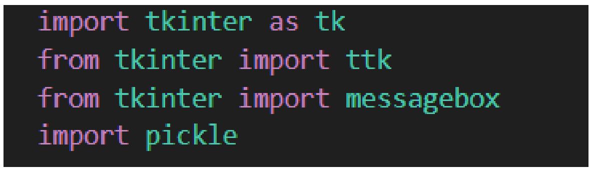
I created the Task Manager in order to provide users with a simple yet efficient tool and an attractive interface design ,the application boasts a clean and modern interface, leveraging the Tkinter library for a consistent and visually appealing user experience.to manage their personal tasks. The application allows users to add, remove, and mark tasks as completed, providing a streamlined approach to task organization.

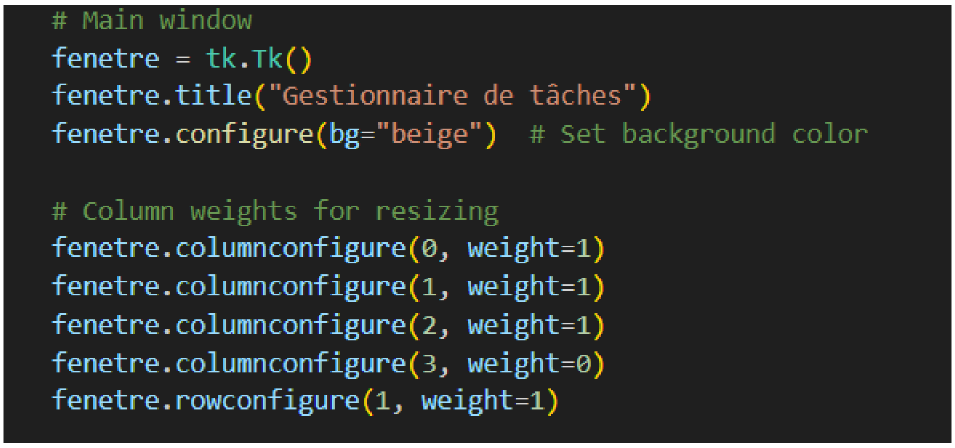
In a fast-paced digital environment, managing personal tasks efficiently is crucial. This program offers a user-friendly interface, empowering users to organize their tasks effortlessly.

I started by importing all the modules that will be usefull in the script to have a better graphic design and to save some tasks as file.



```python
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import pickle
```

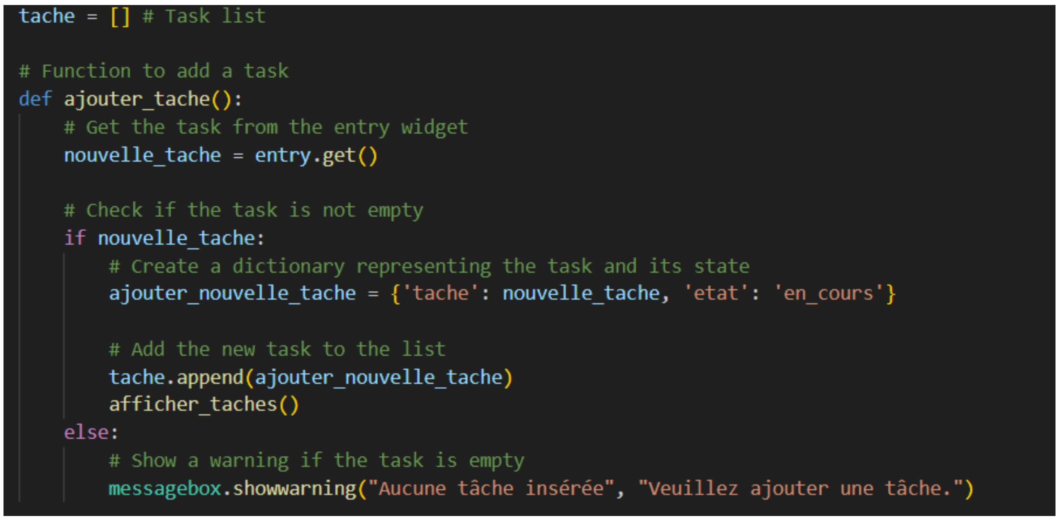
Then there is the creation of the main window, the configuration of background and the columns in case the user wants to change the size of the window.



```python
# Main window
fenetre = tk.Tk()
fenetre.title("Gestionnaire de tâches")
fenetre.configure(bg="beige")  # Set background color

# Column weights for resizing
fenetre.columnconfigure(0, weight=1)
fenetre.columnconfigure(1, weight=1)
fenetre.columnconfigure(2, weight=1)
fenetre.columnconfigure(3, weight=0)
fenetre.rowconfigure(1, weight=1)
```

