import tkinter as tk  
from tkinter import messagebox, simpledialog  
  
  
def main\_window():  
 *"""  
 Sets up the main window for user inputs.  
 This window allows the user to enter the total weight, dolly weight,  
 and whether the load is a bulk load.  
 """* global main\_win # Main window reference  
 main\_win = tk.Tk() # Create the main window  
 main\_win.title("FedEx Weight Calculator") # Set the window title  
  
 # Labels for user input  
 tk.Label(main\_win, text="Enter Total Weight (1100 - 9999 lbs):").grid(row=0, column=0)  
 tk.Label(main\_win, text="Enter Dolly Weight (1000 - 3000 lbs):").grid(row=1, column=0)  
 tk.Label(main\_win, text="Is this a bulk load? (yes/no):").grid(row=2, column=0)  
  
 # Entry fields for user input  
 global total\_weight\_entry # Entry for total weight  
 global dolly\_weight\_entry # Entry for dolly weight  
 global bulk\_load\_entry # Entry for bulk load status  
  
 total\_weight\_entry = tk.Entry(main\_win) # Create entry for total weight  
 dolly\_weight\_entry = tk.Entry(main\_win) # Create entry for dolly weight  
 bulk\_load\_entry = tk.Entry(main\_win) # Create entry for bulk load status  
  
 # Place entry fields in the grid  
 total\_weight\_entry.grid(row=0, column=1)  
 dolly\_weight\_entry.grid(row=1, column=1)  
 bulk\_load\_entry.grid(row=2, column=1)  
  
 # Buttons for actions  
 tk.Button(main\_win, text="Calculate Final Weight", command=calculate\_final\_weight).grid(row=3,  
 column=0) # Calculate button  
 tk.Button(main\_win, text="Exit", command=main\_win.quit).grid(row=3, column=1) # Exit button  
  
 main\_win.mainloop() # Start the main event loop  
  
  
def calculate\_final\_weight():  
 *"""  
 Calculates the final weight based on user inputs.  
 Retrieves input values, validates them, and computes the final weight.  
 Displays the result in a message box.  
 """* try:  
 # Validate and retrieve weights  
 total\_weight = validate\_weight(total\_weight\_entry.get(), 1100, 9999, "Total weight") # Validate total weight  
 dolly\_weight = validate\_weight(dolly\_weight\_entry.get(), 1000, 3000, "Dolly weight") # Validate dolly weight  
 bulk\_load = bulk\_load\_entry.get().strip().lower() # Get and normalize bulk load input  
  
 # Check if the bulk load input is valid  
 if bulk\_load not in ['yes', 'no']:  
 raise ValueError("Please enter 'yes' or 'no' for bulk load.")  
  
 # Calculate final weight based on whether it is a bulk load  
 if bulk\_load == 'yes':  
 # Ask for container weight if it is a bulk load  
 container\_weight = validate\_weight(  
 simpledialog.askstring("Input", "Enter Container Weight (100 - 300 lbs):"), 100, 300,  
 "Container weight") # Validate container weight  
 final\_weight = total\_weight - dolly\_weight - container\_weight # Calculate final weight for bulk load  
 else:  
 final\_weight = total\_weight - dolly\_weight # Calculate final weight for non-bulk load  
  
 # Show the final weight in a message box  
 messagebox.showinfo("Final Weight", f"The final weight is {final\_weight:.2f} lbs.")  
  
 except ValueError as ve:  
 # Handle input errors and show a message box with the error  
 messagebox.showerror("Input Error", str(ve))  
 except Exception as e:  
 # Handle unexpected errors  
 messagebox.showerror("Error", "An unexpected error occurred.")  
  
  
def validate\_weight(value, min\_val, max\_val, field\_name):  
 *"""  
 Validates the weight input.  
 Checks if the input is empty, a valid number, and within the specified range.  
  
 Args:  
 value: The input value to validate.  
 min\_val: Minimum acceptable value.  
 max\_val: Maximum acceptable value.  
 field\_name: Name of the field for error messages.  
  
 Returns:  
 The validated weight as a float.  
 """* if not value: # Check if input is empty  
 raise ValueError(f"{field\_name} cannot be empty.")  
 try:  
 weight = float(value) # Convert the input to float  
 except ValueError:  
 raise ValueError(f"{field\_name} must be a number.") # Handle non-numeric input  
  
 # Check if the weight is within the specified range  
 if not (min\_val <= weight <= max\_val):  
 raise ValueError(f"{field\_name} must be between {min\_val} and {max\_val}.")  
  
 return weight # Return the validated weight  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main\_window() # Run the main window function to start the application