**PYTHON TO DO LİST UYGUAMASI**

HAZIRLAYAN : SERPİL ÇOBANLAR

NUMARA : 170422055

Genel bakış :

Projem insanların günlük hayattaki planlamalarını kolaylaştırmak amacıyla yapılan bir **To-Do-List** uygulaması. Bu uygulama içerisinde görev ekleme, teslim tarihi seçme ve bunları görüntüleme, görüntülenen görevleri silme, üzerlerinde düzenleme yapma gibi görevleri sunmasının yanında sayfayı temizleme ve görevleri tamamlama gibi özellikler de bulundurur. Tamamlanan görevler alt tarafta bulunan ayrı bir kısımda gösterilir. Tamamlanan görevler tekrar tamamlanmadı kısmına çıkartılabildiği gibi tersi de yapılabilir. Tamamlanan görevler temizlenebilir. Uygulama kapandığında uygulamadaki verilerin kaybedilmemesi için veriler bir dosyada depolanır. Böylece kullanıcı için garanti sağlanmış olur.

1. **Kullanıcı Arayüzü Tasarımı:**

Kullanıcı arayüzünü tasarlarken tkinter kütüphanesinden faydalandım:

metin, ekran görüntüsü, yazılım, web sayfası içeren bir resim

Açıklama otomatik olarak oluşturuldu

* İlk olarak sol üstte görüleceği gibi sayfa ikonuna bir ikon ekledim. Uygulamanın öznelliği açısından faydalı olduğunu düşünüyorum.
* Ben arayüzde bileşenleri yerleştirirken grid() metodunu kullandım. Place ve pack metodlarını da denedim ancak grid kadar işlevsel olmadılar.
* Yukardan aşağı doğru inecek olursak, en üstte bir başlık bulunuyor. Ardından eklemek istenilen görevin adı ve teslim tarihini girdiğimiz bir kısım var. Ekle butonuna basıldığında görevi tasks kısmında görebiliyoruz. Eğer bu görevleri değiştirmek, silmek istersek al kısımda isteğimize uygun butona tıklayarak işlemi halletmiş oluyoruz. Bir görev seçilip completed butonuna basılırsa görev completed tasks bölümüne geçiyor. Tam tersini yapmak da mümkün.



1. **Görevlerin depolanması:**

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, yazı tipi içeren bir resim

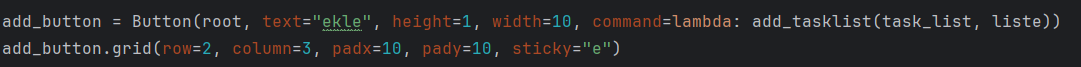
Açıklama otomatik olarak oluşturuldu

Buradan görüleceği üzere görevleri depolamak için liste isimli bir iki boyutlu liste kullandım. Birinci boyutunda görevi ikinci boyutunda ise teslim tarihini tutuyor. Örneğin burada, ekle butonuna basınca görev listesini ve listeyi parametre olarak gönderiyor. Ve verileri değişkenlere aldıktan sonra bunları task\_info isimli bir başka listeye atıyor. En son da parametre olarak verdiğimiz listeye bu yeni listeyi append ile ekliyoruz.

1. **GÖREV EKLEME:**

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu



add\_tasklist fonksiyonu ile görev ekleme kısmını yapıyoruz. Öncelikle görevin yazıldığı yer olan text\_task textfieldından get anahtar sözcüğü ile başından sonuna kadar metni aldık ve new\_task değişkenine atadık. ardından date\_picker isimli widgettan tarihi çekmek için get\_date() metodunu kullandık. Due\_date değişkenine atadık. task\_info listesinde bu iki değişkeni tutuyorum. Ve bunları birleşik olarak ana liste olan listeye atıyorum.

En son da güncellenmiş listeleri kaydetmesi için save\_to\_csv ve update\_task\_list fonksiyonuna bu listeyi ve listboxu gönderiyorum.

