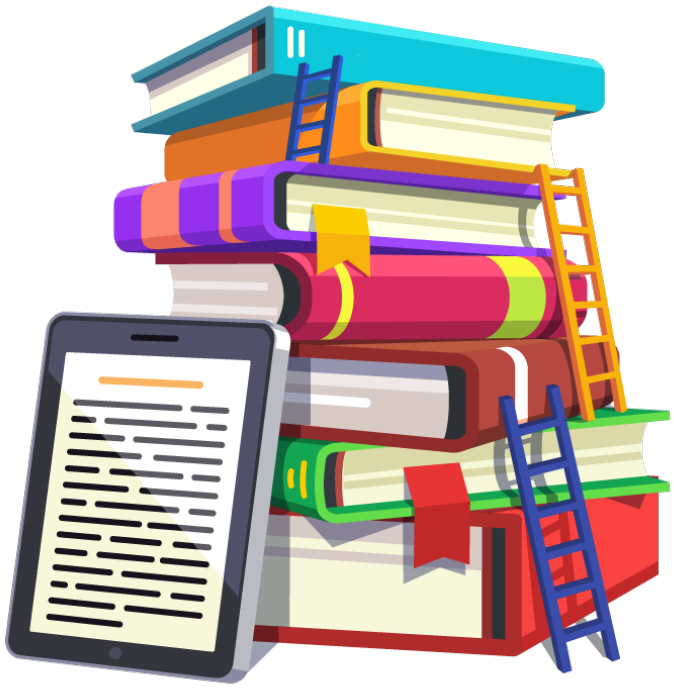
ACIBADEM UNIVERSITY

SPRING 2024

CSE 102 – PROGRAMMING PRACTICE

Assignment 2: Library Management System

(Assigned 6 May 2024 15:00, Due: 19 May 2024 23:59)
(Late Submission 20 May 2024 00:00 – 24 May 2024 23:59)



Objective: In this assignment, you will create a simple Library Management System using Python, focusing on Object-Oriented Programming (OOP) principles. While creating this system, you will remember and use the information you have learned so far. The system will provide the opportunity to add users to the library, add books, remove books, query whether the book is available with ISBN, list, borrow and return books.

Before you start:
Just fill in the name, surname and student_id values in the .py extension file in the assignment folder you downloaded. All values you fill in are strings, do not change their type!

```
############ Do not change the assignment code value ############
assignment_code = 140110202
name = ""
surname = ""
student_id = ""
### Do not change the variable names above, just fill them in ###
```

Attention: After that please update the solution.py file to be your studentID_solution.py. For example, if your school number is 210402005, your file name should be 210402005_solution.py.

Glossary:

ISBN (International Standard Book Number): A unique numerical identifier to identify a particular book

Student ID: A personal identifying number for students.

Attention: Before you start writing code for the assignment, please read the **Before you start:** section and make sure you follow the required steps. When you complete the assignment, make sure you read the **Before you send:** section on the last page and follow the instructions.

Requirements:
There are 3 classes in this assignment. These are Book, User and Library. All Book and User classes are ready for you, do not make any changes. The prototype of the Library class has been shared with you, you just need to write the methods that are required of you.

Attention: Do not change method names.

You will need previously defined classes to write the methods, so before you start writing the methods, please read the information below and understand the code in the solution.py file given to you.

Book class is the class that keeps the necessary information when adding a new book to the library. The features included in this class are title, author, publish date, ISBN, is_borrow and borrower. is_borrow holds the value of whether the book is currently borrowed or not. The borrower value holds the student ID of the person who bought the book if it was borrowed. By default, is_borrow value is given as False and borrower value is given as None. Therefore, you do not need to assign any value to these two values when defining a book.

User class is the class that allows adding new users who register to the library. The properties of this class are first name, last name and student id values. Student ID value is unique for each user.

Certain features and methods of the Library class have been shared with you. The methods requested from you are given function names as prototypes. The *_save_data()* and *_load_data()* methods provided to you save your operations in the *library_data.json* file, allowing you to work on the existing data the next time you run the program. The *_load_data()* method checks whether there is data in the library_data.json file and loads this data if there is. The *_save_data()* method saves the changes you make here through other methods to the *library_data.json* file.

Info: *_save_data()* and *_load_data()* methods to your Library class to save and read your data, there is no need to access these functions from abroad that's why they start with underscore.

Attention: It is forbidden to use any library other than the json library in this assignment. Json library was also used to read and save data from *library_data.json* file. For this, the file given to you has already been imported, you do not need to take any action.

Attention: The methods you need to save in the json file after performing the operation are specified in the definitions. You can use the *self._save_data()* method for this.

Note: The action that each method should perform is stated in the definition section. The value it should return is shown in the return values section.

Attention: Each method will return the specified value, not print.

Attention: Take Student ID and ISBN values as integers.

