Python To-Do List

DAE

07/10/25

AJ

Variables & Data Types

- `tasks = []` → Creates a list to hold task dictionaries
- `TASKS_FILE = "tasks.json"` → Stores file name as a string
- Each task uses `id` (int), `description` (str), and `completed` (bool)

Functions

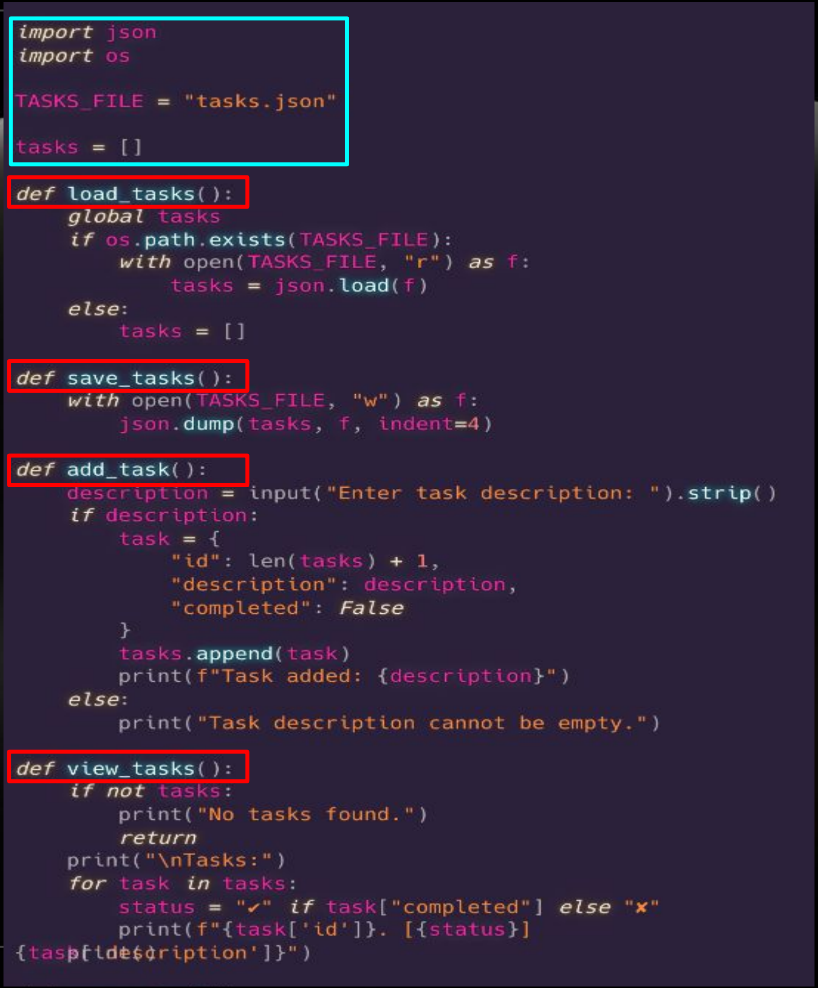
- Defined with `def` to organize code: `add_task()`, `view_tasks()`
- Functions make the code reusable and modular
- Called from the menu loop to execute specific actions

Lists

- `tasks` is a list holding all user-created tasks
- Tasks added using `tasks.append(task)`
- Looping over the list with `for task in tasks:` to display or find tasks

Dictionaries

- Each task is stored as a dictionary:
  `{"id": 1, "description": "Learn Python", "completed": False}`
- Accessed with keys like `task["description"]` or `task["completed"]`



```
import json
import os

TASKS_FILE = "tasks.json"

tasks = []
def load_tasks():
    global tasks
    if os.path.exists(TASKS_FILE):
        with open(TASKS_FILE, "r") as f:
            tasks = json.load(f)
    else:
        tasks = []

def save_tasks():
    with open(TASKS_FILE, "w") as f:
        json.dump(tasks, f, indent=4)

def add_task():
    description = input("Enter task description: ").strip()
    if description:
        task = {
            "id": len(tasks) + 1,
            "description": description,
            "completed": False
        }
        tasks.append(task)
        print(f"Task added: {description}")
    else:
        print("Task description cannot be empty.")

def view_tasks():
    if not tasks:
        print("No tasks found.")
        return
    print("\nTasks:")
    for task in tasks:
        status = "✓" if task["completed"] else "✗"
        print(f"{task['id']}. [{status}]
{task['description']}")
```

