CSE 1321L: Programming and Problem Solving I Lab

Assignment 7 Fall 2025

Review

What students will learn

- Problem Solving.
- Basic Program Structure.
- Input and Output with the user.
- Combine loops, conditionals, and functions into cohesive programs
- Apply exception handling for robust user interaction.

Content

Overview:

For this assignment, you're going to practice making logic in your code. It will include all modules covered in this course. In practical terms, this means you're going to expand on the concepts from previous assignments but also include things like classes, exception handling. Again, start early, practice, and ask a lot of questions.

Final note: *Do not cheat*

If your temptation is to look online, don't. Come see us instead and ask questions – we are here to help. Remember, you are going to have to write codes in your future job interviews, so learn it now to secure a high-paying job later.

Assignment7A: Owl Cafe (Order Management System)

You are hired to help automate order management for a small cafe. The cafe sells the following items:

- Coffee – $3.50
- Sandwich – $5.75
- Smoothie – $4.25
- Muffin – $2.75

Requirements:

- Display a main menu where the user can:
    1. Place an Order
    2. View Daily Sales
    3. Exit
- When choosing Place Order, show the product list and allow the user to:
    - Add multiple items in one order session.
    - After each item, ask if they want to add another (Y/N).
    - Use try–except to handle invalid input (non-numeric or out-of-range values).
    - Save each order (item name, quantity, and cost) into a list.
- Option 2 displays:
    - A summary of all items ordered that day.
    - The total revenue earned.
- Option 3 exists the program.
- Write functions for (display_menu(), take_order(), show_sales())

Example runs are shown below. The user input is shown in **red**.

Sample Output #1:

```
[Owl Cafe]
1. Place Order
2. View Daily Sales
3. Exit
> 1

Menu Items:
1. Coffee - $3.50
2. Sandwich - $5.75
3. Smoothie - $4.25
4. Muffin - $2.75
Select an item (1-4): 8
Error: Please enter a number between 1 and 4.

Menu Items:
1. Coffee - $3.50
2. Sandwich - $5.75
3. Smoothie - $4.25
4. Muffin - $2.75
Select an item (1-4): 1
Enter quantity: 5
Added 5 Coffee(s) - $17.50
Add another item? (Y/N): y
```

```
Menu Items:
1. Coffee - $3.50
2. Sandwich - $5.75
3. Smoothie - $4.25
4. Muffin - $2.75
Select an item (1-4): 2
Enter quantity: 4
Added 4 Sandwich(s) - $23.00
Add another item? (Y/N): n

Order complete! Total for this order: $40.50

[Owl Cafe]
1. Place Order
2. View Daily Sales
3. Exit
> 1

Menu Items:
1. Coffee - $3.50
2. Sandwich - $5.75
3. Smoothie - $4.25
4. Muffin - $2.75
Select an item (1-4): 4
Enter quantity: 3
Added 3 Muffin(s) - $8.25
Add another item? (Y/N): n

Order complete! Total for this order: $8.25

[Owl Cafe]
1. Place Order
2. View Daily Sales
3. Exit
> 2

Today's Sales:
- 5 Coffee(s): $17.50
- 4 Sandwich(s): $23.00
- 3 Muffin(s): $8.25
Total Sales: $48.75

[Owl Cafe]
1. Place Order
2. View Daily Sales
3. Exit
> 3
Thank you for visiting Owl Cafe!
```

Assignment7B: Owl Library

You are going to develop an object-oriented based library management system for *Owl Library*. The system should allow users to **view, borrow, return, and add books** through a menu-driven interface.

Your program will include two classes:

1. **Book** – represents an individual book
2. **Library** – manages all books and handles user interactions

The main program will create a Library object and display a main menu that lets users manage the collection.

Class 1: Book

Attributes:

- title → the name of the book
- is_borrowed → Boolean (default = False)

Methods:

- __init__(self, title)
  Initializes the title and sets is_borrowed to False.
- borrow(self)
  - If available, mark as borrowed and print a confirmation; otherwise, display an error message.
- return_book(self)
  - If currently borrowed, mark as available and print confirmation; otherwise, display an error message.
- display_info(self, index)
  - Prints the book index, title, and status ("Available" / "Borrowed").

Class 2: Library

Attributes:

- books → a list of Book objects.

