Object Oriented Programming

Lab 09

Programs consisting of multiple classes

- A class is like a blueprint for a thing. When you create a class, you're just making a **blueprint**, like drawing out architectural plans for a building. That building, however, has yet to be created. You can also take these plans and create tons of buildings that all are formed in the same way

- A program may contain any number of classes.
  - Model your application in terms of objects/classes first and
  - then connect them to build (think about the relationship between them)

OOP – What's the difference

- Object oriented programming changes our thinking around what our code does
  - instead of thinking of functions being run by a script and primitive variables storing data, we think about objects existing in our program which have information about themselves (fields) and “things” that they can do (methods)
  - Instead of having a script which runs functions, we have programs which create objects which interact with each other (call other objects’ methods)

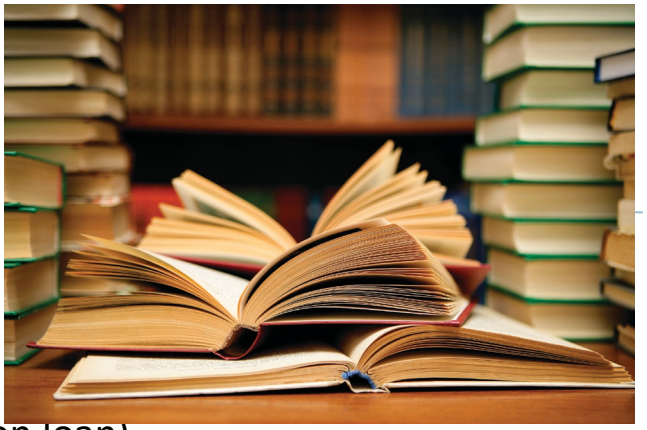
Advantages

- Readability:
  - It is much harder to browse and write code inside a file with 10000 lines than several little and organized files.
- Re-usability:
  - If you write a single class, you can slip up in code duplication. This means more lines of code and probably more bugs(!)
- Testability:
  - What about testing a single functionality? If you isolate one logic functionality in one class, your life will be easier.
- Code maintainability:
  - You can fix a bug or enhance functionality with multiplie classes in one single place: The nice little classes.

Lab09

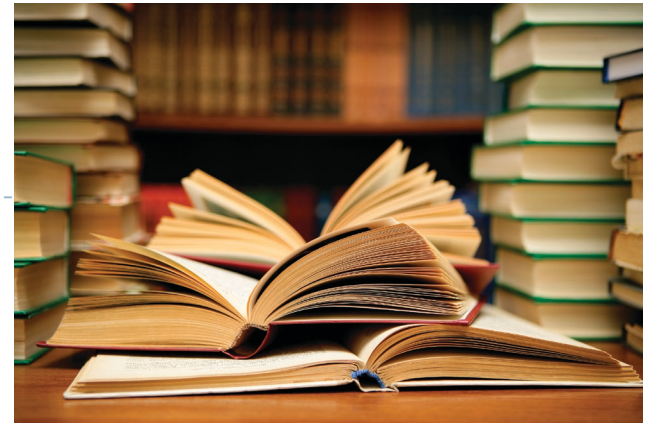
- You are required to implement a simple library system.
- Your program should allow the users to borrow, return, and search for books in the collection.
- The program also provides the functionalities such as displaying the list of books and searching by user on the books borrowed, etc.

The Lab



- You will need to make four classes
  - Book class
    - Represents a book in the library system
    - Fields: __code, __tile, __status (True = available, False = on loan)
    - Methods: is_available(), borrow_book(), return_book(), etc.
  - Member class
    - Represents a mem in the library system
    - Fields: __member_id, __name, __on_loan_books_list
    - methods: borrow_book(book), return_book(book), etc.
  - Record class
    - Represents a borrowing record in the library system
    - Fields: __book, __member, __is_on_loan, __issue_date
    - Methods: is_on_load(), return_book(), etc.
  - MyLibrary class
    - Represents a library system
    - Fields: __books_list, __on_loan_records_list
    - Methods: show_available_books(), find_book(code), borrow_book(…), find_record(code), return_book(…)

Work Flow



- Load all books from a text file
- Borrow a book
  - Create a member object
  - Find a book by a book code (and it is available)
  - If found -> borrow the book

Add a new Record
Set the book's status to on loan
Add the book to the on-loan book lists of the member

- Return a book
  - Find a record by a book code (and it is on loan)
  - If found -> return the book

Update the record's status
Set the book's status to available
Remove the book from the on-loan book lists of the member

Coderunner Tips!

- Before you start writing a new method STOP!!! And think:
  - What types of things is this method going to have to do?
  - What methods does THIS class have? And what methods do the objects I have available to me have?
  - Can I use any of those pre-existing methods in this new method?

- Q5: in the get_member_id() method we need to get the member id from the __member field. To do this we can use self.__member.get_member_id()
  You can do the same thing to get the book code in the get_book_code method
- Q6: you will need to call the return_book method on the book and the record – and remember to set the __is_on_loan field to False
- Q7: A try-except-else might be helpful here