Class Activity: Student Grade Averager

Objective:

Students will learn how to read from and write to files in Python, manipulate data, and handle exceptions. They will also practice using dictionaries and lists to store and process data.

Activity Description:

Students will create a program that reads student data from a source file, calculates the average score for each student, and writes the results to a destination file. The source file will contain student names and their scores, and the destination file will store the student names along with their average scores.

Instructions:

Prepare the Source File:

Create a text file named student_scores.txt with the following content:

Name Score1 Score2 Score3
Alice 85 90 88
Bob 78 82 80
Charlie 92 95 89

Task for Students:

Write a Python program that:
- Prompts the user to enter the source file name (e.g., student_scores.txt).
- Prompts the user to enter the destination file name (e.g., student_averages.txt).
- Reads the source file line by line, skipping the header.
- For each student, calculates the average of their scores.
- Stores the student's name and their average score in a dictionary.
- Writes the student names and their average scores to the destination file.
- Handles the FileNotFoundError exception if the source file does not exist.
- Ensures that both files are properly closed after operations.

Expected Output:

Class Activity: Student Grade Averager

If the source file is student_scores.txt and the destination file is student_averages.txt, the destination file should contain:

Alice 87.66666666666667
Bob 80.0
Charlie 92.0

Discussion Questions:

- What happens if the source file contains non-numeric values in the score columns?
- How would you modify the program to handle missing scores for some students?
- What are the benefits of using a dictionary to store the results?

Extension (Optional):

- Modify the program to round the average scores to two decimal places.
- Add functionality to sort the students by their average scores before writing to the destination file.

Learning Outcomes:

Students will gain experience with file I/O operations in Python.
They will understand how to handle exceptions and ensure proper resource management.
They will practice using dictionaries and lists for data manipulation.
They will develop problem-solving skills by addressing edge cases and extending the program's functionality.
This activity encourages hands-on learning and critical thinking while reinforcing key programming concepts.