**TO-DO LIST Uygulaması**

**Amaç**

Bu program, kullanıcıların günlük görevlerini takip etmelerini sağlayan bir TO-DO LIST uygulamasıdır. Kullanıcılar, uygulama üzerinden görev ekleyebilir, silme işlemleri yapabilir ve kaydedilmiş görev listelerini yönetebilirler.

**Kod Açıklamaları**

**1. Modül İçe Aktarma**

from tkinter import \*  
from tkinter.font import Font  
from tkinter import filedialog, messagebox  
import pickle

**2. Listbox Seçim İşlemleri**

def on\_select(event):  
 *""" Listbox'ta yapılan seçimleri takip etmek için kullanılır. Seçilen görevin indeksini ve içeriğini konsola yazdırır. """* selected\_index = my\_list.curselection()  
 if selected\_index:  
 print(f"Selected Index: {selected\_index[0]}")  
 selected\_task = my\_list.get(selected\_index)  
 print(f"Selected Task: {selected\_task}")

**3. Görev Silme İşlemi**

def delete\_item():  
 *""" Listbox'ta seçili olan görevi siler. """* my\_list.delete(ANCHOR)

**4. Görev Tamamlama İşlemi**

def cross\_off\_item():  
 my\_list.itemconfig(  
 my\_list.curselection(),  
 fg="#dedede"  
 )  
 my\_list.selection\_clear(0, END)

**5. Görevi Tamamlanmamış Olarak İşaretleme İşlemi**

def uncross\_item():  
 my\_list.itemconfig(  
 my\_list.curselection(),  
 fg="#30001a"  
 )  
 my\_list.selection\_clear(0, END)

**6. Tamamlanmış Görevleri Silme İşlemi**

def delete\_crossed\_items():  
 reversed\_indices = reversed(range(my\_list.size()))  
 for index in reversed\_indices:  
 if my\_list.itemcget(index, "fg") == "#dedede":  
 my\_list.delete(index)

**7. Görev Ekleme İşlemi**

Kullanıcının girdiği yeni görevi listeye ekler. Boş giriş durumunda uyarı mesajı gösterir.

def add\_item():  
 new\_task = my\_entry.get().strip() # Girişten boşlukları temizleyerek al  
 if new\_task: # Eğer görev boş değilse  
 my\_list.insert(END, new\_task)  
 my\_entry.delete(0, END)  
 else:  
 # Boş giriş eklenemez uyarısı  
 messagebox.showwarning("Uyarı", "Boş görev eklenemez!")

**8. Dosya İşlemleri Fonksiyonları**

**a. Dosya Kaydetme Fonksiyonu (save\_list)**

**Bu fonksiyon, TO-DO listesini belirtilen bir dosyaya kaydetmekten sorumludur.**

* **file\_name: Kullanıcının dosyayı kaydetmek istediği konumu ve dosya adını belirlemek için dosya iletişim kutusunu açar.**
* **Eğer kullanıcı bir dosya adı belirlerse:**
  + **.json uzantısı yoksa, dosya adına .json uzantısı ekler.**
  + **TO-DO listesindeki bitirilmiş görevleri (gri renkte olanları) temizler.**
  + **TO-DO listesini alır ve belirtilen dosyaya pickled formatında kaydeder.**
  + **Veri kaydetme işlemi başarılıysa herhangi bir hata mesajı göstermez.**
  + **Eğer bir hata oluşursa, try-except bloğu içindeki Exception türündeki hataları yakalar ve kullanıcıya bir hata mesajı gösterir.**
* def save\_list():  
   file\_name = filedialog.asksaveasfilename(  
   initialdir="C:python\_odev\170423822\_1\_ipekakpinar\data",  
   title="Dosyayı Kaydet",  
   filetypes=(  
   ("Dat Dosyaları", "\*.json"),  
   ("Tüm Dosyalar", "\*.\*")  
   )  
   )  
    
   if file\_name:  
   try:  
   if not file\_name.endswith(".json"):  
   file\_name = f'{file\_name}.json'  
    
   count = 0  
    
   while count < my\_list.size():  
   if my\_list.itemcget(count, "fg") == "#dedede":  
   my\_list.delete(my\_list.index(count))  
   else:  
   count += 1  
    
   task = my\_list.get(0, END)  
    
   with open(file\_name, 'wb') as output\_file:  
   pickle.dump(task, output\_file)  
    
   except Exception as e:  
   # Veri kaydetme hatası uyarısı  
   messagebox.showerror("Hata", f"Veri kaydedilirken bir hata oluştu:\n{str(e)}")