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

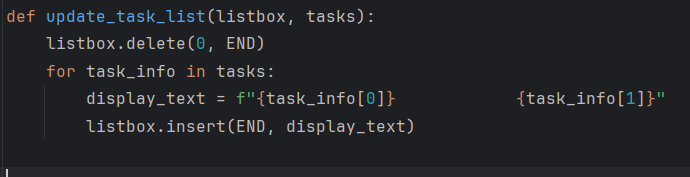
1. **GÖREVLERİ GÖRÜNTÜLEME:**

label\_tasks = Label(root, text="Tasks", font="verdana 11 bold", bg="light blue", fg="black")  
label\_tasks.grid(row=3, column=0, padx=10, pady=10, sticky="w")  
  
frame\_tasks = Frame(root, bg="#cdc9c9", height=350, width=550)  
frame\_tasks.grid(row=4, column=0, padx=10, pady=10, sticky="w", columnspan=4)  
  
task\_list = Listbox(frame\_tasks, bg="#eee8aa", fg="black", font="verdana 10 bold", height=10, width=50,  
 selectbackground="#00868b", activestyle="none", selectmode=tkinter.SINGLE)  
task\_list.pack(side="left", fill="both")  
  
label\_completed\_tasks = Label(root, text="Completed Tasks", font="verdana 11 bold", bg="light blue", fg="black")  
label\_completed\_tasks.grid(row=6, column=0, padx=10, pady=10, sticky="w", columnspan=2)  
  
frame\_completed\_tasks = Frame(root, bg="#cdc9c9", height=40, width=400)  
frame\_completed\_tasks.grid(row=7, column=0, padx=10, pady=10, sticky="w", columnspan=4)  
  
completed\_font = font.Font(family='Tahoma', weight="bold", size=10, underline=TRUE, slant="italic")  
completed\_tasks = Listbox(frame\_completed\_tasks, bg="#9cff72", fg="black", font=completed\_font, height=4, width=60,  
 selectbackground="#00868b", activestyle="none", selectmode=tkinter.SINGLE)  
completed\_tasks.pack(side="left", fill="both")

görevleri görüntüleme adımında kodumu açıklayacak olursam:

tasks isimli bir label ile başlık oluşturdum listboxun üzerine yerleşebilmesi için frame\_tasks isimli frame oluşturdum. Task\_list isimli görevleri görüntüleyecek listboxu üzerine koydum. Ve bunları grid ile yerleştirdikten sonra aynılarını completed\_tasks listboxu için yaptım. Şuan bunları görüntüleyecek durumdayız.





Eklenen görevlerin veya güncellenen, silinen görevlerin listboxta görünmesini sağlayan kısım ise update\_task\_list fonksiyonu. Değişikliğin gösterileceği listboxu ve verilerin tutulduğu listeyi parametre olarak alır. Başından sonuna kadar siler, ardından for döngüsü kullanarak her bir liste elemanı için öncelikle görev ismi indexini ardından teslim tarihi indexini araya boşluk bırakarak display\_text değişkenine atar. Sonra da bunu insert komutuyla listboxa ekler. Bu şekilde de oluşturduğumuz listbox içinde görevleri ve yapılan değişiklikleri anında görürüz.

1. **GÖREVLERİ TAMAMLAMA:**

metin, ekran görüntüsü, yazı tipi, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

complete\_button = Button(root, text="Completed", bg="#024f00", fg="white", font=buttonfont, width=10, height=2, command=lambda:completed\_task(task\_list, completed\_tasks))  
complete\_button.grid(row=5, column=1, padx=10, pady=10, sticky="w")  
  
uncomplete\_button = Button(root, text="Uncompleted", bg="#ff430a", fg="white", font=buttonfont, width=10, height=2, command=lambda:not\_completed\_task(task\_list, completed\_tasks))  
uncomplete\_button.grid(row=8, column=0, padx=10, pady=10, sticky="w")

Bu işlemler için oluşturulan butonları açıklayayım önce. Complete\_button isimli buton görülen özelliklere sahip ve command=lambda: ile belirtilen completed\_task fonksiyonuna task\_list ve completed\_tasks listboxlarını iletiyor.

metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Completed\_task fonksiyonunda bu parametrelerden ilki olan listboxu kullanarak seçilen satırı selection\_items değişkenine atıyor. Selected\_items dolu ise if bloğu içine girerek listboxtan seçilen satırı alıp completed değişkenine atıyoruz. Görevler listboxundan bu satırı sildikten sonra tamamlanan görevler listboxuna insert ile ekliyoruz. Ardından iki listbox için de kaydetme işlenmlerini yapıyoruz. Bu şekilde görevler tamamlandı listesinde görülüyor.

metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Bu da completed\_tasks listboxundan task\_list listboxuna görev atamak için kullanılan fonksiyon. İşlemler aynı şekilde hallediliyor.

