🗁 File Handling in Python

Definition
File handling means interacting with external files (like `.txt`, `.csv`, `.json`). Programs can open, read, write, or close files. This ensures data can be stored permanently and shared between programs.

Main Purposes

- Store data permanently
- Exchange data between programs
- Logging and reporting

File Types

1. **Text files**: `.txt`, `.csv`, `.json`
2. **Binary files**: images, videos, audio, etc.

File Access Modes

- `"r"` → Read only
- `"w"` → Write (overwrites existing content)
- `"a"` → Append (adds data to the end)
- `"r+"` → Read & write

Safe File Handling

- Files should be properly opened and closed.
- If not closed, risks include memory leaks, data loss, or system errors.
- Best practice: use `with open(...) as f:` which closes automatically.

⚠ Exception Handling in Python

Definition
Exception handling ensures a program can handle unexpected errors gracefully (like invalid user input, missing files, division by zero, or network issues). Python uses **try, except, else, finally** blocks for this.

Why Errors Occur

- Wrong user input
- File not found
- Math errors (e.g., division by zero)
- Environment issues (e.g., no internet)

Common Error Types

- `SyntaxError`: coding mistake in syntax
- `NameError`: variable not defined
- `TypeError`: wrong data type in an operation
- `FileNotFoundError`: missing file
- `ZeroDivisionError`: dividing by zero

Importance of Exception Handling

- Prevents program crashes
- Improves user experience
- Allows controlled error logging/reporting
- Protects data in critical operations

Control Structures

- `try`: code that may cause an error
- `except`: executed if an error occurs
- `else`: runs if no error occurs
- `finally`: always runs (e.g., closing files, disconnecting resources)