import tkinter as tk

from tkinter import ttk, messagebox, simpledialog

import csv

import matplotlib.pyplot as plt

class ExpenseTrackerApp(tk.Tk):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.title("Expense Tracker")

        self.geometry("1300x600")

        self.expenses = []

        self.categories = [

            "Food",

            "Transportation",

            "Utilities",

            "Entertainment",

            "Other",

        ]

        self.category\_var = tk.StringVar(self)

        self.category\_var.set(self.categories[0])

        self.currencies = ["USD", "EUR", "GBP", "JPY", "INR"]

        self.create\_widgets()

    def create\_widgets(self):

        self.label = tk.Label(

            self, text="Expense Tracker", font=("Helvetica", 20, "bold")

        )

        self.label.pack(pady=10)

        self.frame\_input = tk.Frame(self)

        self.frame\_input.pack(pady=10)

        self.expense\_label = tk.Label(

            self.frame\_input, text="Expense Amount:", font=("Helvetica", 12)

        )

        self.expense\_label.grid(row=0, column=0, padx=5)

        self.expense\_entry = tk.Entry(

            self.frame\_input, font=("Helvetica", 12), width=15

        )

        self.expense\_entry.grid(row=0, column=1, padx=5)

        self.item\_label = tk.Label(

            self.frame\_input, text="Item Description:", font=("Helvetica", 12)

        )

        self.item\_label.grid(row=0, column=2, padx=5)

        self.item\_entry = tk.Entry(self.frame\_input, font=("Helvetica", 12), width=20)

        self.item\_entry.grid(row=0, column=3, padx=5)

        self.category\_label = tk.Label(

            self.frame\_input, text="Category:", font=("Helvetica", 12)

        )

        self.category\_label.grid(row=0, column=4, padx=5)

        self.category\_dropdown = ttk.Combobox(

            self.frame\_input,

            textvariable=self.category\_var,

            values=self.categories,

            font=("Helvetica", 12),

            width=15,

        )

        self.category\_dropdown.grid(row=0, column=5, padx=5)

        self.date\_label = tk.Label(

            self.frame\_input, text="Date (YYYY-MM-DD):", font=("Helvetica", 12)

        )

        self.date\_label.grid(row=0, column=6, padx=5)

        self.date\_entry = tk.Entry(self.frame\_input, font=("Helvetica", 12), width=15)

        self.date\_entry.grid(row=0, column=7, padx=5)

        self.add\_button = tk.Button(self, text="Add Expense", command=self.add\_expense)

        self.add\_button.pack(pady=5)

        self.frame\_list = tk.Frame(self)

        self.frame\_list.pack(pady=10)

        self.scrollbar = tk.Scrollbar(self.frame\_list)

        self.scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

        self.expense\_listbox = tk.Listbox(

            self.frame\_list,

            font=("Helvetica", 12),

            width=70,

            yscrollcommand=self.scrollbar.set,

        )

        self.expense\_listbox.pack(pady=5)

        self.scrollbar.config(command=self.expense\_listbox.yview)

        self.edit\_button = tk.Button(

            self, text="Edit Expense", command=self.edit\_expense

        )

        self.edit\_button.pack(pady=5)

        self.delete\_button = tk.Button(

            self, text="Delete Expense", command=self.delete\_expense

        )

        self.delete\_button.pack(pady=5)

        self.save\_button = tk.Button(

            self, text="Save Expenses", command=self.save\_expenses

        )

        self.save\_button.pack(pady=5)

        self.total\_label = tk.Label(

            self, text="Total Expenses:", font=("Helvetica", 12)

        )

        self.total\_label.pack(pady=5)

        self.show\_chart\_button = tk.Button(

            self, text="Show Expenses Chart", command=self.show\_expenses\_chart

        )

        self.show\_chart\_button.pack(pady=5)

        self.update\_total\_label()

    def add\_expense(self):

        expense = self.expense\_entry.get()

        item = self.item\_entry.get()

        category = self.category\_var.get()

        date = self.date\_entry.get()

        if expense and date:

            self.expenses.append((expense, item, category, date))

            self.expense\_listbox.insert(

                tk.END, f"{expense} - {item} - {category} ({date})"

            )

            self.expense\_entry.delete(0, tk.END)

            self.item\_entry.delete(0, tk.END)

            self.date\_entry.delete(0, tk.END)

        else:

            messagebox.showwarning("Warning", "Expense and Date cannot be empty.")

        self.update\_total\_label()

    def edit\_expense(self):

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

        if selected\_index:

            selected\_index = selected\_index[0]

            selected\_expense = self.expenses[selected\_index]

            new\_expense = simpledialog.askstring(

                "Edit Expense", "Enter new expense:", initialvalue=selected\_expense[0]

            )

            if new\_expense:

                self.expenses[selected\_index] = (

                    new\_expense,

                    selected\_expense[1],

                    selected\_expense[2],

                    selected\_expense[3],

                )

                self.refresh\_list()

                self.update\_total\_label()

    def delete\_expense(self):

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

        if selected\_index:

            selected\_index = selected\_index[0]

            del self.expenses[selected\_index]

            self.expense\_listbox.delete(selected\_index)

            self.update\_total\_label()

    def refresh\_list(self):

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

        for expense, item, category, date in self.expenses:

            self.expense\_listbox.insert(

                tk.END, f"{expense} - {item} - {category} ({date})"

            )

    def update\_total\_label(self):

        total\_expenses = sum(float(expense[0]) for expense in self.expenses)

        self.total\_label.config(text=f"Total Expenses: USD {total\_expenses:.2f}")

    def save\_expenses(self):

        with open("expenses.csv", "w", newline="") as csvfile:

            writer = csv.writer(csvfile)

            column\_headers = ["Expense Amount", "Item Description", "Category", "Date"]

            writer.writerow(column\_headers)

            for expense in self.expenses:

                writer.writerow(expense)

    def show\_expenses\_chart(self):

        category\_totals = {}

        for expense, \_, category, \_ in self.expenses:

            try:

                amount = float(expense)

            except ValueError:

                continue

            category\_totals[category] = category\_totals.get(category, 0) + amount

        categories = list(category\_totals.keys())

        expenses = list(category\_totals.values())

        plt.figure(figsize=(8, 6))

        plt.pie(

            expenses, labels=categories, autopct="%1.1f%%", startangle=140, shadow=True

        )

        plt.axis("equal")

        plt.title(f"Expense Categories Distribution (USD)")

        plt.show()

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

    app = ExpenseTrackerApp()

    app.mainloop()