| | |
|---|---|
| **Name** | **Farhan-Ul-Haq** |
| **Sap id** | **55853** |
| **Submitted to** | **Sir, Usman Sharif** |
| **Subject** | **Human Computer Interaction** |

## Lab#03

## 1. WIMP Interface: Task Manager (Tkinter)

### Code:

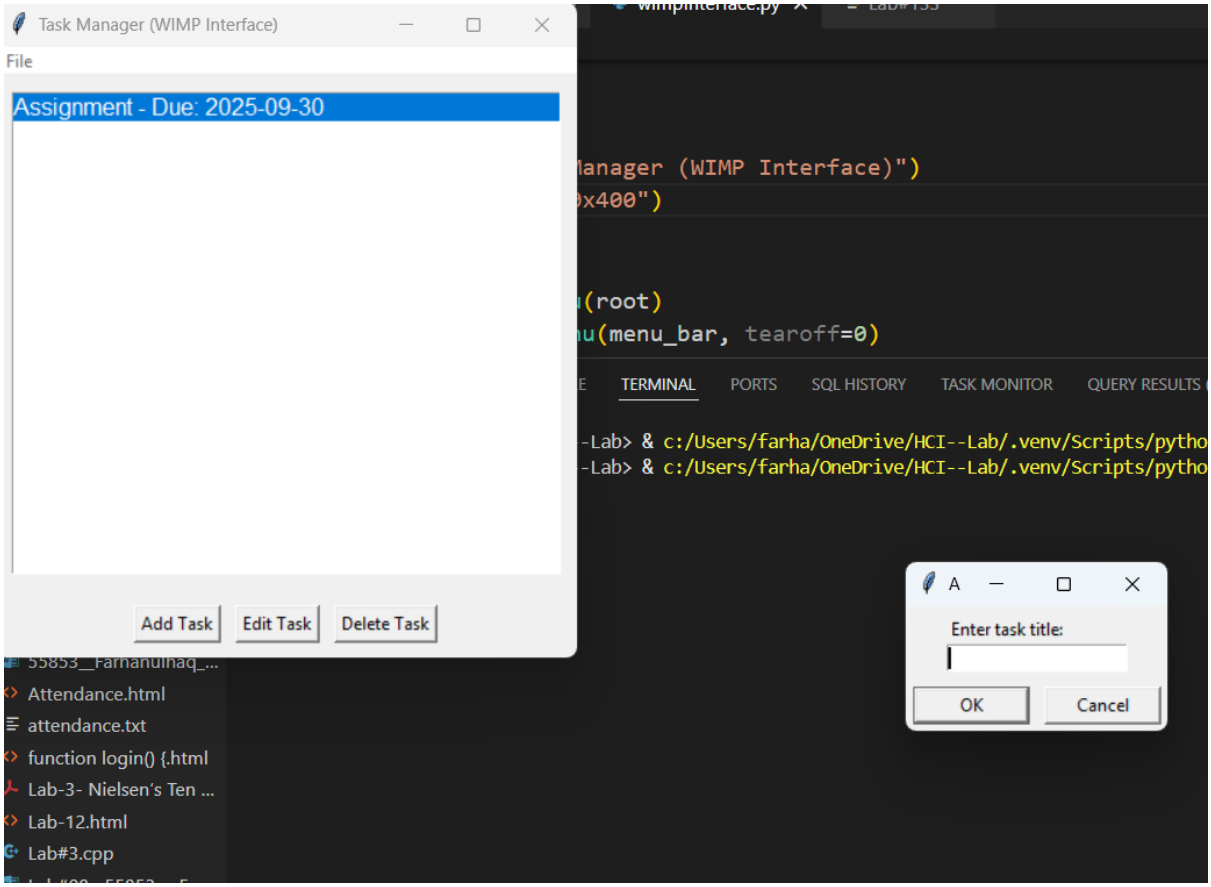
```
import tkinter as tk
from tkinter import simpledialog, messagebox, filedialog
import json

tasks = []

def add_task():
    title = simpledialog.askstring("Add Task", "Enter task title:")
    deadline = simpledialog.askstring("Add Task", "Enter deadline (e.g., 2025-05-10):")
    if title and deadline:
        tasks.append({"title": title, "deadline": deadline})
        refresh_tasks()

def edit_task():
    selected = task_listbox.curselection()
```

```
    if selected:
        index = selected[0]
        task = tasks[index]
        new_title = simpledialog.askstring("Edit Task", "Edit title:",
initialvalue=task["title"])
        new_deadline = simpledialog.askstring("Edit Task", "Edit deadline:",
initialvalue=task["deadline"])
        if new_title and new_deadline:
            tasks[index] = {"title": new_title, "deadline": new_deadline}
            refresh_tasks()

def delete_task():
    selected = task_listbox.curselection()
    if selected:
        index = selected[0]
        tasks.pop(index)
        refresh_tasks()

def refresh_tasks():
    task_listbox.delete(0, tk.END)
    for task in tasks:
        task_listbox.insert(tk.END, f"{task['title']}- Due: {task['deadline']}")

def save_tasks():
    file_path = filedialog.asksaveasfilename(defaultextension=".json")
    if file_path:
        with open(file_path, "w") as f:
            json.dump(tasks, f)
```

```
        messagebox.showinfo("Saved", "Tasks saved successfully.")

def load_tasks():
    file_path = filedialog.askopenfilename(filetypes=[("JSON files", "*.json")])
    if file_path:
        with open(file_path, "r") as f:
            loaded_tasks = json.load(f)
            tasks.clear()
            tasks.extend(loaded_tasks)
            refresh_tasks()

# Setup window
root = tk.Tk()
root.title("Task Manager (WIMP Interface)")
root.geometry("400x400")

# Menu bar
menu_bar = tk.Menu(root)
file_menu = tk.Menu(menu_bar, tearoff=0)
file_menu.add_command(label="Save", command=save_tasks)
file_menu.add_command(label="Load", command=load_tasks)
file_menu.add_separator()
file_menu.add_command(label="Exit", command=root.quit)
menu_bar.add_cascade(label="File", menu=file_menu)
root.config(menu=menu_bar)

# Task list
task_listbox = tk.Listbox(root, font=("Arial", 12))
```

```
task_listbox.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

# Buttons
btn_frame = tk.Frame(root)
btn_frame.pack(pady=10)

tk.Button(btn_frame, text="Add Task", command=add_task).grid(row=0, column=0, padx=5)
tk.Button(btn_frame, text="Edit Task", command=edit_task).grid(row=0, column=1, padx=5)
tk.Button(btn_frame, text="Delete Task", command=delete_task).grid(row=0, column=2, padx=5)

root.mainloop()
```

