Python File Handling – Beginner's Notes

1. Introduction to File Handling

- A file is like a digital notebook where you can store information permanently.
- In Python, you can open, read, write, and close files.
- File handling is essential for saving user data, logs, reports, and more.

2. Why File Handling Matters

- Variables in Python only exist while the program is running. Once the program ends, the data is lost.
- Files help us **persist data** even after the program stops.
- Example: Saving a game progress, storing a to-do list, or keeping student grades.

3. Opening Files

- Python provides the `open()` function.
- Syntax: `open(filename, mode)`
- Common modes:
- `'r'` → Read (file must exist)
- `'w'` → Write (creates file if it doesn't exist, erases contents if it does)
- `'a'` → Append (adds new data without erasing old data)
- `'b'` → Binary mode (for images, audio, etc.)

4. Reading from Files

- `read()` → Reads the entire file as one string.
- `readline()` → Reads one line at a time.
- `readlines()` → Reads all lines into a list.

Example: Reading names from a file line by line.

5. Writing to Files

- Using `'w'` mode overwrites the file.
- Using `'a'` mode adds to the file.
- `write()` → Writes a string.
- `writelines()` → Writes multiple lines from a list.

Example: Writing student names into a file.

6. Closing Files

- Always close files after use with `close()`.

- Alternatively, use the `with` statement:

```
with open("file.txt", "r") as f:
    data = f.read()
```

- Advantage: Python automatically closes the file.

7. File Paths

- Relative path: Refers to the file in the same folder as your script.
- Absolute path: Full address of the file on your computer.

8. Exception Handling with Files

- Sometimes files don’t exist, or permissions are restricted.
- Use `try-except` to handle errors.
- Example:

```
try:
    with open("grades.txt", "r") as f:
        print(f.read())
except FileNotFoundError:
    print("File not found!")
```

9. Custom Exceptions

- You can create your own exceptions by making a class.
- Example:

```
class EmptyFileError(Exception):
    pass
```

- This allows you to raise meaningful errors in your program.

10. Real-Life Use Cases

- Saving user progress in games.
- Storing logs for debugging.
- Writing reports automatically.
- Reading configuration files.

11. Best Practices

- Always close files (or use `with`).

- Handle exceptions gracefully.
- Use the right file mode.
- Don't overwrite files accidentally – prefer append when possible.

Practice Exercises

Exercise 1: Reading Safely

Write a program that tries to open a file `students.txt` and prints its contents. If the file doesn't exist, show a friendly error message.

Exercise 2: Writing and Reading

Write a program that asks the user for their favorite movies and saves them to a file. Then read the file back and display the list.

Exercise 3: Append Mode

Write a diary program where every time it runs, it asks the user for an entry and appends it to `diary.txt`.

Exercise 4: Exception Handling

Write a program that asks for a filename from the user and tries to open it. If it doesn't exist, print "Oops, no such file!"

Exercise 5: Custom Exception

Create a custom exception called `EmptyFileError`. If a user tries to read a file that is empty, raise this exception.

Exercise 6: Witty Challenge

Make a program called `forgive_me.py` that checks if the file `apology.txt` exists. If not, ask the user to write an apology message and save it into the file.