# Fancy To-Do List App - Code with Explanation

## Code:

import tkinter as tk  
from tkinter import messagebox

## Explanation:

We import the tkinter module for GUI creation and messagebox for showing pop-up messages.

## Code:

def add\_task():  
 task = entry.get().strip()  
 if task:  
 listbox.insert(tk.END, task)  
 entry.delete(0, tk.END)  
 else:  
 messagebox.showwarning("Input Error", "Please enter a task.")

## Explanation:

This function adds a new task from the input field to the listbox. If the input is empty, it shows a warning.

## Code:

def delete\_task():  
 selected = listbox.curselection()  
 if selected:  
 listbox.delete(selected[0])  
 else:  
 messagebox.showwarning("Selection Error", "Please select a task to delete.")

## Explanation:

Deletes the selected task from the list. If no task is selected, a warning is shown.

## Code:

def view\_tasks():  
 tasks = listbox.get(0, tk.END)  
 if tasks:  
 all\_tasks = "\n".join([f"{i+1}. {task}" for i, task in enumerate(tasks)])  
 messagebox.showinfo("📋 Your Tasks", all\_tasks)  
 else:  
 messagebox.showinfo("📭 No Tasks", "Your to-do list is empty.")

## Explanation:

Displays all current tasks in a pop-up message. If the list is empty, it informs the user.

## Code:

root = tk.Tk()  
root.title("📝 Fancy To-Do List")  
root.geometry("400x500")  
root.configure(bg="#f5f5f5")  
root.resizable(False, False)

## Explanation:

Initializes the main GUI window with a title, size, background color, and non-resizable properties.

## Code:

label = tk.Label(root, text="My To-Do List 🗂️", font=("Helvetica", 20, "bold"), bg="#f5f5f5", fg="#333")  
label.pack(pady=20)

## Explanation:

Creates the main heading for the app.

## Code:

entry = tk.Entry(root, font=("Helvetica", 14), width=28, bd=2, relief="solid", justify="center")  
entry.pack(pady=10)

## Explanation:

Creates the entry box where users can type their tasks.

## Code:

button\_frame = tk.Frame(root, bg="#f5f5f5")  
button\_frame.pack(pady=10)

## Explanation:

Creates a frame to hold the buttons.

## Code:

btn\_style = {  
 "font": ("Helvetica", 12, "bold"),  
 "bd": 0,  
 "width": 12,  
 "padx": 5,  
 "pady": 5,  
 "activebackground": "#e0e0e0"  
}

## Explanation:

Defines a dictionary for styling all buttons uniformly.

## Code:

add\_btn = tk.Button(button\_frame, text="➕ Add", command=add\_task, bg="#4CAF50", fg="white", \*\*btn\_style)  
add\_btn.pack(side=tk.LEFT, padx=5)

## Explanation:

Creates the 'Add' button with styling and command binding.

## Code:

delete\_btn = tk.Button(button\_frame, text="🗑️ Delete", command=delete\_task, bg="#F44336", fg="white", \*\*btn\_style)  
delete\_btn.pack(side=tk.LEFT, padx=5)

## Explanation:

Creates the 'Delete' button with styling and command binding.

## Code:

view\_btn = tk.Button(button\_frame, text="👁️ View", command=view\_tasks, bg="#2196F3", fg="white", \*\*btn\_style)  
view\_btn.pack(side=tk.LEFT, padx=5)

## Explanation:

Creates the 'View' button that displays all current tasks.

## Code:

listbox\_frame = tk.Frame(root)  
listbox\_frame.pack(pady=20)

## Explanation:

Creates a frame to hold the task list.

## Code:

listbox = tk.Listbox(listbox\_frame, width=40, height=15, font=("Helvetica", 12), bd=2, relief="groove", selectbackground="#ddd")  
listbox.pack()

## Explanation:

Creates the actual listbox to display tasks.

## Code:

footer = tk.Label(root, text="Created with ❤️ in Python", font=("Arial", 10), bg="#f5f5f5", fg="#888")  
footer.pack(pady=10)

## Explanation:

Adds a footer note at the bottom of the app window.

## Code:

root.mainloop()

## Explanation:

Starts the GUI event loop to make the window responsive.