## Interface:



## 2. Direct Manipulation Enhancements:

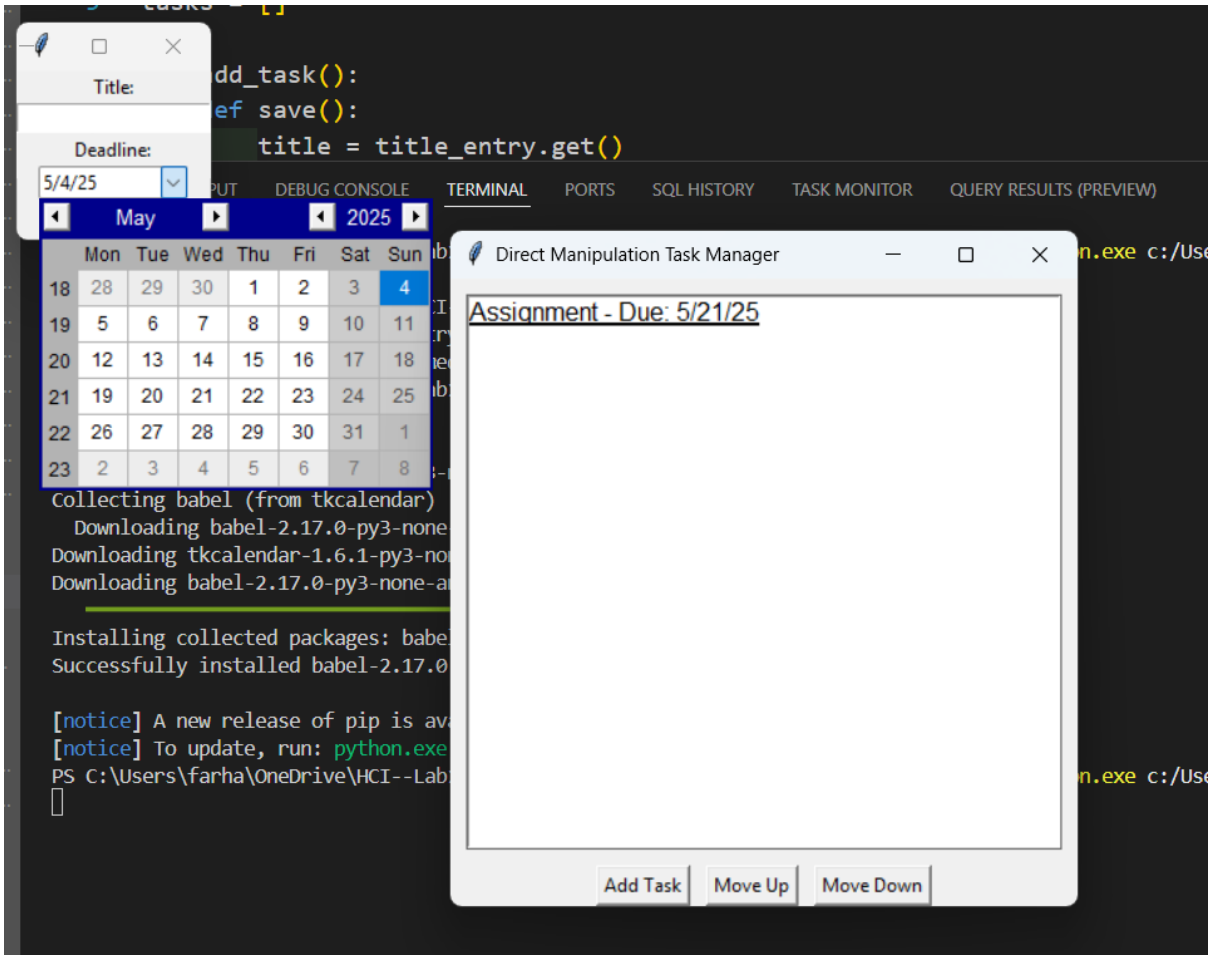
## Code:

```
import tkinter as tk
from tkinter import simpledialog, messagebox
from tkcalendar import DateEntry

tasks = []

def add_task():
    def save():
        title = title_entry.get()
        deadline = date_entry.get()
        if title:
            tasks.append({'title': title, 'deadline': deadline})
            refresh_tasks()
            popup.destroy()

    popup = tk.Toplevel(root)
    popup.title("Add Task")
    tk.Label(popup, text="Title:").pack()
    title_entry = tk.Entry(popup)
    title_entry.pack()
    tk.Label(popup, text="Deadline:").pack()
    date_entry = DateEntry(popup, width=12, background='darkblue',
                foreground='white', borderwidth=2)
    date_entry.pack()
    tk.Button(popup, text="Save", command=save).pack()
```

```
def move_up():
    index = task_listbox.curselection()
    if index and index[0] > 0:
        tasks[index[0] - 1], tasks[index[0]] = tasks[index[0]], tasks[index[0] - 1]
        refresh_tasks()
        task_listbox.selection_set(index[0] - 1)

def move_down():
    index = task_listbox.curselection()
    if index and index[0] < len(tasks) - 1:
        tasks[index[0] + 1], tasks[index[0]] = tasks[index[0]], tasks[index[0] + 1]
        refresh_tasks()
        task_listbox.selection_set(index[0] + 1)

def refresh_tasks():
    task_listbox.delete(0, tk.END)
    for task in tasks:
        task_listbox.insert(tk.END, f"{task['title']} - Due: {task['deadline']}")

root = tk.Tk()
root.title("Direct Manipulation Task Manager")
root.geometry("400x400")

task_listbox = tk.Listbox(root, font=("Arial", 12))
task_listbox.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

