

**American International University-Bangladesh (AIUB)**

**Faculty of Science & Technology (FST)**

**Department of Computer Science**

**Programming in Python Mid-Term Project Report**

**Summer 2024-2025**

**Section: A**

|  |  |  |  |
| --- | --- | --- | --- |
| **SL #** | **Student Name** | **Student ID** | **Contribution (%)** |
| **1.** | **AZRA MOKARAMMA KOSHIN** | **22-46545-1** | **100%** |

**Project Implementation Detail**

**Introduction**

This project is a simple **To-Do List Manager** created using Python’s Tkinter library.  
It allows users to:

* Add tasks
* Mark tasks as complete
* Remove tasks
* Set task priority (1–5)
* Set task due dates
* Save and load tasks from a file

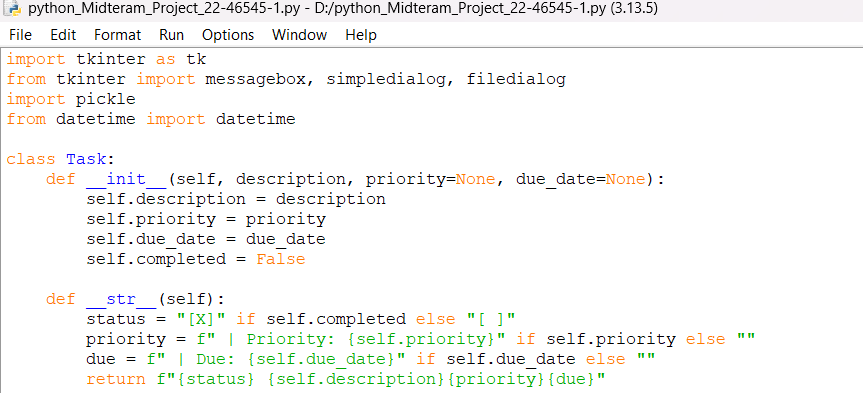
The project is divided into several parts for easy understanding.

**Objectives**

* To create a GUI-based task management system.
* To allow task creation, updating, and deletion.
* To store tasks with priority and due dates.
* To save tasks for future use.
* To make the project easy to use and user-friendly.

**Project Code (Section-wise)**

**Part 1: Import & Task Class**

****

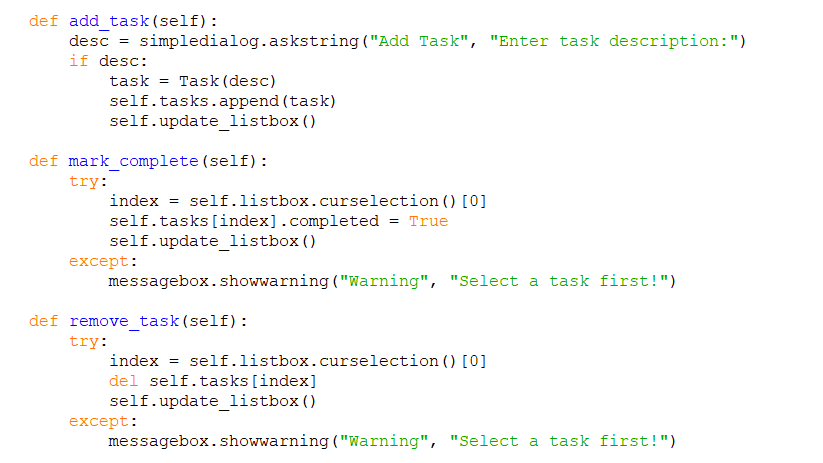
This part imports necessary modules. Task class stores details about each task: description, priority, due date, and status. \_\_str\_\_ defines how a task is displayed in the list.

