Library Simulator

You will be writing a Library simulator involving multiple classes. You will write the LibraryItem, Patron and Library classes, and the three classes that inherit from LibraryItem (Book, Album and Movie). All data members of each class should be marked as private (a leading underscore in the name). Since they're private, if you need to access them from outside the class, you should do so via get or set methods. Any get or set methods should be named per the usual convention ("get\_" or "set\_" followed by the name of the data member).

Here are descriptions of the three classes:

LibraryItem:

library\_item\_id - a unique identifier for a LibraryItem - you can assume uniqueness, you don't have to enforce it

title - cannot be assumed to be unique

location - a LibraryItem can be "ON\_SHELF", "ON\_HOLD\_SHELF", or "CHECKED\_OUT"

checked\_out\_by - refers to the Patron who has it checked out (if any)

requested\_by - refers to the Patron who has requested it (if any); a LibraryItem can only be requested by one Patron at a time

date\_checked\_out - when a LibraryItem is checked out, this will be set to the current\_date of the Library

init method - takes a library item ID and title; checked\_out\_by and requested\_by should be initialized to None; a new LibraryItem's location should be on the shelf

get\_location returns the Library Item's location

other get and set methods as needed

Book/Album/Movie:

These three classes all inherit from LibraryItem.

All three will need a method called get\_check\_out\_length that returns the number of days that type of library item may be checked out for. For a Book it's 21 days, for an Album it's 14 days, and for a Movie it's 7 days.

All three will have an additional field. For Book, it's a string field called author. For Album, it's a string field called artist. For Movie, it's a string field called director. There will also need to be get methods to return the values of these fields.

Patron:

patron\_id - a unique identifier for a Patron - you can assume uniqueness, you don't have to enforce it

name - cannot be assumed to be unique

checked\_out\_items - a collection of LibraryItems that a Patron currently has checked out

fine\_amount - how much the Patron owes the Library in late fines (measured in dollars); this is allowed to go negative

init method - takes a patron ID and name

get\_fine\_amount - returns the fine\_amount

other get and set methods as needed

add\_library\_item - adds the specified LibraryItem to checked\_out\_items

remove\_library\_item - removes the specified LibraryItem from checked\_out\_items

amend\_fine - a positive argument increases the fine\_amount, a negative one decreases it; this is allowed to go negative

Library:

holdings - a collection of the LibraryItems that belong to the Library

members - a collection of the Patrons who are members of the Library

current\_date - stores the current date represented as an integer number of "days" since the Library object was created

an init method that initializes the current\_date to zero

add\_library\_item - takes a LibraryItem object as a parameter and adds it to the holdings

add\_patron - takes a Patron object as a parameter and adds it to the members

lookup\_library\_item\_from\_id - returns the LibraryItem object corresponding to the ID parameter, or None if no such LibraryItem is in the holdings

lookup\_patron\_from\_id - returns the Patron object corresponding to the ID parameter, or None if no such Patron is a member

check\_out\_library\_item

takes as parameters a patron ID and a library item ID, in that order

if the specified Patron is not in the Library's members, return "patron not found"

if the specified LibraryItem is not in the Library's holdings, return "item not found"

if the specified LibraryItem is already checked out, return "item already checked out"

if the specified LibraryItem is on hold by another Patron, return "item on hold by other patron"

otherwise update the LibraryItem's checked\_out\_by, date\_checked\_out and location

if the LibraryItem was on hold for this Patron, update requested\_by

update the Patron's checked\_out\_items

return "check out successful"

return\_library\_item

takes as its parameter a library item ID

if the specified LibraryItem is not in the Library's holdings, return "item not found"

if the LibraryItem is not checked out, return "item already in library"

update the Patron's checked\_out\_items

update the LibraryItem's location depending on whether another Patron has requested it (if so, it should go on the hold shelf)

update the LibraryItem's checked\_out\_by

return "return successful"

request\_library\_item

takes as parameters a patron ID and a library item ID, in that order

if the specified Patron is not in the Library's members, return "patron not found"

if the specified LibraryItem is not in the Library's holdings, return "item not found"

if the specified LibraryItem is already requested, return "item already on hold"

update the LibraryItem's requested\_by

if the LibraryItem is on the shelf, update its location to on hold

return "request successful"

pay\_fine

takes as parameters a Patron ID and the amount (in dollars) being paid (in that order)

if the specified Patron is not in the Library's members, return "patron not found"

use amend\_fine to update the Patron's fine; return "payment successful"

increment\_current\_date

takes no parameters

increment current date

increase each Patron's fines by 10 cents for each overdue LibraryItem they have checked out (by calling amend\_fine)

Note - a LibraryItem can be on request without its location being the hold shelf (if another Patron has it checked out);

One limited example of how your classes might be used is:

b1 = Book("345", "Phantom Tollbooth", "Juster")

a1 = Album("456", "...And His Orchestra", "The Fastbacks")

m1 = Movie("567", "Laputa", "Miyazaki")

print(b1.get\_author())

print(a1.get\_artist())

print(m1.get\_director())

p1 = Patron("abc", "Felicity")

p2 = Patron("bcd", "Waldo")

lib = Library()

lib.add\_library\_item(b1)

lib.add\_library\_item(a1)

lib.add\_patron(p1)

lib.add\_patron(p2)

lib.check\_out\_library\_item("bcd", "456")

loc = a1.get\_location()

lib.request\_library\_item("abc", "456")

for i in range(57):

lib.increment\_current\_date() # 57 days pass

p2\_fine = p2.get\_fine\_amount()

lib.pay\_fine("bcd", p2\_fine)

lib.return\_library\_item("456")

You are responsible for testing all of the required functions to make sure they operate as specified.

Your file must be named: Library.py

Just to think about: Since there are three possible locations for a LibraryItem, there are six hypothetical changes in location. Are all six possible according to these specifications?