btn_frame = tk.Frame(root)
```

```
btn_frame.pack()

tk.Button(btn_frame, text="Add Task", command=add_task).grid(row=0, column=0, padx=5)
tk.Button(btn_frame, text="Move Up", command=move_up).grid(row=0, column=1, padx=5)
tk.Button(btn_frame, text="Move Down", command=move_down).grid(row=0, column=2, padx=5)

root.mainloop()
```

## Output:



## 3.CLI Calculator — Add More Operations

Here we add exponentiation and modulus operations to the CLI calculator:

## Code:

```
def cli_calculator():
    print("Welcome to the Extended CLI Calculator!")
    while True:
        print("\nAvailable Commands: add, subtract, multiply, divide, exponent, modulus, quit")
        command = input("Enter a command: ").strip().lower()
        if command == "quit":
            print("Exiting the calculator. Goodbye!")
            break
        try:
            num1 = float(input("Enter first number: "))
            num2 = float(input("Enter second number: "))
            if command == "add":
                print(f"Result: {num1 + num2}")
            elif command == "subtract":
                print(f"Result: {num1 - num2}")
            elif command == "multiply":
                print(f"Result: {num1 * num2}")
            elif command == "divide":
                print(f"Result: {num1 / num2}")
            elif command == "exponent":
                print(f"Result: {num1 ** num2}")
            elif command == "modulus":
                print(f"Result: {num1 % num2}")
```

```
        else:
            print("Invalid command. Try again.")
    except ValueError:
        print("Invalid input. Please enter numbers.")
    except ZeroDivisionError:
        print("Error: Division by zero.")
cli_calculator()
```