**Part 2: Main App Setup**

**A computer screen shot of a computer code

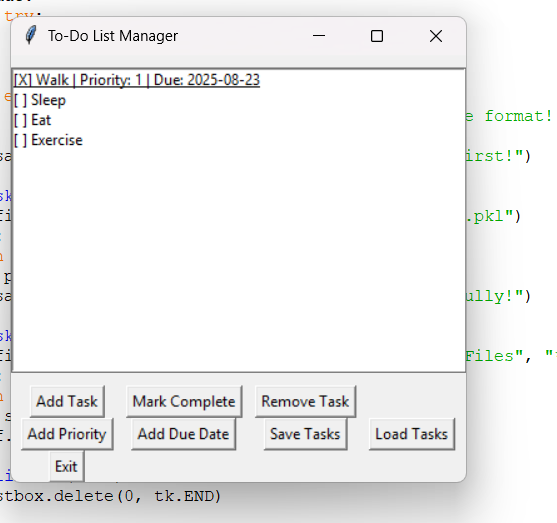
AI-generated content may be incorrect.**This part creates the main window of the application. A Listbox is used to display tasks. Buttons are created for adding, completing, removing, and saving tasks.

**Part 3: Basic Task Functions**

****

add\_task → adds new task. mark\_complete → marks a selected task as completed. remove\_task → deletes a selected task.

Output:



