Expense tracker ( final project)

import json

# Task class to represent each task

class Task:

def \_\_init\_\_(self, title, description, category):

self.title = title

self.description = description

self.category = category

self.completed = False

def mark\_completed(self):

self.completed = True

# Load tasks from a JSON file

def load\_tasks(filename='tasks.json'):

try:

with open(filename, 'r') as file:

return [Task(\*\*data) for data in json.load(file)]

except (FileNotFoundError, json.JSONDecodeError):

return [] # Return an empty list if the file does not exist or is empty

# Save tasks to a JSON file

def save\_tasks(tasks, filename='tasks.json'):

with open(filename, 'w') as file:

json.dump([task.\_\_dict\_\_ for task in tasks], file, indent=4)

# Add a new task

def add\_task(tasks):

title = input("Enter task title: ").strip()

description = input("Enter task description: ").strip()

category = input("Enter task category (e.g., Work, Personal, Urgent): ").strip()

if title: # Ensure the title is not empty

task = Task(title, description, category)

tasks.append(task) # Add the task to the list

print("Task added successfully.")

else:

print("Task title cannot be empty.")

# View all tasks

def view\_tasks(tasks):

if not tasks:

print("No tasks available.")

return

for idx, task in enumerate(tasks):

status = "✓" if task.completed else "✗" # Show completion status

print(f"{idx + 1}. [{status}] {task.title} - {task.category}\n {task.description}")

# Mark a task as completed

def mark\_task\_completed(tasks):

if not tasks:

print("No tasks to mark as completed.")

return

view\_tasks(tasks) # Display current tasks

try:

task\_number = int(input("Enter task number to mark as completed: ")) - 1

if 0 <= task\_number < len(tasks):

tasks[task\_number].mark\_completed() # Mark the task

print("Task marked as completed.")

else:

print("Invalid task number.")

except ValueError:

print("Invalid input. Please enter a valid number.")

# Edit a task

def edit\_task(tasks):

if not tasks:

print("No tasks to edit.")

return

view\_tasks(tasks) # Display current tasks

try:

task\_number = int(input("Enter task number to edit: ")) - 1

if 0 <= task\_number < len(tasks):

# Prompt for new values

new\_title = input("Enter new title (or press Enter to keep current): ").strip()

new\_description = input("Enter new description (or press Enter to keep current): ").strip()

new\_category = input("Enter new category (or press Enter to keep current): ").strip()

# Update fields if new values are provided

if new\_title:

tasks[task\_number].title = new\_title

if new\_description:

tasks[task\_number].description = new\_description

if new\_category:

tasks[task\_number].category = new\_category

print("Task updated successfully.")

else:

print("Invalid task number.")

except ValueError:

print("Invalid input. Please enter a valid number.")

# Delete a task

def delete\_task(tasks):

if not tasks:

print("No tasks to delete.")

return

view\_tasks(tasks) # Display current tasks

try:

task\_number = int(input("Enter task number to delete: ")) - 1

if 0 <= task\_number < len(tasks):

tasks.pop(task\_number) # Remove the task

print("Task deleted successfully.")

else:

print("Invalid task number.")

except ValueError:

print("Invalid input. Please enter a valid number.")

# Main menu loop

def todo\_list\_manager\_menu():

tasks = load\_tasks() # Load tasks from file

while True:

print("\n--- Personal To-Do List Manager ---")

print("1. Add Task")

print("2. View Tasks")

print("3. Mark Task as Completed")

print("4. Edit Task")

print("5. Delete Task")

print("6. Exit")

choice = input("Choose an option (1-6): ").strip()

if choice == "1":

add\_task(tasks)

save\_tasks(tasks) # Save tasks after adding

elif choice == "2":

view\_tasks(tasks) # View all tasks

elif choice == "3":

mark\_task\_completed(tasks)

save\_tasks(tasks) # Save tasks after marking completed

elif choice == "4":

edit\_task(tasks)

save\_tasks(tasks) # Save tasks after editing

elif choice == "5":

delete\_task(tasks)

save\_tasks(tasks) # Save tasks after deletion

elif choice == "6":

print("Exiting the application. Goodbye!")

break

else:

print("Invalid choice. Please select a valid option.")

# Entry point of the program

if \_\_name\_\_ == "\_\_main\_\_": # Corrected entry point

todo\_list\_manager\_menu()