**b. Dosya Açma Fonksiyonu (open\_list)**

**Bu fonksiyon, kaydedilmiş bir TO-DO listesini açmaktan sorumludur.**

* **file\_name: Kullanıcının açmak istediği dosyayı seçmesi için dosya iletişim kutusunu açar.**
* **Eğer kullanıcı bir dosya seçerse:**
  + **TO-DO listesini temizler.**
  + **Seçilen dosyayı açar ve içindeki görevleri yükler.**
  + **Yüklenen görevleri TO-DO listesine ekler.**
* def open\_list():  
   file\_name = filedialog.askopenfilename(  
   initialdir="C:\python\_odev\170423822\_1\_ipekakpinar\data",  
   title="Open File",  
   filetypes=(  
   ("JSON Files", "\*.json"),  
   ("All Files", "\*.\*")  
   )  
   )

**c. Listeyi Temizleme Fonksiyonu (clear\_list)**

**Bu fonksiyon, TO-DO listesini temizlemekten sorumludur. Yani, listedeki tüm görevleri siler.**

* def clear\_list():  
   my\_list.delete(0,END)

**Formun Üstü**

**9. Ana Pencere ve Arayüz Tasarımı**

# Ana Tkinter penceresi  
root = Tk()  
root.title('TO-DO LIST')  
root.geometry("500x650")  
root.configure(bg="#FFE5EC")  
  
# Özel font ayarları  
my\_font = Font(  
 family="Sagoe UI",  
 size=15,  
 slant="italic"  
)  
  
  
# Arayüzdeki başlıklar  
Label(root, text="TO-DO LIST", font=("Cooper Siyah", 25, "bold"), bg="#FB6F92", fg="#FFE5EC", bd=2,  
 relief="groove").place(x=10, y=10)  
Label(root, text="My Tasks:", font=my\_font, bg="#FF8FAB", fg="#FFE5EC", bd=2, relief="groove").place(x=10, y=60)

**10. Listbox ve Scrollbar**

# Frame oluşturma  
my\_frame = Frame(root)  
my\_frame.place(x=10, y=100) # Adjust coordinates  
  
# Listbox oluşturma  
my\_list = Listbox(my\_frame,  
 font=my\_font,  
 width=42,  
 height=10,  
 fg="#30001a",  
 bd=2,  
 relief="groove",  
 highlightthickness=0,  
 selectbackground="#FF8FAB",  
 activestyle="none")  
  
# Scrollbar oluşturma  
my\_scrollbar = Scrollbar(my\_frame, orient=VERTICAL, command=my\_list.yview)  
my\_scrollbar.pack(side=RIGHT, fill=Y)  
my\_list.config(yscrollcommand=my\_scrollbar.set)  
  
my\_list.pack(side=LEFT, fill=BOTH) #listeyi sola yerleştirme  
  
my\_list.bind("<<ListboxSelect>>", on\_select)

**11. Butonlar ve Menü**

# Butonlar ve frame'ler ekleniyor  
button\_frame1 = Frame(root, bg="#FFE5EC")  
button\_frame1.place(x=10, y=360)  
  
button\_frame2 = Frame(root, bg="#FFE5EC")  
button\_frame2.place(x=10, y=400)  
  
button\_frame3 = Frame(root, bg="#FFE5EC")  
button\_frame3.place(x=10, y=570)

**12. Diğer Fonksiyonlar ve Giriş Kutusu**

# Butonlar oluşturuluyor  
delete\_button = Button(button\_frame1, text="Delete Task", bg="#FF8FAB", bd=2, relief="groove", command=delete\_item)  
cross\_off\_button = Button(button\_frame1, text="Done", bg="#FF8FAB", bd=2, relief="groove", command=cross\_off\_item)  
uncross\_button = Button(button\_frame1, text="Not Done Yet", bg="#FF8FAB", bd=2, relief="groove", command=uncross\_item)  
delete\_crossed\_button = Button(button\_frame2, text="Clear Finished Tasks", bg="#FF8FAB", bd=2, relief="groove", command=delete\_crossed\_items)  
add\_button = Button(button\_frame3, text="Add Task", bg="#FF8FAB", bd=2, relief="groove", command=add\_item)  
  
space\_label = Label(button\_frame1,  
 text=" ",  
 bg="#FFE5EC")  
delete\_button.pack(side=LEFT)  
space\_label.pack(side=LEFT)  
uncross\_button.pack(side=RIGHT, padx=20)  
cross\_off\_button.pack(side=RIGHT)  
delete\_crossed\_button.pack(side=LEFT, ipadx=175)  
add\_button.pack(side=LEFT)  
  
  
# "Add Tasks" etiketi ekleniyor  
Label(root, text="Add Tasks:", font=my\_font, bg="#FF8FAB", fg="#FFE5EC", bd=2, relief="groove").place(x=10, y=480)  
  
  
# Entry kutusu ekleniyor  
my\_entry = Entry(root, font=(my\_font, 26), width=25)  
my\_entry.place(x=10, y=520)  
  
  
# Tkinter ana döngüsü başlatılıyor  
root.mainloop()