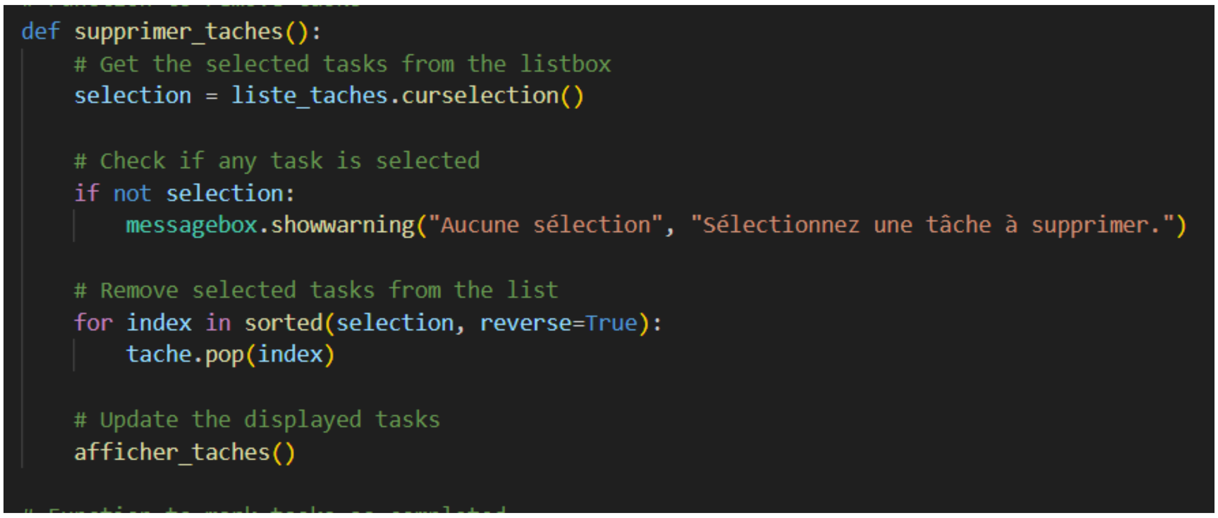
Now we shift to the tasking management by differents functions which are :

Add new tasks by entering task details into the input field.



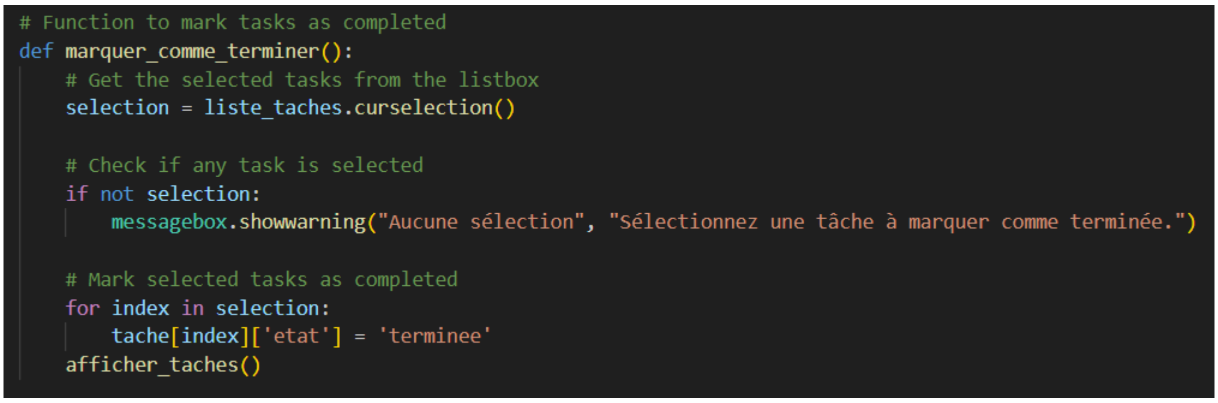
```python
tache = [] # Task list

# Function to add a task
def ajouter_tache():
    # Get the task from the entry widget
    nouvelle_tache = entry.get()

    # Check if the task is not empty
    if nouvelle_tache:
        # Create a dictionary representing the task and its state
        ajouter_nouvelle_tache = {'tache': nouvelle_tache, 'etat': 'en_cours'}

        # Add the new task to the list
        tache.append(ajouter_nouvelle_tache)
        afficher_taches()
    else:
        # Show a warning if the task is empty
        messagebox.showwarning("Aucune tâche insérée", "Veuillez ajouter une tâche.")
```

Delete tasks selected by the user.



