1. File Operations in Python 🚪🔑

**Objectives Of the Topic 🌟**

* **Master Files:** Get hands-on with opening, reading, writing, and closing files in Python!
* **Handle Errors Like a Pro:** Learn how to catch and manage errors, ensuring your code runs smoothly even when things don’t go as planned.

**What is File Handling? 📁**

**File Handling** in Python is the ability to perform various operations on files, like reading from and writing to them. Files are used to store data permanently (like text documents, images, or spreadsheets). Unlike variables, which lose their values when a program ends, files provide **persistent storage**.

File handling in Python allows you to:

1. **Open files** in different modes (e.g., read-only or write mode).
2. **Read and write data** in a variety of formats.
3. **Close files** to free up system resources.

Python’s built-in open() function is at the heart of this process, and **learning how to handle files** can make your programs more robust, flexible, and useful.

**File Operations in Python 🚪🔑 (1.5 hours)**

Let’s start with the basics of file handling!

* **Opening Files** 🔓
  + Use Python’s open() function to access a file.
  + Syntax: open(filename, mode), where:
    - filename: The name of the file you want to work with.
    - mode: The mode you want to open the file in.
  + Modes include:
    - 'r': Read mode, used for reading files.
    - 'w': Write mode, creates a new file or overwrites an existing one.
    - 'a': Append mode, adds new content without deleting existing data.
    - 'rb', 'wb': Binary modes for non-text files, like images.

**Example:**

file = open("example.txt", "r") # Opens the file in read mode

* **Reading Files** 📜
  + Python provides multiple ways to read file contents:
    - .read(): Reads the entire file.
    - .readline(): Reads a single line at a time.
    - .readlines(): Reads all lines and returns a list.
  + **Example:**

with open("example.txt", "r") as file: data = file.read() print(data)

* + Use cases: **Processing large datasets** or analyzing **text documents.**
* **Writing & Appending to Files** ✍️
  + Writing is essential for **saving data**, like storing a user’s progress or keeping a record.
  + write(): Overwrites content, while append() allowing adding without deleting.
  + **Example:**

with open("output.txt", "w") as file: file.write("Hello, Python!")

* **Closing Files** 🚪
  + Files should be closed after processing to release system resources.
  + Python’s with statement automatically handles closing, ensuring efficient resource management.

1. Exception Handling 🌪️

**Exception Handling 🌪️ (1.5 hours)**

Errors happen! To make sure your programs are **error-proof** and user-friendly, Python provides **Exception Handling**. It’s the art of catching errors and handling them gracefully.

* **Basic Structure of try-except Blocks** ⚙️
  + try: Runs code that might throw an error.
  + except: Catches the error, allowing you to respond without crashing.
  + **Example**

try:

with open("nonexistent.txt", "r") as file:

data = file.read()

except FileNotFoundError:

print("File not found. Please check the filename.")

**Advanced Error Handling with finally and Custom Errors** 🎩

* **finally**: Runs no matter what, often used to **clean up** (like closing a file).
* **Custom Errors**: Create custom exceptions for special cases (e.g., EmptyFileError).

**Example with finally**:

try:

file = open("sample.txt", "r")

data = file.read()

except FileNotFoundError:

print("File not found.")

finally:

file.close()

**Best Practices 📏**

* **Use with for file handling**: Auto-close files, preventing potential leaks.
* **Check file existence** before reading/writing, to avoid crashes.
* **Handle specific exceptions** over general ones (e.g., FileNotFoundError instead of Exception).
* **Document error messages** clearly for easier debugging and user support.

**File Handling and Exception Handling Recording**

<https://powerlearnproject-org.zoom.us/rec/share/DwBrnDbsm_kzKe57Dv9yJ9X-HU3XotSdzMSL0uJuBVqLi6eNGkudLdJ2TVh8dcnM.OG-1C2MrLq9NE89s>

**Repository For reference**

<https://github.com/PLPAfrica/August-Python.git>