1. **GÖREV SİLME:**
2. deleteAll\_button = Button(root, text="Delete All", bg="#890505", font=buttonfont, fg="white", width=10, height=2, command=lambda:del\_all\_tasks(task\_list, liste))  
   deleteAll\_button.grid(row=5, column=0, padx=10, pady=10, sticky="w")

metin, yazı tipi, ekran görüntüsü içeren bir resim

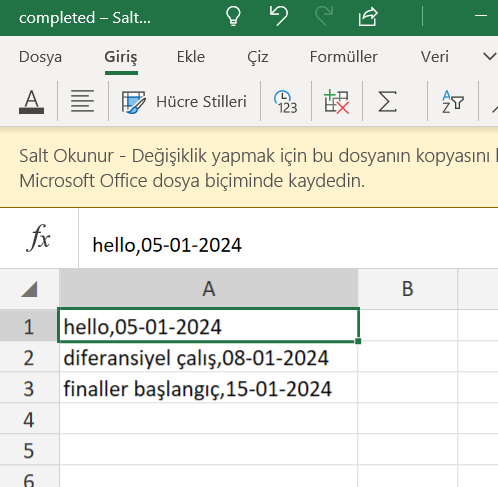
Açıklama otomatik olarak oluşturuldu

1. delete\_button = Button(root, text="Delete", bg="#ff0022", fg="white", font=buttonfont, width=10, height=2, command=lambda:del\_selected\_task(task\_list, liste))  
   delete\_button.grid(row=5, column=0, padx=10, pady=10, sticky="e")  
    metin, ekran görüntüsü, yazı tipi içeren bir resim

   Açıklama otomatik olarak oluşturuldu
2. clear\_button = Button(root, text="Clear", bg="#559606", fg="white", width=10, font=buttonfont, height=2, command=lambda:del\_all\_tasks(completed\_tasks, completed\_list))  
   clear\_button.grid(row=8, column=1, padx=10, pady=10, sticky="w")
3. Delete\_all\_tasks: Bu maddedeki kod kısmı hepsini silme butonu del\_all\_tasks metoduna gider. Bu metodda gelen listboxu başından sonuna kadar (0 to END) delete komutuyla siler ve görevler listesini temizler. Son olarak kaydeder ve günceller.
4. Delete\_selected\_task: bu maddede delete\_butonu oluşturduk ve içine komut olarak del\_selected\_task fonksiyonunu çağırdık. Seçilen satır için itemleri döngüde döndürür ve bunları siler. Sonrasında del tasks koduyla listeden de itemi siler. Son olarak kodu kaydedip günceller.
5. Clear: bu kod da aslında completed\_tasks listboxu için oluşturulan bir buton fakat komut olarak del\_all\_tasks fonksiyonuna gidiyor. Çünkü parametreleri zaten farklı verdiğimiz için kod tekrarına düşmek istemedim.

Tüm silme fonksiyonları bu şekilde tek seferde temizleme ve tek tek seçerek temizleme komutlarını ayrı ayrı koymak istedim, kullanıcının istekleri farklılık gösterebiliyor.

1. **VERİYİ DOSYAYA KAYDETME:**

 metin, ekran görüntüsü, sayı, numara, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Genel bir bilgi verecek olursam, dosyaya kaydetme işlemi için CSV dosya formatını kullandım. Görevler listboxu ve tamamlanan görevler listboxu için iki ayrı dosya kullandım.