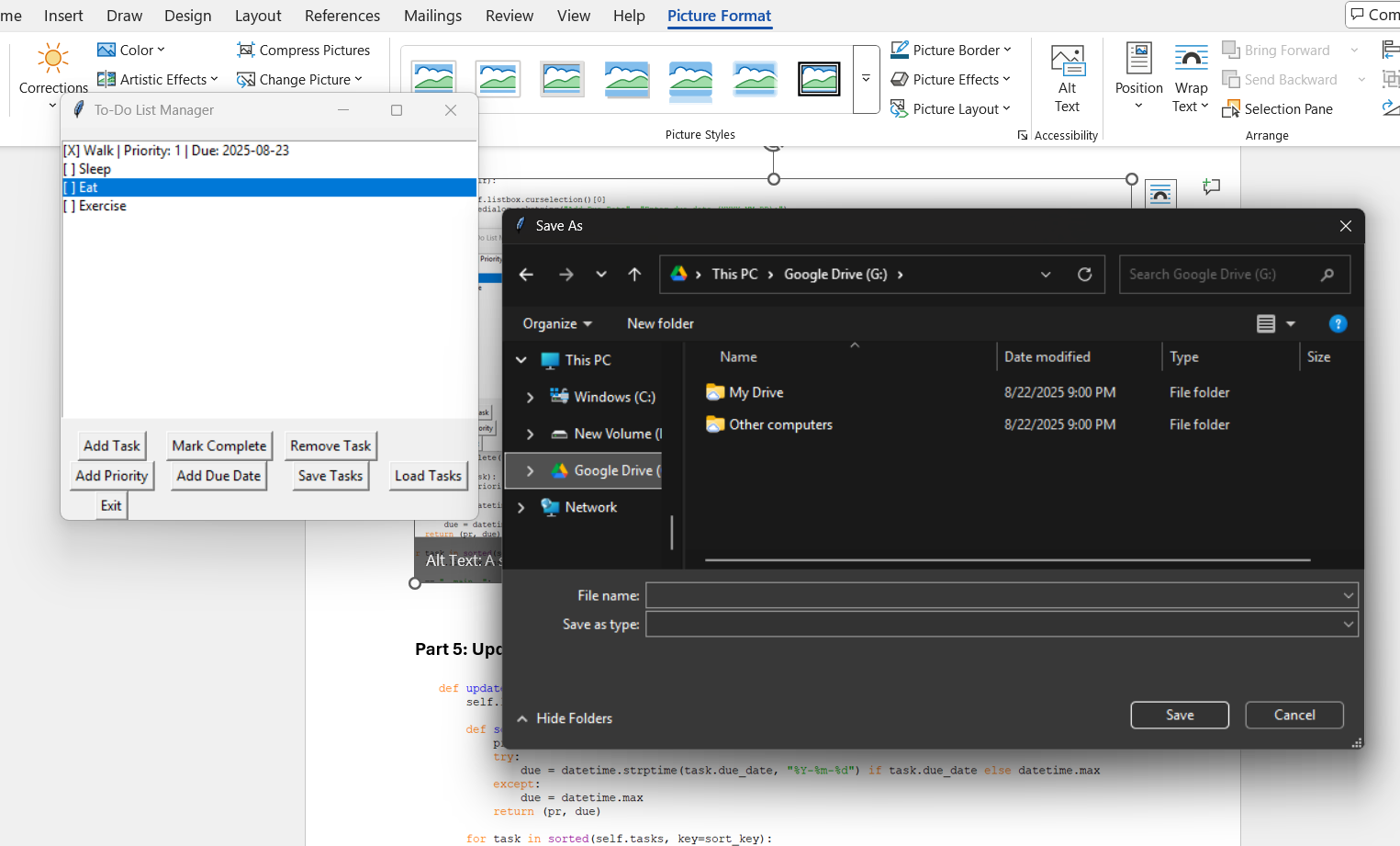
**Part 4: Extra Features**

**A computer screen shot of a program

AI-generated content may be incorrect.**

add\_priority → sets task priority (1–5). add\_due\_date → sets deadline in format YYYY-MM-DD. save\_tasks → saves all tasks in a .pkl file. load\_tasks → loads saved tasks.

**Output:**

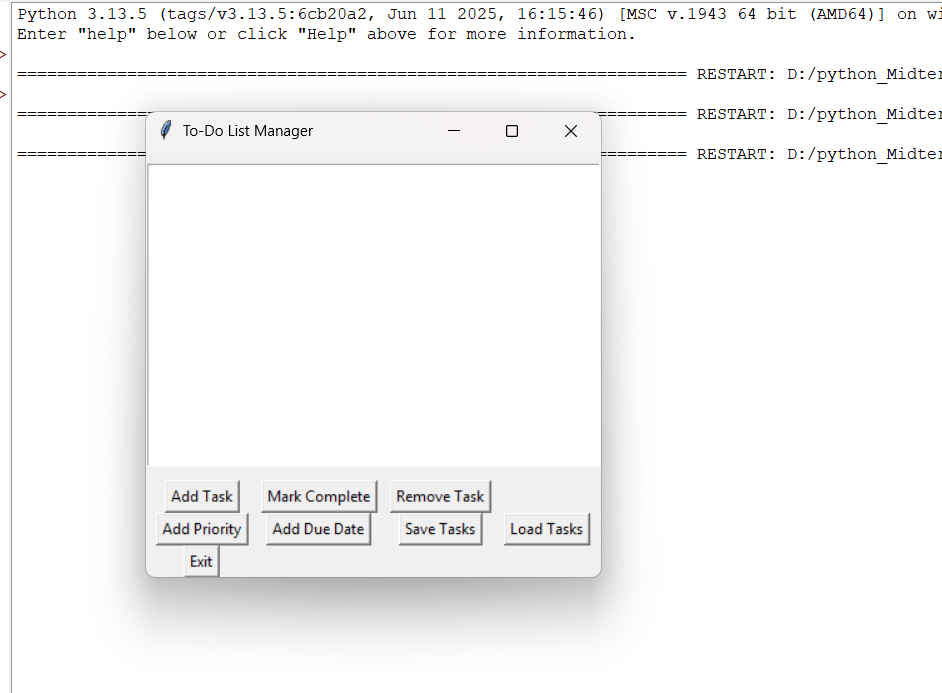
****

**Part 5: Updating List and Running App**

A computer code on a white background

AI-generated content may be incorrect.update\_listbox → clears and updates task list. Sorting is done by priority and then by due date. At the end, program runs the GUI application.

**Output**

When running the program: A **GUI window** appears. Users can add tasks, mark them complete, set priority, set due date, save and load tasks. Tasks are shown in sorted order.

