2. Handout

File Handling in Python: Student Handout

Welcome to the session on **File Handling in Python**! This guide will help you understand how to work with files in Python, including reading, writing, and appending data. Let's dive into the key concepts and examples.

---

1. What is File Handling?

File handling involves opening, reading, writing, and closing files in a program. Files can store various types of data, such as text or numbers.

---

2. Opening a File

To work with a file, you must first open it using the `open()` function.

Syntax:

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

File Modes:

- `"r"` : Read mode
- `"w"` : Write mode
- `"a"` : Append mode
- `"r+"` : Read and write mode

Examples:

1. **Open a file for reading:**

   ```
   file = open("example.txt", "r")
   ```

2. **Open a file for writing:**

```
file = open("example.txt", "w")
```

3. **Open a file for appending:**

```
file = open("example.txt", "a")
```

---

3. Reading from a File

Once a file is opened in read mode, you can read its content.

Examples:

1. **Read the entire file:**

```
file = open("example.txt", "r")
content = file.read()
print(content)
file.close()
```

2. **Read line by line:**

```
file = open("example.txt", "r")
for line in file:
    print(line)
file.close()
```

3. **Read specific number of characters:**

```
file = open("example.txt", "r")
content = file.read(10)  # Reads the first 10 characters
print(content)
file.close()
```

---

4. Writing to a File

To write data to a file, open it in write (`"w"`) or append (`"a"`) mode.

Examples:

1. **Write to a file (overwrite):**

```
file = open("example.txt", "w")
file.write("Hello, World!")
file.close()
```

2. **Append to a file:**

```
file = open("example.txt", "a")
file.write("\nThis is a new line.")
file.close()
```

3. **Write multiple lines:**

```
file = open("example.txt", "w")
lines = ["Line 1\n", "Line 2\n", "Line 3\n"]
file.writelines(lines)
file.close()
```

---

5. Closing a File

Always close a file after use to free up system resources.

Example:

```
file.close()
```

---

6. Working with Different File Modes

Summary of File Modes:

Mode	Description
`"r"`	Read mode (default). Opens the file for reading.
`"w"`	Write mode. Opens the file for writing (overwrites existing content).
`"a"`	Append mode. Opens the file for appending new data.
`"r+"`	Read and write mode. Opens the file for both reading and writing.

---

7. Writing Programs to Read and Process Data from Files

Example: Calculate the sum of numbers in a file

```
file = open("data.txt", "r")
total = 0

for line in file:
    total += int(line.strip())  # Convert each line to an integer and add
to total

file.close()
print("The sum of the numbers is:", total)
```

---

8. Understanding Python Modules and Libraries

Python modules and libraries provide additional functionality.

Popular Libraries:

- **math**: Mathematical functions
- **datetime**: Date and time functions
- **os**: Operating system interactions
- **sys**: System-specific parameters and functions

---

9. Activity: Using a Python Library and Handling Errors

Example: Check if a file exists and handle errors

```python
import os

filename = "example.txt"

try:
    if os.path.exists(filename):
        file = open(filename, "r")
        print(file.read())
        file.close()
    else:
        print(f"The file {filename} does not exist.")
except Exception as e:
    print(f"An error occurred: {e}")
```

Conclusion

In this session, you learned about file handling in Python, including:

- Opening files in different modes
- Reading from and writing to files
- Closing files
- Using Python libraries and handling errors

Key Takeaways:

- Always close a file after use.
- Choose the appropriate file mode for your task.
- Handle errors using `try-except` blocks.

This handout should take approximately **10-15 minutes** to read. Feel free to practice the examples and ask questions if needed!