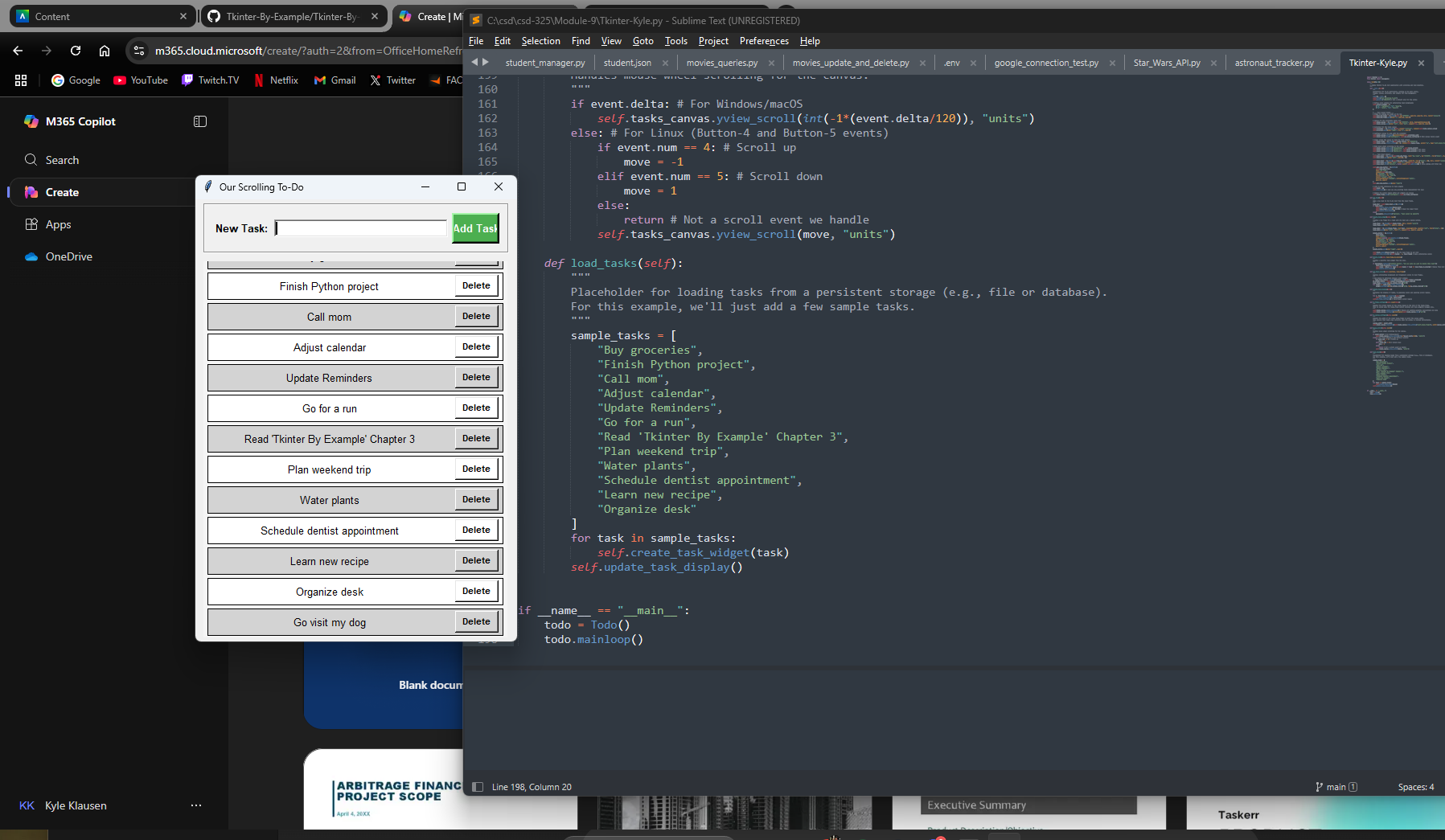
Kyle Klausen

CSD-325

07/10/25

Module10\_2

**First set of screenshots for successful run and adding tasks:**



Code:

**# Author: Kyle Klausen**

**# Assignment: Module10\_2**

**# Date: 07/10/25**

**# Description: A simple code which creates a to-do list that has functions to help improve functionality**

import tkinter as tk

from tkinter import messagebox

class Todo(tk.Tk):

"""

A simple Tkinter To-Do list application with scrolling and task deletion.

"""

def \_\_**init\_\_**(self):

super().**init**()

self.title("Klausen-ToDo")

self.geometry("400x550")

# --- Menu Bar ---  
 menu\_bar = tk.Menu(self, bg="purple", fg="white")  
 file\_menu = tk.Menu(menu\_bar, tearoff=0, bg="gold", fg="black")  
 file\_menu.add\_command(label="Exit", command=self.quit)  
 menu\_bar.add\_cascade(label="File", menu=file\_menu)  
 self.config(menu=menu\_bar)  
  
 # Instruction Label  
 instruction\_label = tk.Label(self, text="Right-click a task to delete it",  
 bg="#fff8dc", fg="black", font=("Inter", 10, "italic"))  
 instruction\_label.pack(pady=(5, 0))  
  
 # --- Colors for alternating tasks ---  
 self.colour\_schemes = [  
 {"bg": "lightblue", "fg": "black"},  
 {"bg": "lightgrey", "fg": "black"}  
 ]  
  