Loops

- `for task in tasks:` → Iterates over all tasks
- `while True:` → Keeps the menu active until user exits
- Loops keep the interface responsive and repeat actions

Conditionals / Decision-Making

- `if`, `elif`, `else` control the flow of logic
- Menu options selected using `if choice == '1'`, etc.
- Completion check: `status = "✔" if task["completed"] else "✘"`

File Input/Output

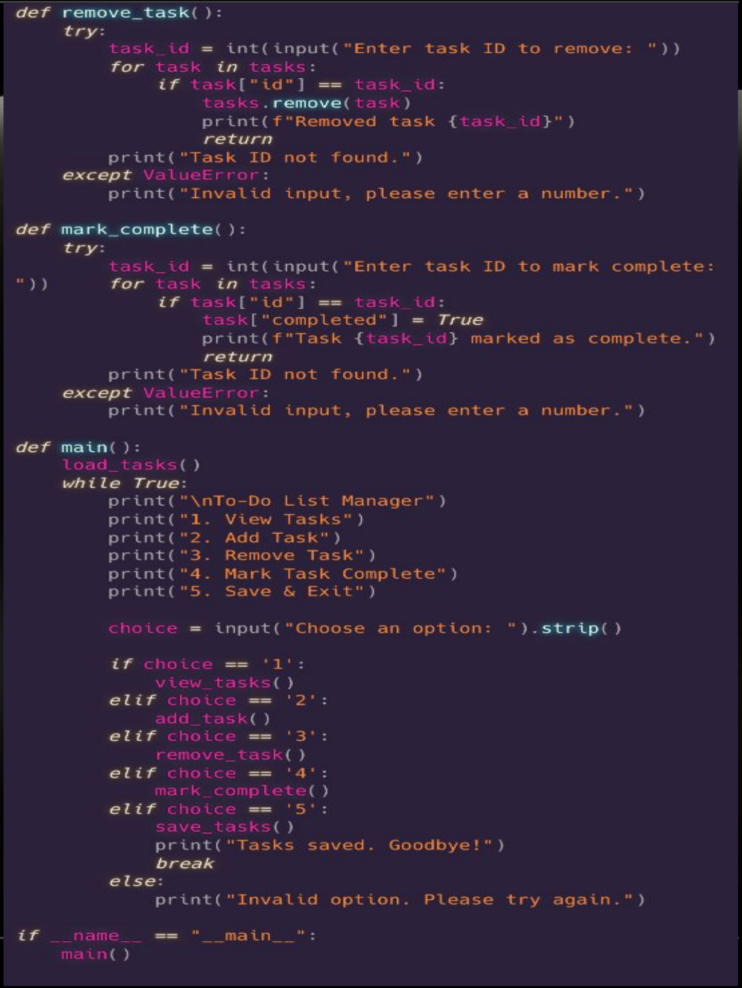
- `with open(TASKS_FILE, "r") as f:` reads saved tasks
- `with open(TASKS_FILE, "w") as f:` writes updated tasks
- `json.load()` and `json.dump()` handle file conversion to/from JSON

Exception Handling

- `try/except ValueError:` → Prevents crashes from invalid input
- Used when converting `input()` to `int` for task IDs

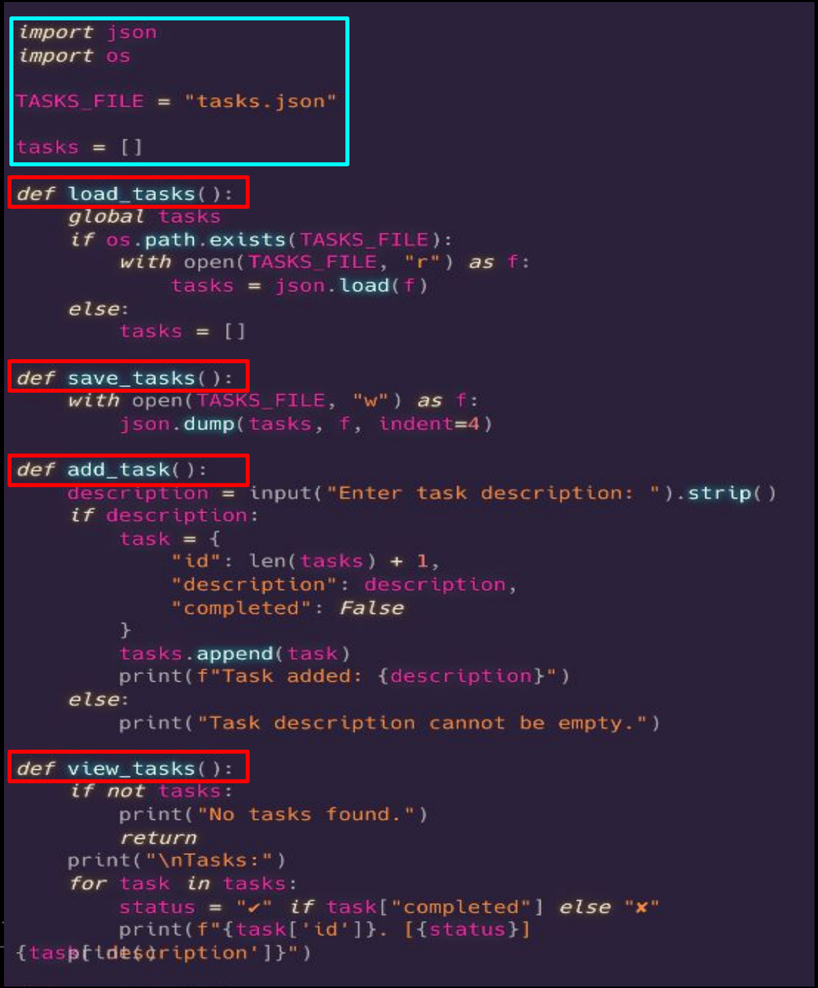
Entry Point

- `if __name__ == "__main__":` → Runs the app only when file is executed directly
- Prevents accidental execution when imported as a module



```python
def remove_task():
    try:
        task_id = int(input("Enter task ID to remove: "))
        for task in tasks:
            if task["id"] == task_id:
                tasks.remove(task)
                print(f"Removed task {task_id}")
                return
        print("Task ID not found.")
    except ValueError:
        print("Invalid input, please enter a number.")

def mark_complete():
    try:
        task_id = int(input("Enter task ID to mark complete: "))
        for task in tasks:
            if task["id"] == task_id:
                task["completed"] = True
                print(f"Task {task_id} marked as complete.")
                return
        print("Task ID not found.")
    except ValueError:
        print("Invalid input, please enter a number.")

def main():
    load_tasks()
    while True:
        print("\nTo-Do List Manager")
        print("1. View Tasks")
        print("2. Add Task")
        print("3. Remove Task")
        print("4. Mark Task Complete")
        print("5. Save & Exit")

        choice = input("Choose an option: ").strip()

        if choice == '1':
            view_tasks()
        elif choice == '2':
            add_task()
        elif choice == '3':
            remove_task()
        elif choice == '4':
            mark_complete()
        elif choice == '5':
            save_tasks()
            print("Tasks saved. Goodbye!")
            break
        else:
            print("Invalid option. Please try again.")

if __name__ == "__main__":
    main()
```

- `json` lets us save and load structured task data.
- `os` checks if the file already exists before loading.
- `TASKS_FILE` is the filename for saving.
- `tasks` holds the task list (a list of dictionaries).
- Def=functions



```python
import json
import os

TASKS_FILE = "tasks.json"

tasks = []

def load_tasks():
    global tasks
    if os.path.exists(TASKS_FILE):
        with open(TASKS_FILE, "r") as f:
            tasks = json.load(f)
    else:
        tasks = []

def save_tasks():
    with open(TASKS_FILE, "w") as f:
        json.dump(tasks, f, indent=4)

def add_task():
    description = input("Enter task description: ").strip()
    if description:
        task = {
            "id": len(tasks) + 1,
            "description": description,
            "completed": False
        }
        tasks.append(task)
        print(f"Task added: {description}")
    else:
        print("Task description cannot be empty.")

def view_tasks():
    if not tasks:
        print("No tasks found.")
        return
    print("\nTasks:")
    for task in tasks:
        status = "✓" if task["completed"] else "x"
        print(f"{task['id']}. [{status}]
{task['description']}")
```

- `json.dump()` → "**DUMP** this object **into** a file"
- `indent=4` → "Make it **pretty** with 4-space indents"
- ("Take the `tasks` list, turn it into formatted JSON text, and write it to the open file `f` with nice spacing (4 spaces per indent level)."
-