## Output:

```
Welcome to the Extended CLI Calculator!

Available Commands: add, subtract, multiply, divide, exponent, modulus, quit
Enter a command: add
Enter first number: 45
Enter second number: 4
Result: 49.0

Available Commands: add, subtract, multiply, divide, exponent, modulus, quit
Enter a command:
```

---

## 4. Menu-Driven To-Do List — Mark Tasks as Completed

Here we add an option to mark tasks as completed:

## Code:

```
def menu_driven_todo():
    tasks = []  # List to store tasks with completed status
    while True:
        print("\n--- To-Do List Menu ---")
        print("1. Add Task")
        print("2. View Tasks")
        print("3. Mark Task as Completed")
```

```
print("4. Delete Task")
print("5. Quit")

choice = input("Enter your choice: ").strip()

if choice == "1":
    task = input("Enter the task: ")
    tasks.append({"title": task, "completed": False})
    print(f"Task '{task}' added.")

elif choice == "2":
    if tasks:
        print("\nTasks:")
        for i, task in enumerate(tasks, 1):
            status = "✓" if task["completed"] else "✗"
            print(f"{i}. [{status}] {task['title']}")
    else:
        print("No tasks found.")

elif choice == "3":
    if tasks:
        task_num = int(input("Enter the task number to mark as completed: "))
        if 1 <= task_num <= len(tasks):
            tasks[task_num - 1]["completed"] = True
            print(f"Task '{tasks[task_num - 1]['title']}' marked as completed.")
        else:
            print("Invalid task number.")
    else:
```

```
            print("No tasks to mark.")

        elif choice == "4":
            if tasks:
                task_num = int(input("Enter the task number to delete: "))
                if 1 <= task_num <= len(tasks):
                    removed_task = tasks.pop(task_num - 1)
                    print(f"Task '{removed_task['title']}' deleted.")
                else:
                    print("Invalid task number.")
            else:
                print("No tasks to delete.")

        elif choice == "5":
            print("Exiting the to-do list. Goodbye!")
            break

        else:
            print("Invalid choice. Try again.")

menu_driven_todo()
```

**Output:**

Output

```
--- To-Do List Menu ---
1. Add Task
2. View Tasks
3. Mark Task as Completed
4. Delete Task
5. Quit
Enter your choice: 1
Enter the task: Assignment
Task 'Assignment' added.

--- To-Do List Menu ---
1. Add Task
2. View Tasks
3. Mark Task as Completed
4. Delete Task
5. Quit
Enter your choice: 2

Tasks:
1. [X] Assignment

--- To-Do List Menu ---
1. Add Task
2. View Tasks
3. Mark Task as Completed
4. Delete Task
5. Quit
Enter your choice: 5
Exiting the to-do list. Goodbye!
```

## 5.Form-Based Interface — Complete Registration Form

Here we extend the user registration form with more fields and validation:

## Code:

```
def form_based_registration():
    print("--- User Registration Form ---")
    name = input("Enter your name: ")
    email = input("Enter your email: ")
    age = input("Enter your age: ")
    password = input("Create a password: ")

    # Basic Validation
    if not name or not email or not age or not password:
        print("Error: All fields are required.")
        return

    if "@" not in email or "." not in email:
        print("Error: Invalid email format.")
        return

    try:
        age = int(age)
        if age <= 0:
            print("Error: Age must be positive.")
            return
    except ValueError:
        print("Error: Age must be a number.")
        return

    print("\nRegistration Successful!")
```

```
    print(f"Name: {name}")
    print(f"Email: {email}")
    print(f"Age: {age}")
    # Note: For security reasons, we should not print password details.

form_based_registration()
```

**Output**:

```
Output
--- User Registration Form ---
Enter your name: Farhan-Ul-Haq
Enter your email: F.ulhaq@gmail.com
Enter your age: 19
Create a password: Farhan123

Registration Successful!
Name: Farhan-Ul-Haq
Email: F.ulhaq@gmail.com
Age: 19

=== Code Execution Successful ===
```

**THE END**