "John Smith"

books_list.append("Data Structures and Algorithms")
authors_set.add("Jane Doe")
books_dict["Data Structures and Algorithms"] = "Jane Doe"

books_list.append("Machine Learning Basics")
authors_set.add("Alice Johnson")
books_dict["Machine Learning Basics"] = "Alice Johnson"

```

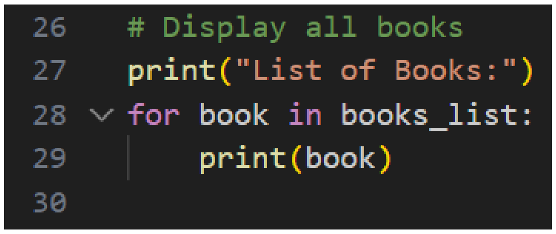
c. Searching for a Book:
   i. The user is prompted to input the title of the book they want to search for.
   ii. The code checks if the entered title exists in the books_list.
   iii. If the book is found, it prints the book's title and author by accessing the information from books_dict.
   iv. If the book is not found, it prints a message indicating that the book was not found.



```python
# Search for a book
search_title = input("Enter the title of the book to search: ")
if search_title in books_list:
    print(f"Book found! Author: {books_dict[search_title]}")
else:
    print("Book not found!")

```

d. Displaying All Books:
   i. All the books stored in books_list are displayed using a for loop.
   ii. It iterates through each book title in the list and prints it.

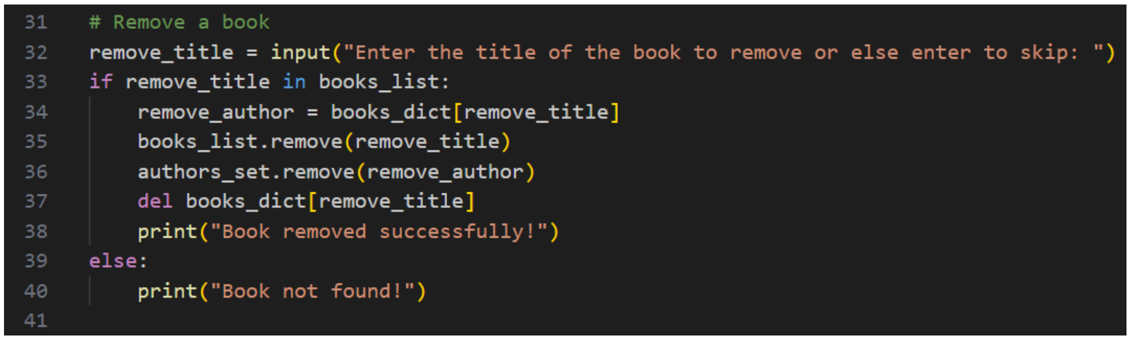


