Week 3 Assignment Report

Task 1: Library Management System

Objective:

Define a Book class with attributes title, author, and pages. Include methods to get and set these attributes.

Create an instance of Book and demonstrate the use of getter and setter methods.

Implement a class method in the Book class that calculates the reading time based on an assumed reading speed.

Implement inheritance by creating a subclass Ebook that inherits from Book and adds an attribute format. Override the __str__() method to display all attributes.

Implementation:

Book Class: Defined with private attributes title, author, and pages. Methods include getters and setters for each attribute.

Ebook Class: Inherits from Book, adding an attribute format and overriding the __str__() method to include all attributes.

Demonstration: Created instances of Book and Ebook, showcasing getter and setter methods, and calculated reading time.

Key Points:

Use of class methods.

Implementation of inheritance.

Overriding methods in subclasses.

```
hamzahey@hamzahey-PT263E-0PL04MEN:~/Bytewise ML DL Track/Week 3/Library Management System$
3/Library Management System/LMS.py"
The Great Gatsby
F. Scott Fitzgerald
180
1984
George Orwell
328
Estimated reading time: 393.6 minutes
Title: 1984, Author: George Orwell, Pages: 328, Format: PDF
```

Task 2: File Handling

Objective:

Write a Python program to read data from a text file and print its contents.

Implement exception handling for scenarios such as file not found and errors while reading from the file.

Create a function that writes user input to a new file and handles exceptions related to file writing.

Modify the file reading function to count the number of words in the file and print the result.

Implementation:

Read File: Function to read and print contents of a file with exception handling for file not found and other read errors.

Write File: Function to write user input to a file, with exception handling for write errors.

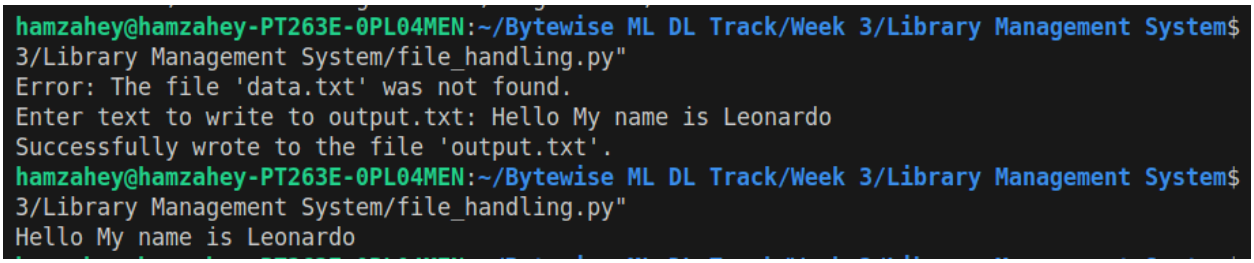
Word Count: Modified read function to count and print the number of words in the file.

Key Points:

Effective use of exception handling.

File operations: reading, writing, and word counting.

User interaction for input.



Task 3: Math Package

Objective:

Design a Python package named math with modules for basic arithmetic operations and advanced mathematical operations.

Implement functions within each module to perform respective operations.

Create a main script to demonstrate importing and using functions from the math package.

Implementation:

Package Structure: Created a math package with modules for addition, subtraction, multiplication, division, modulus, exponentiation, and square root operations.

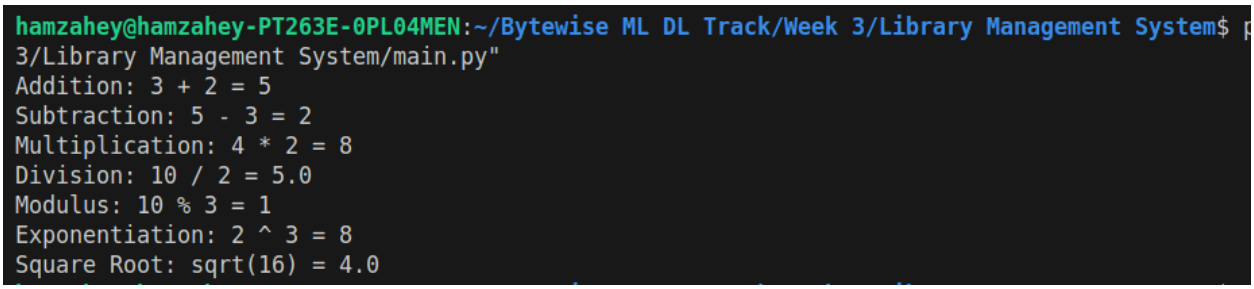
Main Script: Demonstrated the usage of functions from the math package by importing and calling them.