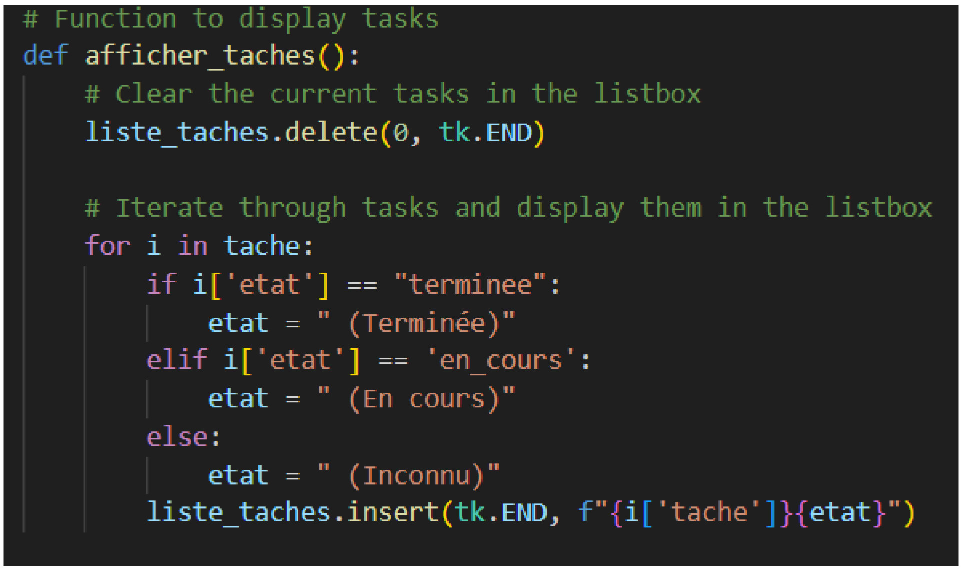
```python
def supprimer_taches():
    # Get the selected tasks from the listbox
    selection = liste_taches.curselection()

    # Check if any task is selected
    if not selection:
        messagebox.showwarning("Aucune sélection", "Sélectionnez une tâche à supprimer.")

    # Remove selected tasks from the list
    for index in sorted(selection, reverse=True):
        tache.pop(index)

    # Update the displayed tasks
    afficher_taches()
```

Mark tasks as completed by selecting them.



```python
# Function to mark tasks as completed
def marquer_comme_terminer():
    # Get the selected tasks from the listbox
    selection = liste_taches.curselection()

    # Check if any task is selected
    if not selection:
        messagebox.showwarning("Aucune sélection", "Sélectionnez une tâche à marquer comme terminée.")

    # Mark selected tasks as completed
    for index in selection:
        tache[index]['etat'] = 'terminee'
    afficher_taches()
```

