**Bangladesh-Bharot Digital Service for Employment & Training (BDSET)**

**Module: Python Programming**

**Lab 06  
Problem Solving using Basics of Python for Artificial Intelligence**

**Basic I/O Functions**

**Learning Objectives**

By the end of this lab session, students will be able to:  
✅ Open and close files in Python  
✅ Understand different file access modes  
✅ Create, read, update, and delete files

### ****1. Opening and Closing Files****

Python provides the open() function to work with files.  
**Syntax:**

file = open("filename.txt", "mode")

|  |  |
| --- | --- |
| **Mode** | **Description** |
| "r" | Read mode (default) – file must exist |
| "w" | Write mode – creates a new file or overwrites existing content |
| "a" | Append mode – adds new data at the end of the file |
| "r+" | Read and Write – file must exist |
| "w+" | Write and Read – creates/overwrites file |
| "a+" | Append and Read – creates file if it doesn’t exist |

### ****📌 2. Closing a File****

Closing a file ensures that changes are saved and resources are freed.

file.close()

**Why is Closing a File Important in Python?**

When working with files in Python, **closing the file** is crucial for several reasons:

**Prevents Data Loss & Corruption**

* If you don’t close a file after writing, data might not be saved properly.
* Python **buffers** file operations, meaning changes might be temporarily stored in memory before being written to disk.

⚡ **Best Practice: Using with Statement**  
Python automatically closes the file when with is used.

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

content = file.read()

print(content) # No need to close the file

### ****Exercise 1: Creating & Writing to a File****

Create a file and write data into it.

with open("example.txt", "w") as file:

file.write("Hello, this is a test file.\n")

file.write("We are learning Python file handling.")

### ****Exercise 2: Reading from a File****

Read the file we just created.

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

content = file.read()

print("File Contents:\n", content)

✅ **Task:** Modify the script to read only the first 10 characters.

### ****Exercise 3: Appending Data to a File****

Instead of overwriting, we can add new content using "a" mode.

with open("example.txt", "a") as file:

file.write("\nAppending new information.")

✅ **Task:** Open the file and verify the new content.

### ****Exercise 4: Reading a File Line by Line****

Instead of reading the entire content, read line by line:

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

for line in file:

print(line.strip()) # Removes extra newline characters

✅ **Task:** Modify the script to print only lines that contain the word “Python”.

### ****Exercise 5: Using Different File Modes****

Let’s create a file and use **r+, w+, and a+** modes.

# Using r+ mode (Read & Write)

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

content = file.read()

print("Before writing:", content)

file.write("\nUpdated content!")

✅ **Task:** Explain why "r+" doesn’t overwrite the file but starts writing at the last position.

**Summary Table: Comparing r+, w+, and a+ Modes in Python**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Mode | Read (file.read()) | Write (file.write()) | Overwrites File? | Appends to End? | Cursor Position on Open | Creates File if Not Exists? |
| r+ | ✅ Yes | ✅ Yes | ❌ No (Edits in place) | ❌ No | Start of the file | ❌ No (Raises error if file is missing) |
| w+ | ✅ Yes | ✅ Yes | ✅ Yes (Clears the file) | ❌ No | Start of the file | ✅ Yes |
| a+ | ✅ Yes | ✅ Yes | ❌ No | ✅ Yes (Always writes at the end) | End of the file | ✅ Yes |

**🔍 Explanations**

1. **r+ (Read & Write)**
   * Does **not delete existing content**.
   * **Starts writing from the beginning**, replacing characters if necessary.
   * If you need to read first, use seek(0) before read().
   * **File must exist**, otherwise, it raises an error.
2. **w+ (Write & Read)**
   * **Overwrites (clears) the file** every time it’s opened.
   * Allows both reading and writing.
   * If the file **does not exist**, it creates a new one.
3. **a+ (Append & Read)**
   * **Does not erase existing content**.
   * Always **appends to the end** of the file.
   * Reading requires seek(0) to view the full content.
   * **Creates a file if it doesn’t exist**.

**📌 Choosing the Right Mode**

| **Use Case** | **Best Mode** |
| --- | --- |
| Read & modify an existing file without deleting content | r+ |
| Completely overwrite and start fresh | w+ |
| Append new data while keeping the existing content | a+ |

Would you like some practical coding tasks to test these modes? 🚀😊

### ****Exercise 6: Deleting and Checking if a File Exists in Google Colab****

In Google Colab, you can use the os and pathlib modules to check if a file exists and delete it.

## **🔹 Method 1: Using** os **Module**

import os

file\_path = "/content/sample.txt"

# Check if file exists

if os.path.exists(file\_path):

print("File exists. Deleting now...")

os.remove(file\_path) # Delete the file

else:

print("File does not exist.")

## **🔹 Method 2: Using** pathlib **Module**

from pathlib import Path

file\_path = Path("/content/sample.txt")

# Check if file exists

if file\_path.exists():

print("File exists. Deleting now...")

file\_path.unlink() # Delete the file

else:

print("File does not exist.")

## **📝 Explanation**

* **os.path.exists(file\_path)** → Checks if the file exists.
* **os.remove(file\_path)** → Deletes the file if found.
* **Path(file\_path).exists()** → Another way to check file existence using pathlib.
* **file\_path.unlink()** → Deletes the file using pathlib.