Python Exceptions and File Handling Lab

Lab Overview

This lab will cover the following topics:

1. Reading from a file.
2. Writing to a file.
3. Handling exceptions effectively.
4. Storing data in JSON format.
5. Validating user inputs.

Problem 1: File Reader with Exception Handling

Write a Python program that:

1. Asks the user for a file name.
2. Reads the contents of the file.
3. Handles the following exception:
   - `FileNotFoundError` if the file does not exist.
4. If the file is read successfully, print its contents.
5. Include an option for the user to try again if an error occurs.

Example Input:

```
Enter file name: example.txt
```

Example Output:

```
Contents of example.txt:
Hello, this is a sample file.
```

Error Example:

```
Enter file name: missing.txt
Error: File 'missing.txt' not found.
Would you like to try again? (yes/no): yes
```

Problem 2: Write Data to a File

Write a Python program that:

1. Asks the user for a file name.
2. Prompts the user to enter some text.
3. Writes the entered text to the specified file.
4. Handles the following exceptions:
    - `PermissionError` if the file cannot be written to.
5. Confirm successful write by reopening the file and displaying its contents.

Example Input:

```
Enter file name: notes.txt
Enter text to save: This is a note.
```

Example Output:

```
Data saved successfully to notes.txt.
Contents of notes.txt:
This is a note.
```

Error Example:

```
Enter file name: /restricted/notes.txt
Error: Permission denied to write to '/restricted/notes.txt'.
```

Problem 3: Store Data in JSON Format

Write a Python program that:

1. Creates a dictionary with the following structure:

```
2. data = {
3.     "name": "Alice",
4.     "age": 25,
5.     "city": "New York"
6. }
```

7. Saves this dictionary into a file named `data.json` in JSON format.
8. Handles the following exceptions:
    - `IOError` if there is an error writing to the file.
9. Reloads the JSON file and displays its contents to confirm successful saving.

Example Output:

```
Data saved successfully to data.json.
Contents of data.json:
{"name": "Alice", "age": 25, "city": "New York"}
```

Problem 4: Complete Workflow

Write a program that combines all the above steps:

1. Reads a file and prints its contents.
2. Writes user input to a new file.
3. Saves data in JSON format.
4. Validates all user inputs to ensure correctness.
5. Ensures all exceptions are handled gracefully.
6. Provides a menu system to navigate between the tasks.

Example Input/Output:

```
Main Menu:
1. Read a file
2. Write to a file
3. Save data to JSON
4. Exit
Choose an option: 1

Step 1: Enter file name to read: data.txt
Error: File 'data.txt' not found.
Would you like to try again? (yes/no): no

Main Menu:
1. Read a file
2. Write to a file
3. Save data to JSON
4. Exit
Choose an option: 2

Enter file name to write: output.txt
Enter text to save: Hello, world!
Data saved successfully to output.txt.
Contents of output.txt:
Hello, world!

Main Menu:
1. Read a file
2. Write to a file
3. Save data to JSON
4. Exit
Choose an option: 4
Goodbye!
```

Problem 5: Validate User Inputs

Write a Python program that:

1. Prompts the user to enter their name, age, and email.
2. Validates the inputs as follows:
    - Name must only contain alphabetic characters and spaces.
    - Age must be a positive integer.
    - Email must follow a standard email format (e.g., contain "@" and ".").
3. Handles the following exceptions:
    - `ValueError` if the age is not an integer.
4. Outputs the validated data.

Example Input:

```
Enter your name: John Doe
Enter your age: 25
Enter your email: john.doe@example.com
```

Example Output:

```
Validated Data:
Name: John Doe
Age: 25
Email: john.doe@example.com
```

Error Example:

```
Enter your age: twenty-five
Error: Age must be a valid integer.
```

Bonus Task

Enhance Problem 3 by allowing the user to input their own data (e.g., name, age, and city) instead of using hardcoded values. Include validation to ensure:

1. Name is a string.
2. Age is a positive integer.
3. City is a non-empty string.

Example Input:

```
Enter your name: Bob
Enter your age: 30
Enter your city: Cairo
```

Example Output:

```
Data saved successfully to data.json.
Contents of data.json:
{"name": "Bob", "age": 30, "city": "Cairo"}
```

Instructions for Submission

- Submit the Python script(s) for each problem.
- Ensure the scripts include comments explaining the exception handling.