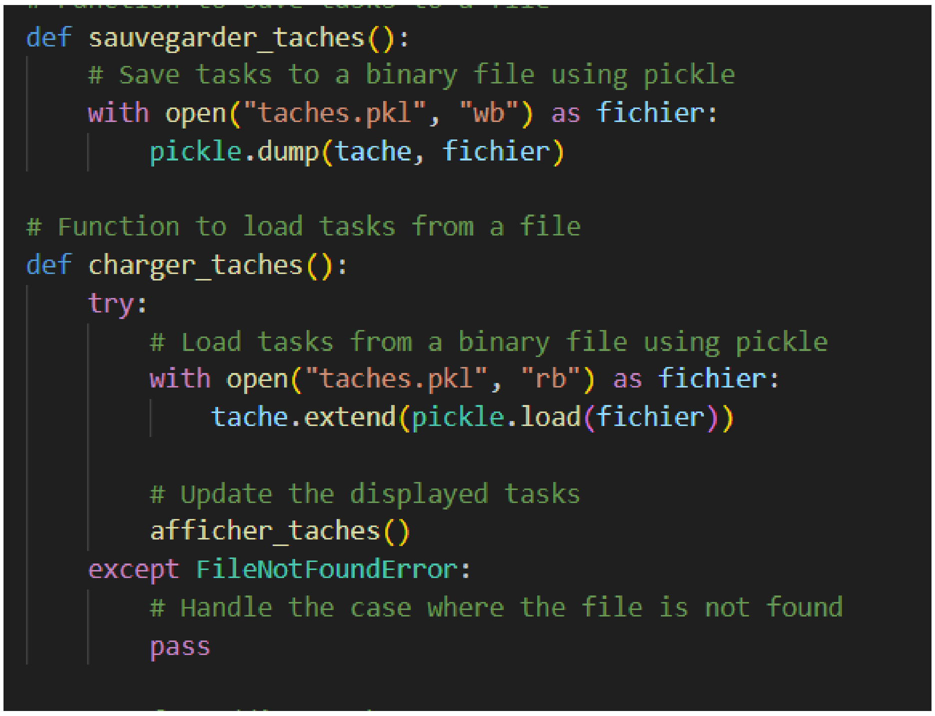
Display the tasks and their state.



```python
# Function to display tasks
def afficher_taches():
    # Clear the current tasks in the listbox
    liste_taches.delete(0, tk.END)

    # Iterate through tasks and display them in the listbox
    for i in tache:
        if i['etat'] == "terminee":
            etat = " (Terminée)"
        elif i['etat'] == 'en_cours':
            etat = " (En cours)"
        else:
            etat = " (Inconnu)"
        liste_taches.insert(tk.END, f"{i['tache']}{etat}")
```

Save the tasks you added as a file.

Load the tasks saved as files before.

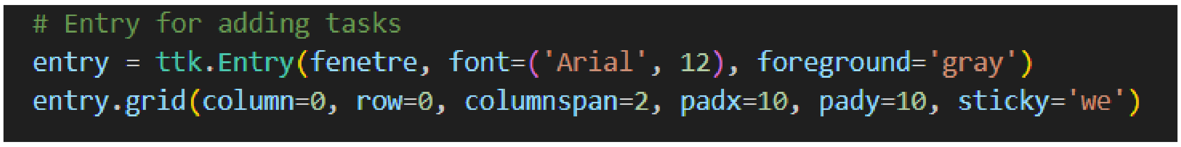


```python
def sauvegarder_taches():
    # Save tasks to a binary file using pickle
    with open("taches.pkl", "wb") as fichier:
        pickle.dump(tache, fichier)

# Function to load tasks from a file
def charger_taches():
    try:
        # Load tasks from a binary file using pickle
        with open("taches.pkl", "rb") as fichier:
            tache.extend(pickle.load(fichier))

        # Update the displayed tasks
        afficher_taches()
    except FileNotFoundError:
        # Handle the case where the file is not found
        pass
```

As you saw there are warning messages to guide the users when attemptiong actions without selecting tasks, enhancing the overall user experience.

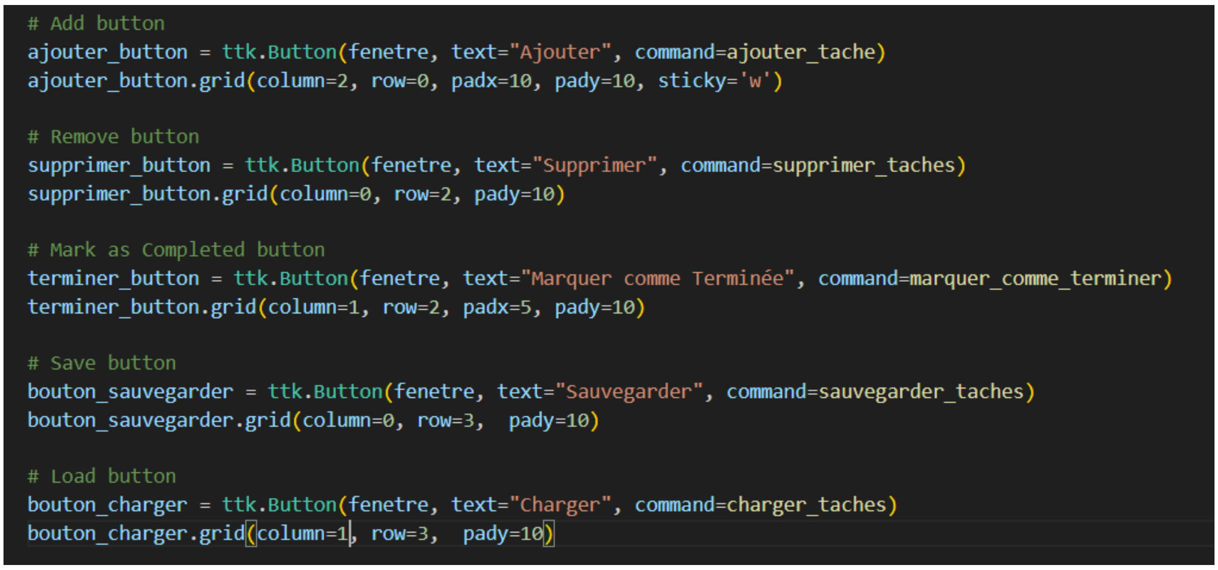
Then there are all the widget that will have a huge impact on the user experience,

