ZyBooks Chapter 12: Files in Python

Jeremy Evert

October 20, 2025

Contents

Chapter 1

12.1 Reading Files

Overview

In this section, students learn how to read from files using Python's built-in `open()` function. Files allow programs to save data permanently and later retrieve it. Instead of entering data manually each time a program runs, we can read information directly from a file.

Learning goals:

- Understand how to open, read, and close text files in Python.
- Explore the difference between `read()`, `readline()`, and `readlines()`.
- Learn to process data from files (e.g., computing averages).

1.1 Reading from a File

The most basic way to read data from a file is with `open()` and `read()`.

Listing 1.1: Reading text from a file.

```
# Example 1: Reading the entire contents of a file

# Open the file in read mode
myjournal = open("journal.txt")

# Read the entire file into a single string
contents = myjournal.read()

# Display what was read
print(contents)

# Close the file after use
myjournal.close()
```

Key points:

- `open("filename")` creates a file object.
- `read()` reads all text at once and returns it as a string.
- Always close the file using `close()` when done.

—

1.2 A More Complete Example

This version adds print statements and clarifies program flow.

Listing 1.2: Creating a file object and reading text.

```
print("Opening file myfile.txt.")
f = open("myfile.txt")  # create file object

print("Reading file myfile.txt.")
contents = f.read()     # read text into a string

print("Closing file myfile.txt.")
f.close()               # close the file

print("\nContents of myfile.txt:")
print(contents)
```

Tip: The file must be in the same directory as your Python script unless you specify a full path (e.g., `C:\Users\evertj\myfile.txt`).

—

1.3 Reading Line by Line

The `readlines()` method reads each line into a list of strings.

Listing 1.3: Reading all lines into a list.

```
# Example 2: Read lines from a file
my_file = open("readme.txt")
lines = my_file.readlines()

# Print the second line (remember, Python starts counting at 0)
print(lines[1])

my_file.close()
```

Note: Each element of `lines` includes the newline character `"\n"`.

—

1.4 Processing Data from a File

Programs often read data from files to compute a result, such as an average.

Listing 1.4: Calculating the average value of integers stored in a file.

```
# Example 3: Calculating an average from a file

print("Reading in data...")
f = open("mydata.txt")
lines = f.readlines()
f.close()

# Process data
print("\nCalculating average...")
total = 0
for ln in lines:
    total += int(ln)

avg = total / len(lines)
print(f"Average value: {avg}")
```

This example demonstrates:

- How to iterate through file lines.
- Converting strings to integers using `int()`.
- Computing an average from numeric data.

—

1.5 Iterating Directly Over a File Object

Python lets you loop through a file directly, one line at a time.

Listing 1.5: Iterating over the lines of a file.

```
"""Echo the contents of a file."""
f = open("myfile.txt")

for line in f:
    print(line, end="")  # end="" avoids double newlines

f.close()
```

This approach is memory-efficient and ideal for large files.

—

1.6 Practice Exercise

Challenge: Create a Python program that reads a filename from user input, opens that file, and prints its contents in uppercase.

Listing 1.6: Challenge Activity: Read and modify file contents.

```
# Example 4: Read and transform file content
```

```
filename = input("Enter filename: ")

with open(filename) as f:        # 'with' auto-closes the file
    contents = f.read()

print(contents.upper())
```

—

1.7 Explore More

For additional reading and examples:

- Python Documentation: Reading and Writing Files
- W3Schools: Python File Handling
- Real Python: Working with Files in Python

—

Summary

- Use `open()` to access a file.
- `read()`, `readline()`, and `readlines()` offer flexibility.
- Always close files, or use the `with` statement.
- Practice reading, processing, and displaying file data.

Notes

This book was created to accompany the zyBooks interactive textbook, Chapter 12: Files. Each section demonstrates the concepts with well-structured LaTeX code examples and clear explanations.