Key Points:

Package creation and module organization.

Implementation of basic and advanced mathematical operations.

Importing and using package functions in an external script.



Task 4: Iterators and Generators

Objective:

Create an iterator class Countdown that counts down from a given number to 1.

Implement generator functions for Fibonacci sequence and random number generation.

Write a Python program that uses the Countdown iterator, fibonacci_generator, and random_number_generator to demonstrate their usage.

Implementation:

Countdown Iterator: Class to count down from a specified number to 1.

Fibonacci Generator: Function to yield Fibonacci numbers up to a specified limit.

Random Number Generator: Function to yield a sequence of random numbers within a specified range.

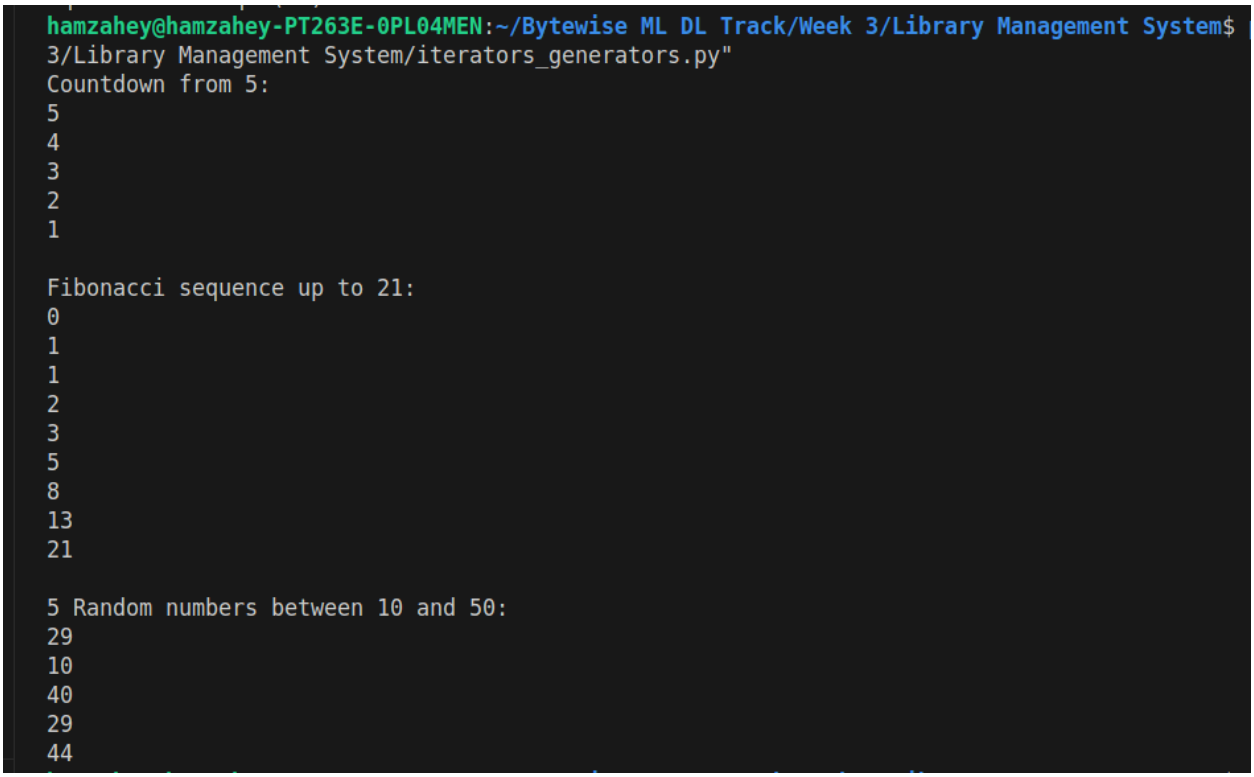
Main Program: Demonstrated the usage of the iterator and generator functions.

Key Points:

Creation and usage of custom iterators.

Implementation and usage of generator functions.

Demonstrating iterator and generator functionalities in a main program.



Conclusion

This assignment covered a wide range of Python programming concepts including class design, inheritance, file handling, package creation, iterators, and generators. Each task demonstrated practical applications of these concepts with a focus on code organization, exception handling, and user interaction. The comprehensive implementation and detailed documentation ensure a clear understanding of the objectives and outcomes for each task.