Methods:

- __init__(self)
  Initializes the library with the following default books:
  - Python Basics, Intro to AI
- show_books(self)
  - Displays all books with their index and current status.
- borrow_book(self)
  - Displays available books.
  - Prompts the user for the index.
  - Uses try–except to catch invalid input or out-of-range values.
  - Calls the borrow() method of the selected book.
- return_book(self)
  - Displays all books.
  - Prompts for index.
  - Uses try–except to handle invalid input.
  - Calls the return_book() method of the selected book.
- add_book(self)
  - Prompts the user to enter a book title.
  - Creates a new Book object and adds it to the books list.
  - Prints a confirmation message.

Program Flow

1. Create a Library object.
2. Continuously show the main menu.
3. Perform actions based on user input.
4. Use exception handling to validate all inputs.
5. Stop when the user chooses to exit.

Example runs are shown below. The user input is shown in **red**.

Sample Output #1:

```
[Owl Library]
1. View Books
2. Borrow Book
3. Return Book
4. Add Book
5. Exit
> 1

Books in Library:
0 - Python Basics (Available)
1 - Intro to AI (Available)

[Owl Library]
1. View Books
2. Borrow Book
3. Return Book
4. Add Book
5. Exit
> 4
Enter the title of the new book: Game Design 101
"Game Design 101" has been added to the library!

[Owl Library]
1. View Books
2. Borrow Book
3. Return Book
4. Add Book
5. Exit
> 2

Books in Library:
0 - Python Basics (Available)
1 - Intro to AI (Available)
2 - Game Design 101 (Available)
Select a book to borrow: 1
You borrowed "Intro to AI".

[Owl Library]
1. View Books
2. Borrow Book
3. Return Book
4. Add Book
5. Exit
```

```
> 3

Books in Library:
0 - Python Basics (Available)
1 - Intro to AI (Borrowed)
2 - Game Design 101 (Available)
Select a book to return: 2
Error: "Game Design 101" was not borrowed.

[Owl Library]
1. View Books
2. Borrow Book
3. Return Book
4. Add Book
5. Exit
> 3

Books in Library:
0 - Python Basics (Available)
1 - Intro to AI (Borrowed)
2 - Game Design 101 (Available)
Select a book to return: 1
You returned "Intro to AI".

[Owl Library]
1. View Books
2. Borrow Book
3. Return Book
4. Add Book
5. Exit
> 5
Exiting program... Goodbye!
```

Assignment7C: Score Analyzer

You are asked to write a program that helps a teacher analyze student test scores using **loops**, **selection statements**, and **functions**.
The program will prompt the teacher to enter the number of students and each student's score.
It should then compute and display:

- The **highest score**
- The **lowest score**
- The **average score**
- How many students **passed** (score ≥ 60)
- How many **failed** (score < 60)

Requirements:

- Implement **three functions**:
  - get_scores()
    - asks the user for the number of students, loops to collect scores, and returns a list of integers.
  - analyze_scores(scores)
    - takes the list of scores and calculates:
      - highest, lowest, average, passes, and fails.
      - returns these values as a tuple.
  - display_results(high, low, avg, passed, failed)
      - prints the results according to the format.
- In main(), call these functions appropriately.
- Use **loops** and **if/else** for logic.

Example runs are shown below. The user input is shown in **red**.

Sample Output #1:

```
Enter number of students: 5
Enter score for student 1: 78
Enter score for student 2: 45
Enter score for student 3: 92
Enter score for student 4: 60
Enter score for student 5: 79

--- Score Report ---
Highest score: 92
Lowest score: 45
Average score: 70.8
Students passed: 4
Students failed: 1
```

Sample Output #2:

```
Enter number of students: 6
Enter score for student 1: 35
Enter score for student 2: 85
Enter score for student 3: 92
Enter score for student 4: 40
Enter score for student 5: 60
Enter score for student 6: 75

--- Score Report ---
Highest score: 92
Lowest score: 35
Average score: 64.5
Students passed: 4
Students failed: 2
```

Submission Instructions:

- Programs must follow the output format provided. This includes each blank line, colons (:), and other symbols.
- Programs must be working correctly.
- Programs must be written in Python.
- Programs must be submitted with the correct **.py** format.
- Programs must be saved in files with the correct file name:
  - Assignment7A.py
  - Assignment7B.py
  - Assignment7C.py
- Programs (source code files) must be uploaded to Gradescope by the due date.