Python Programming
Unit 03 – Lecture 01 Notes
File Handling Fundamentals and Directories

Tofik Ali

February 14, 2026

Contents

1 Lecture Overview

Many programs become truly useful only when they can **store and load** data. File handling allows you to:

- save results (reports, logs, grades, summaries),
- reuse data later (datasets, configuration),
- and share data between programs.

This lecture covers file opening modes, reading/writing patterns, the `with` statement, and basic directory handling using `pathlib`.

2 Core Concepts

2.1 The `open()` Function

The function `open(path, mode, encoding=...)` returns a file object. You can use it to read or write.

Best practice: use `with open(...)` so the file is always closed:

```
with open("data.txt", "r", encoding="utf-8") as f:
    content = f.read()
    print(content)
```

2.2 File Modes (Read/Write/Append)

Common modes (text mode):

Mode	Meaning
`r`	read (file must exist)
`w`	write (overwrite if exists, create if not)
`a`	append (write at end, create if not)
`x`	create new (error if file exists)
`t`	text mode (default)
`b`	binary mode (images, PDFs, etc.)
`+`	update mode (read + write)

Important behavior:

- `"w"` truncates the file (clears old content). Use carefully.
- `"a"` keeps old content and adds new lines at the end.
- `"r"` fails if the file does not exist (`FileNotFoundError`).

2.3 Reading Data

Reading the whole file

```
with open("names.txt", "r", encoding="utf-8") as f:
    text = f.read()
```

Good for small files. Not recommended for very large files.

Reading line by line (recommended)

```
with open("names.txt", "r", encoding="utf-8") as f:
    for line in f:
        name = line.strip()
        print(name)
```

`strip()` removes the newline character "
n" and extra spaces.

2.4 Writing Data

To write a line, use `write()` and include a newline "
n":

```
with open("out.txt", "w", encoding="utf-8") as f:
    f.write("Alice\\n")
    f.write("Bob\\n")
```

To write many lines at once, use `writelines()`:

```
lines = ["A\\n", "B\\n", "C\\n"]
with open("out.txt", "w", encoding="utf-8") as f:
    f.writelines(lines)
```

2.5 Why `with` is Better Than Manual Close

Without `with`, you must remember `f.close()`:

```
f = open("data.txt", "r")
try:
    print(f.read())
finally:
    f.close()
```

`with` automatically does this for you. This matters on Windows because open files can remain *locked* if not closed properly.

2.6 File Pointer: `tell()` and `seek()`

Files have a current reading position.

```
with open("data.txt", "r") as f:
    print("pos =", f.tell())
    first = f.readline()
    print("after readline pos =", f.tell())
    f.seek(0) # move back to start
    again = f.readline()
```

2.7 Working with Directories (Pathlib)

The `pathlib` module gives an object-oriented way to handle paths.

```
from pathlib import Path

root = Path(".")
print("Current folder:", root.resolve())

for p in root.iterdir():
    print(p.name, "dir" if p.is_dir() else "file")
```

Common useful operations:

- Create a directory: `Path("data").mkdir(exist_ok=True)`

- Join paths: `root / "data" / "names.txt"`
- Recursive search: `root.rglob("*.py")`

3 Demo Walkthrough

3.1 Demo 1: Read/Write + Simple Stats

File: `demo/file_read_write_demo.py`
What it does:

- Creates a file in `data/` and writes a few names.
- Reads the file back.
- Computes:
  - number of names,
  - longest name,
  - count of names that start with a vowel.

3.2 Demo 2: Directory Walk

File: `demo/directory_walk_demo.py`
What it does:

- Prints a small "tree" of files under the lecture folder.
- Demonstrates `Path.rglob(...)` and basic path operations.

4 Interactive Checkpoints (with Solutions)

Checkpoint 1 Solution

Question: difference between modes `"w"` and `"a"`?
Answer:

- `"w"` overwrites (truncates) the file.
- `"a"` appends to the end of the file without deleting old content.

Checkpoint 2 Solution

Question: what happens if you open a missing file using `"r"`?
Answer: Python raises `FileNotFoundError`.

5 Practice Exercises (with Solutions)

Exercise 1: Count Lines

Task: Read a text file and print number of lines.

Solution:

```
path = input("File path: ").strip()
count = 0
with open(path, "r", encoding="utf-8") as f:
    for _ in f:
        count += 1
print("Line count =", count)
```

Exercise 2: Copy a File

Task: Copy text from one file to another.

Solution:

```
src = input("Source file: ").strip()
dst = input("Destination file: ").strip()
with open(src, "r", encoding="utf-8") as f_in:
    with open(dst, "w", encoding="utf-8") as f_out:
        for line in f_in:
            f_out.write(line)
print("Copied.")
```

Exercise 3: List Python Files

Task: Print all `.py` files inside the current directory and its subfolders.

Solution:

```
from pathlib import Path
root = Path(".")
for p in root.rglob("*.py"):
    print(p)
```

6 Exit Question (with Solution)

Question: Read all lines from `"names.txt"` using `with` and print without newlines.

Solution:

```
with open("names.txt", "r", encoding="utf-8") as f:
    for line in f:
        print(line.strip())
```