SUMMARY OF THE PROGRAM CODE

🧠 Goal of the Program

This Python application is a **desktop-based AI task scheduler**. It allows users to:

- Add, modify, mark complete, or delete tasks.
- Automatically determine task **priority using AI (NLP)**.
- Store all tasks in a local **SQLite database**.
- Display tasks in a well-organized, modern GUI using **ttkbootstrap**.

1️⃣ Imports

```
import tkinter as tk
from tkinter import ttk, messagebox
import sqlite3
import threading
import spacy
from datetime import datetime
import ttkbootstrap as tb
```

What Each Module Does:

Module	Purpose
tkinter, ttk, messagebox	Create GUI, table view, and pop-up alerts
sqlite3	Manage a local database to store tasks
threading	Run background processes without freezing the GUI
spacy	Perform natural language processing (NLP) for task description
datetime	Validate date format
ttkbootstrap	A modern-themed version of tkinter widgets

2️⃣ Loading the NLP Model

```
try:
    nlp = spacy.load("en_core_web_sm")
```

except OSError:

messagebox.showerror("Error", "SpaCy model not found...")

exit()

- Loads the **SpaCy small English model** to process text input.
- If the model isn't installed, it shows an error message and exits the program.

3️⃣ Class: TaskScheduler

This class controls the entire app: GUI, logic, database, and task handling.

a) __init__(self, root)

def __init__(self, root):

- Initializes the main window.
- Sets the theme using ttkbootstrap.
- Calls:
  - init_database() – creates or connects to the task database.
  - create_gui() – builds the user interface.
  - A thread for check_tasks() – checks for updates every hour.

---

b) init_database(self)

def init_database(self):

- Uses SQLite to create a local database file tasks.db.
- Creates a table tasks with the following columns:
  - id (Primary Key)
  - title
  - description
  - due_date
  - priority (Low, Medium, High)
  - status (Pending or Completed)
  - ai_reason (Why the AI assigned that priority)

---

c) ai_prioritize_task(self, description)

def ai_prioritize_task(self, description):

- Uses **SpaCy** to process the task description.
- Assigns priorities based on keywords:
  - If words like “urgent”, “deadline”, “critical” are found → **High**
  - If words like “soon”, “major”, “priority” are found → **Medium**
  - Otherwise → **Low**
- Returns the priority and the reasoning method.

---

d) validate_date(self, date_str)

def validate_date(self, date_str):

- Uses Python’s datetime.strptime() to check if the date is valid and in the format **YYYY-MM-DD**.
- Returns True or False.

4️⃣GUI: create_gui(self)

This builds the full user interface.

Components:

🎯 **Title Label**

tb.Label(... text="AI Task Scheduler")

📝 **Input Fields**

- Title, Description, Due Date (in YYYY-MM-DD format)
- User fills these to add/modify tasks.

🧮 **Buttons**

- **Add Task** – inserts into database
- **Modify Task** – updates existing task
- **Mark as Done** – updates status to Completed
- **Delete Task** – removes task

📋 **Task List**

- A **Treeview** table shows:
  - ID, Title, Description, Due Date, Priority, Status

- Dynamically refreshed after every change.

5️⃣ Task Functions

add_task(self)

- Gathers form inputs.
- Validates title and date.
- Gets AI priority from ai_prioritize_task().
- Inserts task into the SQLite table.
- Calls refresh_tasks() to update the list.

---

modify_task(self)

- Edits the selected task.
- Validates new inputs.
- Updates the task and recalculates AI priority.

---

mark_task_done(self)

- Changes the status of the selected task to **"Completed"**.

---

delete_task(self)

- Deletes the selected task from the database.

---

refresh_tasks(self)

- Clears the current tree view.
- Pulls all tasks from the database.
- Orders them by:
    1. Priority (High > Medium > Low)
    2. Due date (earliest first)

---

check_tasks(self)

- A background thread that waits for 1 hour (3600s) before executing again.

- Currently passive, but could later be used to:
    - Alert user of upcoming deadlines
    - Trigger reminders

6️⃣ Main Program Launcher

if __name__ == "__main__":

❗ The original has _name_, which is incorrect syntax. Should be __name__.

What it does:

- Creates the application window with a modern superhero theme.
- Starts the TaskScheduler class to run the app.

✅ Summary of Features

Feature	Details
✅ NLP Priority	Uses SpaCy to detect task urgency from description
✅ CRUD	Add, Modify, Complete, Delete
✅ Database	Uses SQLite for local storage
✅ GUI	Built with Tkinter + ttkbootstrap
✅ Smart Sorting	Displays tasks by priority and deadline
🔁 Background Thread	Future potential for reminders or updates

---

Would you like me to:

- Add **reminders/notifications**?
- Export tasks to **CSV or Excel**?
- Convert it to a **web app** using Flask or Django?

Let me know how you want to expand or improve it!