Python Primer: Using `json`, `os`, and `with` for File Persistence

When you're ready to **save and load data** in your Python programs, you may find need to use three key tools:

- `json` — to store and retrieve structured data
- `os` — to check if files exist before loading
- `with` — to safely open and close files without forgetting anything

All of these are part of the **Python Standard Library**, so you don't need to install anything.

What Is the `json` Module?

The `json` module lets you convert between **Python data** and **text files**.

JSON stands for **JavaScript Object Notation**, a popular format used by many apps to store data like dictionaries, lists, numbers, and strings.

Common `json` Functions

Function	What it Does
`json.dump(data, file)`	Saves Python data to a file
`json.load(file)`	Loads data from a file
`json.dumps(data)`	Converts Python data to a JSON string
`json.loads(string)`	Converts a JSON string to Python data

Example: Saving a List to a File

```python
import json

data = ["groceries", "homework", "call mom"]

# Save data to a file
with open("tasks.json", "w") as file:
    json.dump(data, file)
```

The `with open(...)` line automatically closes the file when you're done — even if something goes wrong. This is important for safe operation of the program. If you forget `**with**` and run into a problem, you should be able to restart your terminal, fix the problem and then run the code again. In rare cases, you may need to restart your computer.

Example: Loading the List Back

```python
import json

with open("tasks.json", "r") as file:
    data = json.load(file)

print(data)  # ['groceries', 'homework', 'call mom']
```

Why Use `with` When Opening Files?

When you open a file, Python gives your program temporary access to it.
If you don't **close the file** afterward, it can cause problems:

- Data may not be saved correctly

- The file might be locked or unreadable until you restart your program
- You can accidentally corrupt the file

Using `with` handles all of that for you.

```python
with open("filename.txt", "r") as file:
    content = file.read()
# File is automatically closed here
```

You can still use `open()` and `close()` manually, but `with` is safer and preferred.

What You Can and Can't Save with `json`

You Can Save	You Can't Save
Lists	Custom class objects
Dictionaries	Functions
Strings	File handles
Numbers	Anything not built-in

If you want to save objects (like a `Task`), you'll need to **convert it to a dictionary first**. When loading, you'll convert it back.

Think of `json` like a translator — it can only speak "simple Python."

What Is the `os` Module?

The `os` module lets your program interact with your computer's **file system**. For this project, the main thing you'll use it for is checking whether a file exists before trying to load it.

Example: Check if a File Exists

```python
import os

if os.path.exists("tasks.json"):
    print("Loading tasks...")
else:
    print("No saved tasks found.")
```

If you try to open a file in `"r"` mode and it doesn't exist, Python will crash with a `FileNotFoundError`. This check helps prevent that.

What You Should Be Able to Do After This

- Save a list or dictionary to a file using `json.dump()`
- Load a list or dictionary from a file using `json.load()`
- Use `with open(...)` to safely manage files
- Use `os.path.exists()` to check before reading
- Understand why you can't save class instances without converting them
- How do I rebuild my program's data after loading?