- **Library** class should have the following methods:
  - ***add_book(self, title, author, publish date, isbn):***
    Definition:
    This method takes the title, author, publish date and ISBN number required to add a book, and if the book is not in the library, it adds the book and saves it in the json file. If it exists, it does not add it. (10 points)

    Return Values:
    If the book is added successfully,
    `return book.title`
    Otherwise,
    `return False`

  - ***add_user(self, first_name, last_name, student_id)***:
    Definition:
    This method takes first_name, last_name and student_id values and creates this user and saves it in the json file if there is no such user. If there is, it does nothing. Check whether the user exists via student_id, which is unique for each user. (10 points)

    Return Values:
    If the user has been added successfully,
    `return user.student_id`
    Otherwise,
    `return False`

  - ***check_book_by_isbn(self, isbn):***
    Definition:
    This method checks whether the book is in the library by taking the ISBN parameter. (10 points)

    Return Values:
    If the book is available in the library,
    `return True`
    Otherwise,
    `return False`

  - ***remove_book(self, isbn)***:
    Definition:
    This method checks whether the book is in the library by taking the ISBN parameter. If it is in the library, it deletes the book and saves the json file. If not, it does not change anything. (15 points)

    Hint: You can use the .remove() feature for this method.

    Return Values:
    If the book is deleted successfully,
    `return book.title`
    Otherwise,
    `return False`

- ***delete_user(self, student_id)**:*
  Definition:
  This method takes the student ID value as a parameter. If the user is registered to the library and does not currently have any books borrowed, it deletes the user and saves it in the json file. (20 points)

  Hint: You can use the .remove() feature for this method.

  Return Values:
  If the user is deleted successfully,
  `return user.student_id`
  Otherwise,
  `return False`

- ***list_books(self)**:*
  Definition:
  This method does not take any parameters. Returns the ISBN of all books in the library as a list. (10 points)

  Return Values:
  It always returns a **list**. If there is at least one book, the ISBN number will be in the list, if there is no book, return an empty list.

- ***borrow_book(self, isbn, user):***
  Definition:
  If the book and the user are registered in the library, it lends the book to the user. It sets the is_borrowed value of the loaned book to True, and the borrower value gives the student_id value of the person and saves it in the json file. (25 points)

  Return Values:
  In case of successful borrowing of the book
  `return True`
  In all cases otherwise
  `return False`

- ***return_book(self, isbn)**:*
  Definition:
  If the book is available in the library and was taken by someone, the book is returned. Sets is_borrow to False. Sets the borrower value to None then saves it to json file. (20 points)

  Return Values:
  If the book has been successfully returned,
  `return True`
  All situations otherwise
  `return False`

Attention: Since your codes are also tested with autograder, make sure that the specified function names and class names are the same.

For example, your JSON output should look like this.

library_data.json

library_data ⟩ No Selection

```json
{
  "books": [
    {
      "title": "The Call of the Wild",
      "author": "Jack London",
      "publish_date": "1903",
      "isbn": 9781945644511,
      "is_borrowed": true,
      "borrower": 15040203
    },
    {
      "title": "Animal Farm",
      "author": "George Orwell",
      "publish_date": "2003",
      "isbn": 9780452284241,
      "is_borrowed": false,
      "borrower": null
    }
  ],
  "users": [
    {
      "first_name": "Selim",
      "last_name": "Altay",
      "student_id": 15040203
    },
    {
      "first_name": "Aylin",
      "last_name": "Kaya",
      "student_id": 17140107
    }
  ]
}
```

ATTENTION: You will not create the JSON output manually, the program we created will record the transactions made and read the codes here the next time it is run.

NOTE: To obtain the above json output, the following codes were run in a folder where there was no library_data.json file.

```
library = Library()
```

```
library.add_book("The Call of the Wild", "Jack London", "1903", 9781945644511)
```

```
'The Call of the Wild'
```

```
library.add_book("Animal Farm", "George Orwell", "2003", 9780452284241)
```

```
'Animal Farm'
```

```
library.add_book("Animal Farm", "George Orwell", "2003", 9780452284241)
```

```
False
```

```
library.add_user("Selim", "Altay", 15040203)
```

```
15040203
```

```
library.add_user("Selim", "Altay", 15040203)
```

```
False
```

```
library.borrow_book(9781945644511, 15040203)
```

```
True
```

```
library.add_user("Aylin", "Kaya", 17140107)
```

```
17140107
```

```
library.delete_user(15040203)
```

```
False
```

SUGGESTION: You can also test your code on a notebook.

ATTENTION: The name of the JSON file must be library_data.json. Make sure you do not change this name.

ATTENTION: Variable names and types in the JSON file must be the same as in the example above.

NOTE: If your variable names and types are the same, it's okay if your JSON output is not exactly the same as above.

Test your code:
Ensure that the system prevents the following:

- Adding the same book to the library multiple times.
- Borrowing the same book by a user multiple times.
- There cannot be two users with the same student ID.
- If the user has already borrowed a book, it will not delete the user in the *delete_user* method.

Ensure that the system does the following:

- After each operation, it writes the data to JSON file.
- A user can borrow more than one book.
- When the program restarts, it reads the existing data and creates a new JSON file if it does not exist.
- Make sure that Student ID and ISBN values are taken as integers.

Late Submission Policy:
This assignment is rated out of 120 for on-time submission. For late submission, each score above is calculated by multiplying out of 5/6. Therefore, the maximum grade that can be received for late submission is 100. Late submission is between 20 May 00.00 - 24 May 23.59.

Before you send:
Create a new folder and name it your student ID. Then, put only the solution.py file, which contains only the answers, into this folder. Do not put any other files in the folder. Then format the folder as ZIP or RAR. For example, if your school number is 210402005, the name of your folder should be 210402005 and there should only be 210402005_solution.py file in it. The folder should be sent in ZIP or RAR format only.

Submission:
Upload the ZIP or RAR formatted folder within the time specified on the LMS (EYS).