```python
# Display all books
print("List of Books:")
for book in books_list:
    print(book)

```

e. Removing a Book:
   i. The user is prompted to input the title of the book they want to remove or press Enter to skip the removal process.
   ii. The code checks if the entered title exists in books_list.
   iii. If the book is found, It retrieves the author's name associated with the book title from books_dict.
   iv. It removes the book title from books_list.
   v. It removes the author's name from authors_set.

vi. It deletes the book entry from books_dict.
vii. It prints a success message.
viii. If the book is not found, it prints a message indicating that the book was not found.



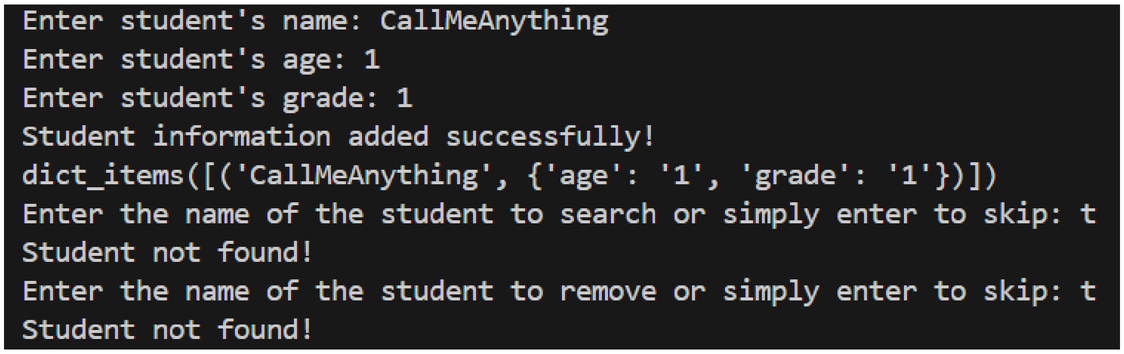
```python
# Remove a book
remove_title = input("Enter the title of the book to remove or else enter to skip: ")
if remove_title in books_list:
    remove_author = books_dict[remove_title]
    books_list.remove(remove_title)
    authors_set.remove(remove_author)
    del books_dict[remove_title]
    print("Book removed successfully!")
else:
    print("Book not found!")
```

Exercise:
Create a simple Student Information Management System.

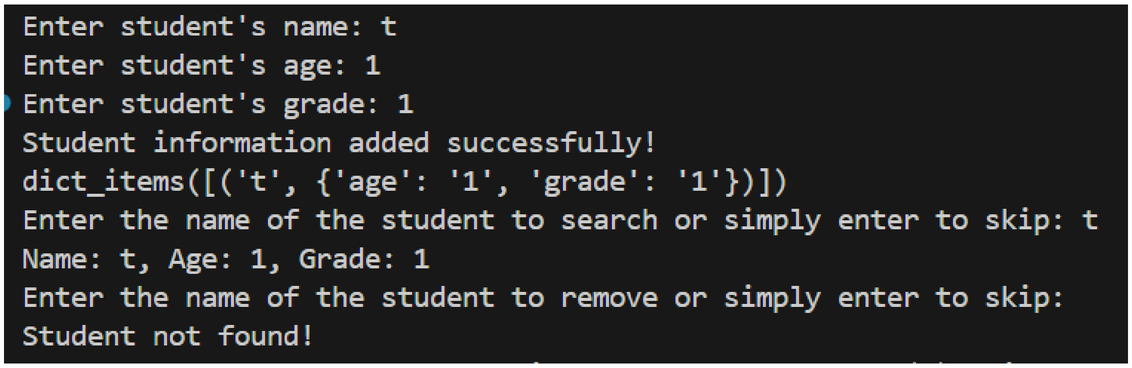
Instructions:

- Initialize empty lists and a dictionary to store student information. students_list as a list to store student names and students_dict as a dictionary to store student names as keys and their corresponding information (age and grade) as values.
- Prompt the user to input the name, age, and grade of a student and add this information in lists and dictionaries. Then print a success message. Additionally, print the items of the dictionary to view student details.
- Allow the user to search for a student by their name. If found, display the student's name, age, and grade.
- Allow the user to remove a student from the system by entering their name. If found, remove the student's information from all data structures.
- Test your system thoroughly by adding, searching and removing student information.

Sample Input/Output:



```
Enter student's name: CallMeAnything
Enter student's age: 1
Enter student's grade: 1
Student information added successfully!
dict_items([('CallMeAnything', {'age': '1', 'grade': '1'})])
Enter the name of the student to search or simply enter to skip: t
Student not found!
Enter the name of the student to remove or simply enter to skip: t
Student not found!
```



```
Enter student's name: t
Enter student's age: 1
Enter student's grade: 1
Student information added successfully!
dict_items([('t', {'age': '1', 'grade': '1'})])
Enter the name of the student to search or simply enter to skip: t
Name: t, Age: 1, Grade: 1
Enter the name of the student to remove or simply enter to skip:
Student not found!
```