1. def save\_to\_csv(data, listbox):  
    try:  
    filename = get\_filename\_for\_listbox(listbox)  
    with open(filename, "w", newline="") as csvfile:  
    csvwriter = csv.writer(csvfile)  
    csvwriter.writerows(data)  
    except ValueError as e:  
    tkinter.messagebox.showerror(title="Error", message=str(e))

1- Save\_to\_csv : listboxu dosyaya kaydeder. Yakından inceleyelim:

Data ve listbox parametrelerini alır. Data -> listedeki veriler. Try-catch blokları kullanarak olası hataları yakalamaya çalıştım. Öncelikle iki dosya olduğu için dosya adını get\_fileneme\_for\_listbox fonksiyonuyla aldık ve filename değişkenine atadık. “with open(dosyaadı, kipi) as dosyadeğişkeni” kalıbıyla yazma türünde dosyayı açtık. Csv dosyasına yazma işlemi için csvwriter isimli nesne oluşturulur ve dosyaya data yani veriler yazılır.

1. def save\_completed\_to\_csv(data, listbox):  
    try:  
    filename = get\_filename\_for\_listbox(listbox)  
    with open(filename, "w", newline="") as csvfile:  
    csvwriter = csv.writer(csvfile)  
    csvwriter.writerows(data)  
    except ValueError as e:  
    tkinter.messagebox.showerror(title="Error", message=str(e))

2-save\_completed to\_csv: ilk fonksiyonla aynı çalışır. Tek farkı bunun completed\_task listboxu için kullanılması

1. def load\_from\_csv(listbox):  
    filename = get\_filename\_for\_listbox(listbox)  
    try:  
    with open(filename, newline="") as csvfile:  
    csvreader = csv.reader(csvfile)  
    return [row for row in csvreader]  
    except FileNotFoundError:  
    tkinter.messagebox.showinfo(title="exception", message="dosya bulunamadı!")

3- load\_from\_csv: csv dosyasından listboxa tekrar veri yüklemeyi sağlar. Try-catch bloklarıyla kodu koontrole alır. Csvreader nesnesi oluştururulur ve kodu okur. Bu şekilde dosya kapatıldığında csv dosyasına kaydedilen kodlar uygulama açılınca tekrardan uygulamaya geçer.

1. def update\_task\_list(listbox, tasks):  
    listbox.delete(0, END)  
    for task\_info in tasks:  
    display\_text = f"{task\_info[0]} {task\_info[1]}"  
    listbox.insert(END, display\_text)

4-update\_task\_list: öncelikle parametre olarak gelen listboxu baştan sona siler. Ve tasks listesindeki bilgileri teker teker alıp insert eder.

1. def get\_filename\_for\_listbox(listbox):  
    # Use different filenames for different listboxes  
    if listbox == task\_list:  
    return "C:/Users/ramco/OneDrive/Masaüstü/tasks.csv"  
    elif listbox == completed\_tasks:  
    return "C:/Users/ramco/OneDrive/Masaüstü/completed.csv"  
    else:  
    tkinter.messagebox.showinfo(title="exception", message="dosya bulunamadı!")  
    return None # Return None in case of an unknown listbox

5-biraz önce de bahsettiğim gibi işlem yapılacak olan listboxu bulur ve dosya yolunu döndürür. Hata varsa kontrol eder.

1. liste = load\_from\_csv(task\_list)  
   completed\_list = []  
   completed\_list = load\_from\_csv(completed\_tasks)

6- burda fonksiyonların çağırılmış hali bulunuyor

1. **VERİ DOĞRULAMA:**

Bu kısımda textfielddan alınan değer boş olunca kullanıcıya mesaj kutusu ile uyarı döndürdüm.

metin, ekran görüntüsü, yazı tipi, web sayfası içeren bir resim

Açıklama otomatik olarak oluşturuldu

Burada görüleceği üzere düzenlemek istediğim satırı seçip görev listesini boş bıraktığımda edite veya ekleye basınca bu uyarıyı alıyorum. Bu da kullanıcıya doğru veriyi girmesi için olanak tanır. Aynı zamanda satır seçmeden görev eklemek istediğimizde de eklenmeyecektir.