 # --- Task Input Frame ---  
 self.task\_add\_frame = tk.Frame(self, bg="#f0f0f0", padx=10, pady=10, bd=2, relief="groove")  
 self.task\_add\_frame.pack(fill="x", padx=10, pady=5)  
  
 self.task\_label = tk.Label(self.task\_add\_frame, text="New Task:", bg="#f0f0f0", font=("Inter", 10, "bold"))  
 self.task\_label.pack(side="left", padx=(0, 5))  
  
 self.task\_input = tk.Entry(self.task\_add\_frame, width=30, font=("Inter", 10), bd=2, relief="sunken")  
 self.task\_input.pack(side="left", fill="x", expand=True, padx=(0, 5))  
 self.task\_input.bind("<Return>", lambda event: self.add\_task())  
  
 self.add\_task\_button = tk.Button(  
 self.task\_add\_frame,  
 text="Add Task",  
 command=self.add\_task,  
 bg="#4CAF50", fg="white",  
 font=("Inter", 10, "bold"),  
 relief="raised", bd=3,  
 activebackground="#45a049", activeforeground="white",  
 padx=10, pady=5  
 )  
 self.add\_task\_button.pack(side="right")  
  
 # --- Task Display Area ---  
 self.tasks\_canvas = tk.Canvas(self, bg="#f5f5f5", bd=0, highlightthickness=0)  
 self.tasks\_canvas.pack(side="left", fill="both", expand=True, padx=10, pady=5)  
  
 self.scrollbar = tk.Scrollbar(self, orient="vertical", command=self.tasks\_canvas.yview)  
 self.scrollbar.pack(side="right", fill="y", pady=5)  
  
 self.tasks\_canvas.configure(yscrollcommand=self.scrollbar.set)  
 self.tasks\_canvas.bind('<Configure>', self.on\_canvas\_configure)  
  
 self.tasks\_frame = tk.Frame(self.tasks\_canvas, bg="#f5f5f5")  
 self.tasks\_canvas.create\_window((0, 0), window=self.tasks\_frame, anchor="nw", tags="self.tasks\_frame")  
  
 self.tasks\_canvas.bind\_all("<MouseWheel>", self.mouse\_scroll)  
 self.tasks\_canvas.bind\_all("<Button-4>", self.mouse\_scroll) # For Linux  
 self.tasks\_canvas.bind\_all("<Button-5>", self.mouse\_scroll)  
  
 self.tasks = []  
 self.load\_tasks()  
  
 self.tasks\_frame.bind("<Configure>", self.on\_frame\_configure)  
  
def add\_task(self):  
 task\_text = self.task\_input.get().strip()  
 if task\_text:  
 self.create\_task\_widget(task\_text)  
 self.task\_input.delete(0, tk.END)  
 self.update\_task\_display()  
 else:  
 messagebox.showwarning("Warning", "Task cannot be empty!")  
  
def create\_task\_widget(self, text):  
 task\_frame = tk.Frame(self.tasks\_frame, bd=1, relief="solid")  
 task\_frame.pack(fill="x", padx=5, pady=2)  
  
 task\_label = tk.Label(task\_frame, text=text, wraplength=250, justify="left", font=("Inter", 10))  
 task\_label.pack(side="left", fill="x", expand=True, padx=5, pady=5)  
  
 # Right-click to delete task  
 task\_frame.bind("<Button-3>", lambda event: self.delete\_task(task\_frame))  
 task\_label.bind("<Button-3>", lambda event: self.delete\_task(task\_frame))  
  
 self.tasks.append(task\_frame)  
 self.set\_task\_colour(len(self.tasks) - 1, task\_frame)  
  
def delete\_task(self, task\_frame\_to\_delete):  
 if messagebox.askyesno("Confirm Delete", "Are you sure you want to delete this task?"):  
 task\_frame\_to\_delete.destroy()  
 self.tasks = [task for task in self.tasks if task != task\_frame\_to\_delete]  
 self.update\_task\_display()  
  
def set\_task\_colour(self, position, task\_frame):  
 \_, style\_index = divmod(position, len(self.colour\_schemes))  
 scheme = self.colour\_schemes[style\_index]  
 task\_frame.configure(bg=scheme["bg"])  
 for widget in task\_frame.winfo\_children():  
 widget.configure(bg=scheme["bg"], fg=scheme["fg"])  
  
def update\_task\_display(self):  
 for i, task\_frame in enumerate(self.tasks):  
 self.set\_task\_colour(i, task\_frame)  
 self.on\_frame\_configure()  
  
def on\_frame\_configure(self, event=None):  
 self.tasks\_canvas.update\_idletasks()  
 self.tasks\_canvas.configure(scrollregion=self.tasks\_canvas.bbox("all"))  
  
def on\_canvas\_configure(self, event):  
 canvas\_width = event.width  
 self.tasks\_canvas.itemconfig(self.tasks\_canvas.find\_withtag("self.tasks\_frame"), width=canvas\_width)  
  
def mouse\_scroll(self, event):  
 if event.delta:  
 self.tasks\_canvas.yview\_scroll(int(-1 \* (event.delta / 120)), "units")  
 else:  
 move = -1 if event.num == 4 else 1 if event.num == 5 else 0  
 self.tasks\_canvas.yview\_scroll(move, "units")  
  
def load\_tasks(self):  
 sample\_tasks = [  
 "Buy groceries",  
 "Finish Python project",  
 "Call mom",  
 "Adjust calendar",  
 "Update Reminders",  
 "Go for a run",  
 "Read 'Tkinter By Example' Chapter 3",  
 "Plan weekend trip",  
 "Water plants",  
 "Schedule dentist appointment",  
 "Learn new recipe",  
 "Organize desk"  
 ]  
 for task in sample\_tasks:  
 self.create\_task\_widget(task)  
 self.update\_task\_display()

if **name** == "**main**": todo = Todo() todo.mainloop()

Screenshots after additional modifications to code:

