

File Management in Python

Created By: The EasyLearn Academy

Contact: **9662512857**

What You Will Learn?

- How to create, open, close, delete files & folders
- How to read data from existing files
- How to write data to files
- Using context managers for safe file handling
- Working with binary files

What is a File?

- A file is a named location on disk to store related information.
- Used to permanently store data in nonvolatile memory (e.g., hard disk, USB, CD).
- Unlike RAM (volatile), files retain data when the computer is turned off.
- File Operation Workflow:
 - 1. Open a file
 - 2. Read or write (perform operations)
 - 3. Close the file to free resources

File Management in Python

- Python handles files as text or binary, impacting how they are processed.
- Text files consist of character sequences, terminated by EOL (End of Line) characters (e.g., newline \n).
- Python provides builtin functions for creating, reading, updating, and deleting files.
- The os module supports file and directory operations.

Opening and Creating Files

- Use the `open()` function to create or open a file.
- Syntax:
- `python`

```
file_object = open(file_name, access_mode, buffering)
```

Parameters:

`file_name`: Name of the file (e.g., "example.txt").

`access_mode`: Mode to open the file (e.g., `read`, `write`).
Default is `r` (`read`).

`buffering`: Optional; controls buffering (`0` = no buffering, `1` = line buffering, `>1` = buffer size).

Common File Modes

- | Mode | Description |
- | r | Readonly (default). File pointer at the start.
- | r+ | Read and write. File pointer at the start.
- | w | Writeonly. Overwrites if exists, else creates new. |
- | w+ | Read and write. Overwrites or creates new.
- | a | Append. File pointer at the end. Creates if not exists. |
- | a+ | Read and append. File pointer at the end.
- | rb, wb, ab | Binary mode for reading, writing, or appending.

Reading from Files

- Use `read()`, `readline()`, or loop over the file object to read data.
- Example: Reading a file line by line
- python

```
file = open("list.txt", "r")
for line in file:
    print(line.strip()) # strip() removes trailing newlines
file.close()
```

Using `read()`: Reads entire file or specified characters

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

Using Context Managers (with Statement)

- The with statement ensures files are properly closed, even if an error occurs.
- Eliminates the need for explicit file.close().
- Example:
- python

```
with open("example.txt", "w") as file:  
    file.write("This is safe file handling!\n")  
# File automatically closed after the block
```

Handling Exceptions

- File operations can raise errors (e.g., file not found, permission issues).
- Use tryexcept to handle exceptions gracefully.
- Example:
- python

```
try:  
    with open("nonexistent.txt", "r") as file:  
        content = file.read()  
except FileNotFoundError:  
    print("Error: File not found!")  
except PermissionError:  
    print("Error: Permission denied!")
```

-

Renaming and Deleting Files

- Use the os module for file operations.

- Rename a file:

- python

```
import os
```

```
os.rename("old_name.txt",  
"new_name.txt")
```

Delete a file:

```
python
```

```
import os
```

```
os.remove("file_to_delete.txt")
```

-

Working with Directories

```
python  
import os  
os.mkdir("new_folder")
```

Change current directory:

```
python  
os.chdir("new_folder")
```

Get current working directory:

```
python  
print(os.getcwd())
```

Remove an empty directory:

```
python  
os.rmdir("new_folder")
```

.

Additional Tips

- Use `os.path` for cross-platform compatibility:

```
python  
import os  
file_path = os.path.join("folder", "file.txt") # Creates  
correct path
```

- Check if a file/directory exists:

```
python  
os.path.exists("file.txt")
```

- Use `shutil` module for advanced operations (e.g., copying
directories):

```
python  
import shutil  
shutil.copy("source.txt", "destination.txt")
```