**Full-Code:**

import tkinter as tk

from tkinter import messagebox, simpledialog, filedialog

import pickle

from datetime import datetime

class Task:

def \_\_init\_\_(self, description, priority=None, due\_date=None):

self.description = description

self.priority = priority

self.due\_date = due\_date

self.completed = False

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

status = "[X]" if self.completed else "[ ]"

priority = f" | Priority: {self.priority}" if self.priority else ""

due = f" | Due: {self.due\_date}" if self.due\_date else ""

return f"{status} {self.description}{priority}{due}"

class ToDoApp:

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

self.root = root

self.root.title("To-Do List Manager")

self.tasks = []

self.listbox = tk.Listbox(root, width=60, height=15)

self.listbox.pack(pady=10)

btn\_frame = tk.Frame(root)

btn\_frame.pack()

tk.Button(btn\_frame, text="Add Task", command=self.add\_task).grid(row=0, column=0, padx=5)

tk.Button(btn\_frame, text="Mark Complete", command=self.mark\_complete).grid(row=0, column=1, padx=5)

tk.Button(btn\_frame, text="Remove Task", command=self.remove\_task).grid(row=0, column=2, padx=5)

tk.Button(btn\_frame, text="Add Priority", command=self.add\_priority).grid(row=1, column=0, padx=5)

tk.Button(btn\_frame, text="Add Due Date", command=self.add\_due\_date).grid(row=1, column=1, padx=5)

tk.Button(btn\_frame, text="Save Tasks", command=self.save\_tasks).grid(row=1, column=2, padx=5)

tk.Button(btn\_frame, text="Load Tasks", command=self.load\_tasks).grid(row=1, column=3, padx=5)

tk.Button(btn\_frame, text="Exit", command=root.quit).grid(row=2, column=0, padx=5)

self.update\_listbox()

def add\_task(self):

desc = simpledialog.askstring("Add Task", "Enter task description:")

if desc:

task = Task(desc)

self.tasks.append(task)

self.update\_listbox()

def mark\_complete(self):

try:

index = self.listbox.curselection()[0]

self.tasks[index].completed = True

self.update\_listbox()

except:

messagebox.showwarning("Warning", "Select a task first!")

def remove\_task(self):

try:

index = self.listbox.curselection()[0]

del self.tasks[index]

self.update\_listbox()

except:

messagebox.showwarning("Warning", "Select a task first!")

def add\_priority(self):

try:

index = self.listbox.curselection()[0]

priority = simpledialog.askinteger("Add Priority", "Enter priority (1-5):")

if priority and 1 <= priority <= 5:

self.tasks[index].priority = priority

self.update\_listbox()

else:

messagebox.showerror("Error", "Priority must be between 1 and 5.")

except:

messagebox.showwarning("Warning", "Select a task first!")

def add\_due\_date(self):

try:

index = self.listbox.curselection()[0]

due = simpledialog.askstring("Add Due Date", "Enter due date (YYYY-MM-DD):")

if due:

try:

datetime.strptime(due, "%Y-%m-%d")

self.tasks[index].due\_date = due

self.update\_listbox()

except:

messagebox.showerror("Error", "Invalid date format! Use YYYY-MM-DD.")

except:

messagebox.showwarning("Warning", "Select a task first!")

def save\_tasks(self):

file = filedialog.asksaveasfilename(defaultextension=".pkl")

if file:

with open(file, "wb") as f:

pickle.dump(self.tasks, f)

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

def load\_tasks(self):

file = filedialog.askopenfilename(filetypes=[("Pickle Files", "\*.pkl")])

if file:

with open(file, "rb") as f:

self.tasks = pickle.load(f)

self.update\_listbox()

def update\_listbox(self):

self.listbox.delete(0, tk.END)

def sort\_key(task):

pr = task.priority if task.priority else 9999

try:

due = datetime.strptime(task.due\_date, "%Y-%m-%d") if task.due\_date else datetime.max

except:

due = datetime.max

return (pr, due)

for task in sorted(self.tasks, key=sort\_key):

self.listbox.insert(tk.END, str(task))

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

root = tk.Tk()

app = ToDoApp(root)

root.mainloop()