Intermediate Python Programming – Lesson 9

Facilitated by Kent State University
Topic: File I/O and Data Persistence
Duration: 1 Hour

Learning Objectives

By the end of this lesson, participants will be able to:

- Open, read, and write text and binary files using built-in functions
- Use context managers (`with` statements) to manage file resources safely
- Understand and apply basic data persistence strategies using files

---

Lesson 9: File I/O and Data Persistence

I. Introduction to File Operations (10 minutes)

Python makes file reading and writing simple and flexible. Files can be opened in various modes:

- `'r'` – read
- `'w'` – write (overwrites existing files)
- `'a'` – append
- `'b'` – binary mode
- `'t'` – text mode (default)

Use the built-in `open()` function:

```
file = open('example.txt', 'r')
contents = file.read()
file.close()
```

Problem: If an error occurs, the file may not close properly.

II. Using Context Managers (`with`) (10 minutes)

The `with` statement ensures the file is automatically closed:

```
with open('example.txt', 'r') as file:
    contents = file.read()
```

You can also write to a file:

```
with open('output.txt', 'w') as f:
    f.write("Hello, world!\n")
    f.write("More text\n")
```

Exercise 1:

Write a program that reads a file and prints its contents line by line.

III. Reading and Writing Text and Binary Files (15 minutes)

Reading files line-by-line:

```
with open('data.txt') as f:
    for line in f:
        print(line.strip())
```

Writing and appending:

```
with open('log.txt', 'a') as f:
    f.write("New log entry\n")
```

Working with binary data:

```
with open('image.png', 'rb') as f:
    image_bytes = f.read()

with open('copy.png', 'wb') as f:
    f.write(image_bytes)
```

Exercise 2:

Copy a binary file using `rb` and `wb` modes.

IV. Basic Data Persistence Techniques (15 minutes)

Persistence refers to storing program data between executions. Python provides multiple ways to do this:

- **Text files** (CSV, plain text)
- **JSON** (structured, readable)
- **Pickle** (binary serialization, Python-specific)

Using JSON:

```
import json

person = {"name": "Alice", "age": 30}

with open('person.json', 'w') as f:
    json.dump(person, f)

with open('person.json') as f:
    loaded = json.load(f)
```

Using Pickle:

```
import pickle

data = {'scores': [88, 92, 95]}

with open('scores.pkl', 'wb') as f:
    pickle.dump(data, f)

with open('scores.pkl', 'rb') as f:
    restored = pickle.load(f)
```

Warning: Only use Pickle with trusted sources. It can execute arbitrary code.

Exercise 3:

Create a dictionary of student names and grades and save it as JSON. Then read it back and print each student's name and grade.

V. Recap and Q&A (10 minutes)

- Use `open()` with the correct mode to read/write files
- Prefer `with` blocks to ensure files are properly closed
- Use `json` for structured, readable persistence
- Use `pickle` for compact but Python-specific binary data

Final Exercise:

Write a program that asks the user for a name and score, appends it to a JSON file, and then displays all scores so far.
