DATA SCIENCE COURSE TUTORIAL # 33

3.20 File Handling

What is File Handling?

File handling in Python allows programs to read, write, and manipulate files stored on the computer. It helps in storing data permanently instead of temporarily keeping it in memory. Files are important for saving logs, user data, or program outputs.

Types of File Operations

File handling mainly involves four basic operations:

1. **Open a file**
2. **Read or write data**
3. **Close the file**
4. **Handle file exceptions**

Opening a File

Files are opened using the **open()** function in Python.

Syntax:

```
open(filename, mode)
```

Parameters:

- **filename:** The name of the file (with path if needed).
- **mode:** Defines the purpose for opening the file.

Mode	Description
'r'	Read mode (default). Opens file for reading.
'w'	Write mode. Creates new file or overwrites existing one.
'a'	Append mode. Adds new data to the end of the file.
'r+'	Read and write mode.
'w+'	Write and read mode (overwrites).
'a+'	Append and read mode.

Example:

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

Reading from a File

Python provides multiple methods to read data from files.

read() Method

Reads the entire content of a file as a single string.

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

readline() Method

Reads only one line from the file.

```
file = open('example.txt', 'r')
line = file.readline()
print(line)
file.close()
```

readlines() Method

Reads all lines and returns a list.

```
file = open('example.txt', 'r')
lines = file.readlines()
print(lines)
file.close()
```

Writing to a File

To write data, open the file in **'w'** (write) or **'a'** (append) mode.

write() Method

Writes a single string to the file.

```
file = open('output.txt', 'w')
file.write('Hello, this is Python file handling!')
file.close()
```

writelines() Method

Writes multiple lines to a file.

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

Appending to a File

If you want to add new data without deleting existing content, use **append mode ('a')**.

```
file = open('output.txt', 'a')
file.write('\nNew line added at the end.')
file.close()
```

Closing a File

Always close a file after use to free system resources.

```
file.close()
```

It is a good practice to use the **with** statement to automatically close the file.

Example:

```
with open('example.txt', 'r') as file:
    content = file.read()
    print(content)
```

Explanation:

- The file automatically closes after the **with** block finishes execution.

Checking if File Exists

You can check if a file exists before opening it to avoid errors.

```
import os
if os.path.exists('example.txt'):
    print('File found!')
else:
    print('File not found!')
```

Deleting a File

You can delete a file using the **remove()** function from the **os** module.

```
import os
os.remove('output.txt')
```

Note: Always check if the file exists before deleting.

Example: File Write and Read Together

```
# Writing data
data = 'This is a sample text.'
with open('sample.txt', 'w') as file:
    file.write(data)

# Reading the same file
with open('sample.txt', 'r') as file:
    content = file.read()
    print(content)
```

Output:

```
This is a sample text.
```

File Handling using try-except Block

Handle file-related errors gracefully using exception handling.

```
try:
    file = open('non_existing.txt', 'r')
    print(file.read())
except FileNotFoundError:
```

```
    print('File not found!')
finally:
    print('Execution completed.')
```

Explanation:

- If the file does not exist, **FileNotFoundError** is handled.
- **finally** ensures that cleanup or end messages run regardless of errors.

Working with Different File Types

Python can handle text, CSV, JSON, and binary files.

Example: Writing JSON File

```
import json
student = {'name': 'Ali', 'age': 22, 'city': 'Karachi'}
with open('student.json', 'w') as file:
    json.dump(student, file)
```

Example: Reading JSON File

```
import json
with open('student.json', 'r') as file:
    data = json.load(file)
    print(data)
```

Summary of File Handling

- File handling allows reading, writing, appending, and deleting files.
- **open()** function is used to access a file in different modes.
- Always close files using **close()** or **with** statement.
- Use **os** module to check or delete files.
- Handle errors using **try-except** to prevent program crashes.
- Python supports multiple file types such as text, JSON, and CSV for data storage.

Common Use Cases

- Reading configuration files.
- Writing logs or output reports.
- Data storage and retrieval.
- Working with JSON or CSV data for data science projects.