1. **TEST ETME:**

Bu kısmı aslında parça parça yukarıdaki metotların içinde verdim. Bu yüzden sadece açıklayayım. Yazdığım kodlarda bulduğum hatalara ve eksikliklere göre ya try-catch bloklarıyla ya da if else bloklarıyla hataları kontrol altına almaya ve kullanıcıya da mesaj kutularıyla göstermeye çalıştım. Örneğin:

metin, yazı tipi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

Burada if not edited\_task kısmında edited\_task dolu değilse içeri girer ve kullnaıcıya mesaj gösterir.

**Ek olarak:** edit\_task fonksiyonu

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Aşağıda bu kod için kullandığım tüm değişken ve fonksiyonlar bulunuyor:**

**add\_tasklist:** Yeni bir görevi listeye ekleyen fonksiyon.

**del\_all\_tasks:** Listeden tüm görevleri silen fonksiyon.

**edit\_task:** Seçilen bir görevi düzenlemeye izin veren fonksiyon.

**del\_selected\_task**: Seçilen görevi listeden silen fonksiyon.

**completed\_task**: Seçilen bir görevi tamamlanan görevler listesine taşıyan fonksiyon.

**not\_completed\_task**: Seçilen tamamlanmış görevi ana görev listesine geri taşıyan fonksiyon.

**save\_to\_csv**: Mevcut liste verilerini bir CSV dosyasına kaydeden fonksiyon.

**save\_completed\_to\_csv:** Tamamlanan görev verilerini bir CSV dosyasına kaydeden fonksiyon.

**load\_from\_csv:** Bir CSV dosyasındaki verileri bir liste içine yükleyen fonksiyon.

**update\_task\_list:** GUI'de görünen listeyi güncelleyen fonksiyon.

**get\_filename\_for\_listbox:** Liste kutusunun türüne bağlı olarak dosya adını belirleyen fonksiyon.

**root:** Ana Tkinter penceresi.

**label\_title**: Başlığı gösteren etiket widget'ı.

**label2:** Bir giriş isteği gösteren etiket widget'ı.

**text\_task**: Yeni görevleri girmek için metin widget'ı.

**label\_due\_date**: Bitiş tarihi seçimi hakkında bir istek gösteren etiket widget'ı.

**date\_picker:** Bitiş tarihi seçimi için kullanılan DateEntry widget'ı.

**label\_tasks**: Ana görev listesini gösteren etiket widget'ı.

**frame\_tasks**: Ana görev listesini içeren widget.

**task\_list:** Ana görev listesini gösteren liste widget'ı.

**label\_completed\_tasks:** Tamamlanan görevler listesini gösteren etiket widget'ı.

**frame\_completed\_tasks**: Tamamlanan görevler listesini içeren widget.

**completed\_font:** Tamamlanan görevler listesi için font yapılandırması.

**completed\_tasks**: Tamamlanan görevler listesini gösteren liste widget'ı.

**liste:** Ana görev listesi verilerini depolamak için kullanılan liste.

**add\_button**: Yeni görev eklemek için kullanılan buton widget'ı.

**scrollbar\_vert:** Ana görev listesi için dikey kaydırıcı.

**deleteAll\_button:** Tüm görevleri silmek için kullanılan buton widget'ı.

**edit\_button:** Görevleri düzenlemek için kullanılan buton widget'ı.

**delete\_button:** Seçili görevleri silmek için kullanılan buton widget'ı.

**complete\_button:** Görevleri tamamlananlar listesine taşımak için kullanılan buton widget'ı.

**uncomplete\_button:** Tamamlanmış görevleri ana liste geri taşımak için kullanılan buton widget'ı.

**clear\_button**: Tamamlanan görevleri silmek için kullanılan buton widget'ı.

**completed\_list:** Tamamlanan görevler listesi verilerini depolamak için kullanılan liste.

**load\_from\_csv(task\_list):** CSV dosyasındaki verileri ana görev listesine yükleyen fonksiyon.

**completed\_list:** CSV dosyasındaki verileri tamamlanan görevler listesine yükleyen fonksiyon.

**update\_task\_list(task\_list, tasks):** GUI'de görünen listeyi güncelleyen fonksiyon.

**root.mainloop():** Tkinter olay döngüsünü başlatan fonksiyon**.**

Kullandığım kütüphaneler:

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu