import tkinter as tk

from tkinter import messagebox

from tkinter import ttk

class Task:

def \_\_init\_\_(self, name, deadline, category="General", status="Pending"):

self.name = name

self.deadline = deadline

self.category = category

self.status = status

def \_\_str\_\_(self):

return f"{self.name} - Due: {self.deadline} (Category: {self.category}, Status: {self.status})"

class TaskManagerApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Task and Time Management System")

self.tasks = []

# UI Components

self.task\_listbox = tk.Listbox(root, height=15, width=50)

self.task\_listbox.grid(row=0, column=0, rowspan=6, padx=10, pady=10)

self.add\_task\_button = tk.Button(root, text="Add Task", command=self.add\_task)

self.add\_task\_button.grid(row=0, column=1, padx=10)

self.edit\_task\_button = tk.Button(root, text="Edit Task", command=self.edit\_task)

self.edit\_task\_button.grid(row=1, column=1, padx=10)

self.delete\_task\_button = tk.Button(root, text="Delete Task", command=self.delete\_task)

self.delete\_task\_button.grid(row=2, column=1, padx=10)

self.gantt\_chart\_button = tk.Button(root, text="Display Gantt Chart", command=self.display\_gantt\_chart)

self.gantt\_chart\_button.grid(row=3, column=1, padx=10)

self.calendar\_view\_button = tk.Button(root, text="Display Calendar View", command=self.display\_calendar\_view)

self.calendar\_view\_button.grid(row=4, column=1, padx=10)

self.exit\_button = tk.Button(root, text="Exit", command=root.quit)

self.exit\_button.grid(row=5, column=1, padx=10)

self.refresh\_task\_list()

def refresh\_task\_list(self):

self.task\_listbox.delete(0, tk.END)

for task in self.tasks:

self.task\_listbox.insert(tk.END, str(task))

def add\_task(self):

self.open\_task\_window("Add Task")

def edit\_task(self):

selected\_index = self.task\_listbox.curselection()

if not selected\_index:

messagebox.showwarning("No Selection", "Please select a task to edit.")

return

selected\_task = self.tasks[selected\_index[0]]

self.open\_task\_window("Edit Task", selected\_task)

def delete\_task(self):

selected\_index = self.task\_listbox.curselection()

if not selected\_index:

messagebox.showwarning("No Selection", "Please select a task to delete.")

return

del self.tasks[selected\_index[0]]

self.refresh\_task\_list()

messagebox.showinfo("Task Deleted", "Task has been deleted successfully.")

def display\_gantt\_chart(self):

messagebox.showinfo("Gantt Chart", "Gantt Chart visualization is under development.")

def display\_calendar\_view(self):

messagebox.showinfo("Calendar View", "Calendar View visualization is under development.")

def open\_task\_window(self, title, task=None):

def save\_task():

name = name\_entry.get()

deadline = deadline\_entry.get()

category = category\_entry.get()

if not name or not deadline:

messagebox.showerror("Input Error", "Name and Deadline are required.")

return

if task: # Editing existing task

task.name = name

task.deadline = deadline

task.category = category

else: # Adding new task

self.tasks.append(Task(name, deadline, category))

task\_window.destroy()

self.refresh\_task\_list()

task\_window = tk.Toplevel(self.root)

task\_window.title(title)

tk.Label(task\_window, text="Task Name:").grid(row=0, column=0, padx=5, pady=5)

name\_entry = tk.Entry(task\_window, width=30)

name\_entry.grid(row=0, column=1, padx=5, pady=5)

if task:

name\_entry.insert(0, task.name)

tk.Label(task\_window, text="Deadline (YYYY-MM-DD):").grid(row=1, column=0, padx=5, pady=5)

deadline\_entry = tk.Entry(task\_window, width=30)

deadline\_entry.grid(row=1, column=1, padx=5, pady=5)

if task:

deadline\_entry.insert(0, task.deadline)

tk.Label(task\_window, text="Category:").grid(row=2, column=0, padx=5, pady=5)

category\_entry = tk.Entry(task\_window, width=30)

category\_entry.grid(row=2, column=1, padx=5, pady=5)

if task:

category\_entry.insert(0, task.category)

save\_button = tk.Button(task\_window, text="Save", command=save\_task)

save\_button.grid(row=3, column=0, columnspan=2, pady=10)

def main():

root = tk.Tk()

app = TaskManagerApp(root)

root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

main()

### **程式說明**

本程式是一個基於 Python tkinter 的圖形化任務與時間管理系統，模擬一個簡化版本的任務管理工具，提供以下功能：

1. **新增任務**
2. **編輯任務**
3. **刪除任務**
4. **列出所有任務**
5. **甘特圖與日曆視圖（未來功能佔位）**

### **功能細節**

#### **1. 新增任務**

用戶可以點擊「**Add Task**」按鈕，在新視窗中輸入以下資訊：

* **任務名稱**：任務的標題。
* **截止日期**：以格式 YYYY-MM-DD 指定任務完成期限。
* **分類**：可選填任務類別（例如：工作、學習、個人）。

新增後，任務會自動加入列表並更新顯示。

#### **2. 編輯任務**

用戶可以選取任務，點擊「**Edit Task**」按鈕進行編輯。在新視窗中可修改：

* **任務名稱**
* **截止日期**
* **分類**

修改完成後，點擊「Save」按鈕以保存更改。

#### **3. 刪除任務**

用戶可選擇任務，點擊「**Delete Task**」按鈕刪除該任務。系統會即時更新任務列表。

#### **4. 列出所有任務**

任務清單會即時更新，顯示每個任務的詳細資訊：

* 任務名稱
* 截止日期
* 分類
* 狀態（預設為 Pending）

#### **5. 甘特圖與日曆視圖**

目前這兩個功能為佔位功能：

* 點擊「**Display Gantt Chart**」按鈕時，會彈出訊息框提示此功能正在開發中。
* 點擊「**Display Calendar View**」按鈕時，顯示相同提示。

未來可利用外部工具如 matplotlib 或 plotly 將這些功能視覺化。

### **技術細節與架構**

#### **1. 程式架構**

* **Task 類別**：
  + 負責封裝任務的資料，包括名稱、截止日期、分類和狀態。
  + 提供基本初始化與資料結構的定義。
* **TaskManagerApp 類別**：
  + 實現 GUI 主要邏輯，包括按鈕點擊事件和介面更新。
  + 使用 Listbox 顯示任務清單，並提供互動功能。
* **視窗互動**：
  + 使用 tk.Toplevel 顯示新增或編輯任務的彈出視窗。
  + 使用 tk.messagebox 提供錯誤提示或成功訊息。

#### **2. tkinter 介面設計**

* **主要介面**
  + 任務清單透過 Listbox 顯示。
  + 各功能（新增、編輯、刪除、甘特圖、日曆視圖、退出）以按鈕的形式呈現。
* **彈出視窗**
  + 獨立的子視窗用於新增或編輯任務。
  + 輸入框 (Entry) 提供輸入介面。

#### **3. 使用事件處理**

按鈕事件連接到相關的功能處理方法，例如：

* 點擊「Add Task」會觸發 self.add\_task()，顯示新增任務的視窗。
* 點擊「Delete Task」會觸發 self.delete\_task()，刪除選中的任務。

### **執行步驟**

1. **環境需求**：
   * 安裝 Python（3.7 或更高版本）。
   * 本程式無需額外的外部庫。
2. **執行程式**：
   * 將程式碼儲存為 .py 檔案（例如 task\_manager.py）。
   * 在終端機或 IDE 中執行程式：python task\_manager.py。
3. **使用步驟**：
   * 啟動程式後，主介面會顯示任務清單區域及功能按鈕。
   * 根據需求操作各按鈕進行任務管理。

### **期望功能擴展**

1. **資料儲存功能**：
   * 使用 SQLite、CSV 或 JSON 保存任務資料，實現任務的持久化存儲。
   * 開啟程式時，自動載入已儲存的任務。
2. **甘特圖與日曆視圖**：
   * 結合 matplotlib 實現甘特圖展示任務進度。
   * 使用第三方工具如 tkcalendar 進行日曆視圖展示。
3. **多語系支援**：
   * 為介面添加語言切換功能（如繁體中文與英文）。
4. **進一步優化用戶體驗**：
   * 提供任務優先級排序。
   * 支援拖放式任務調整。

### **設計理念對應文件內容**

1. **​**：
   * 簡化界面交互體驗：透過直觀的 GUI 設計，減少學習成本，適合初學者。
   * 加入視覺化工具（甘特圖、日曆視圖）：為未來版本提供設計基礎。
2. **​**：
   * 流程設計結合 GUI，實現從用戶登入到任務管理的完整操作邏輯。