File Handling in Python

File handling in Python enables developers to read, write, create, and manage files efficiently. Python provides built-in functions such as `open()` and `close()` to perform file operations.

1. Opening Files with `open()`

The `open()` function opens a file and returns a file object. It accepts:

- **File name or path**: The file's name or location.
- **Mode**: Specifies the action (e.g., reading, writing) and format (text or binary).

File Modes

Mode	Description
`r`	Opens a file for reading (text format, default).
`rb`	Opens a file for reading in binary format.
`r+`	Opens a file for both reading and writing.
`w`	Opens a file for writing (overwrites existing).
`a`	Opens a file for appending new data.
`wb`	Opens a file for writing in binary format.
`ab`	Opens a file for appending in binary format.

2. Closing Files with `close()`

The `close()` function terminates the connection with the file. It ensures that all resources are released.

Example:

```
file = open("test.txt", "r")  # Open file in read mode
data = file.readline()        # Read a single line
print(data)                   # Output: Hello there
file.close()                  # Close the file
```

3. Using `with open()` for File Handling

The `with open()` statement is a better approach as it:

- Automatically closes the file after the block execution.
- Handles exceptions more effectively.

Example:

```
with open("test.txt", "r") as file:
    data = file.readline()  # Read a single line
    print(data)             # Output: Hello there
# File is automatically closed after the block
```

4. Reading Files

- **`readline()`** : Reads one line from the file.
- **`readlines()`** : Reads all lines into a list.
- **`read()`** : Reads the entire content of the file as a string.

Example:

```
with open("test.txt", "r") as file:
    # Read entire file content
    content = file.read()
    print(content)          # Output: Hello there
```

5. Writing and Appending Files

- **Writing (`w`)**: Overwrites the file's content.
- **Appending (`a`)**: Adds content to the end of the file.

Example:

```
# Writing to a file
with open("test.txt", "w") as file:
    file.write("New content.\n")  # Overwrites the file

# Appending to a file
with open("test.txt", "a") as file:
    file.write("Additional content.\n")  # Adds new content
```

6. Binary vs Text Files

- **Text Format**: Default mode for file operations. Content is human-readable.
- **Binary Format**: Compact and efficient for non-text data (e.g., images, audio).

Example:

```
# Reading a binary file
with open("image.jpg", "rb") as file:
    binary_content = file.read()
    print(binary_content)
```

Summary of File Handling Steps:

1. Open the file with `open()` or `with open()`.
2. Perform the desired operation: `read`, `write`, or `append`.
3. Close the file using `close()` (if not using `with open()`).

By mastering file handling, you can work effectively with local, web, or cloud data stored in files.