The entry for adding tasks.



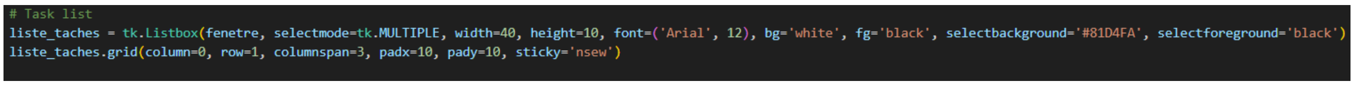
```python
# Entry for adding tasks
entry = ttk.Entry(fenetre, font=('Arial', 12), foreground='gray')
entry.grid(column=0, row=0, columnspan=2, padx=10, pady=10, sticky='we')
```

You can see all the buttons that will prompt the functions we saw earlier.



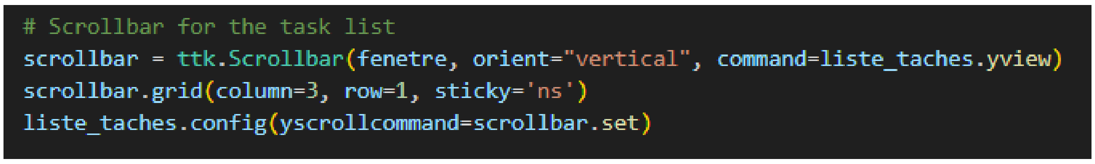
```python
# Add button
ajouter_button = ttk.Button(fenetre, text="Ajouter", command=ajouter_tache)
ajouter_button.grid(column=2, row=0, padx=10, pady=10, sticky='w')

# Remove button
supprimer_button = ttk.Button(fenetre, text="Supprimer", command=supprimer_taches)
supprimer_button.grid(column=0, row=2, pady=10)

# Mark as Completed button
terminer_button = ttk.Button(fenetre, text="Marquer comme Terminée", command=marquer_comme_terminer)
terminer_button.grid(column=1, row=2, padx=5, pady=10)

# Save button
bouton_sauvegarder = ttk.Button(fenetre, text="Sauvegarder", command=sauvegarder_taches)
bouton_sauvegarder.grid(column=0, row=3,  pady=10)

# Load button
bouton_charger = ttk.Button(fenetre, text="Charger", command=charger_taches)
bouton_charger.grid(column=1, row=3,  pady=10)
```

The listbox to see all the tasks.

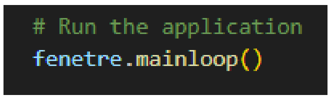


```python
# Task list
liste_taches = tk.Listbox(fenetre, selectmode=tk.MULTIPLE, width=40, height=10, font=('Arial', 12), bg='white', fg='black', selectbackground='#81D4FA', selectforeground='black')
liste_taches.grid(column=0, row=1, columnspan=3, padx=10, pady=10, sticky='nsew')
```

And to finish the scrollbar if there is a lot of tasks in your file and the script that allow teh window to always be open.



```python
# Scrollbar for the task list
scrollbar = ttk.Scrollbar(fenetre, orient="vertical", command=liste_taches.yview)
scrollbar.grid(column=3, row=1, sticky='ns')
liste_taches.config(yscrollcommand=scrollbar.set)
```



```python
# Run the application
fenetre.mainloop()
```