TKİNTER İLE GÖRSEL PROGRAMLAMA

ÖĞR. GÖR. GÖZDE MİHRAN ALTINSOY

TKINTER

01

Tkinter, Python kurulumu ile birlikte gelen ve pencereli-menülü modern programlar yazmamızı sağlayan grafik arayüz geliştirme takımlarından biridir.

02

Tkinter bir standart kütüphane paketi olduğu için, Python programlama dilini kurduğunuzda Tkinter de otomatik olarak kurulur.

03

Python'da grafik arayüzlü programlar yazmamızı sağlayacak tek modül Tkinter değildir. Bunun dışında PyQt, PyGI ve Kivy gibi alternatifler de bulunur. Ancak Tkinter'in öteki alternatiflere karşı en büyük üstünlüğü çok daha kolay olması ve Python'la birlikte gelmesidir.

TKINTER

PyQt, PyGI ve Kivy'yi kullanabilmek için öncelikle bunları bilgisayarınıza kurmanız gerekir.

Tkinter dışındaki alternatifleri kullanarak yazdığınız programları dağıtırken, bu arayüz kütüphanelerini kullanıcılarınızın bilgisayarına bu kütüphaneleri kurmasını talep etmeniz gerekir.

Tkinter dışındaki alternatifleri kullanarak yazdığınız programları dağıtırken, bu arayüz kütüphanelerini kullanıcılarınızın bilgisayarına bu kütüphaneleri kurmasını talep etmeniz gerekir.

Tkinter hem nesne tabanlı programlama hem de grafik arayüz geliştirme kavramlarını öğrenmek açısından son derece uygun bir ortamdır.

TKINTER PENCERE

```
import tkinter
#Modül içeri aktarılıyor.
pencere = tkinter.Tk()
#Tk() sınıfını pencere adıyla örnekledik
pencere.mainloop()
```

```
import tkinter as tk
#Modül içeri aktarılıyor.
pencere = tk.Tk()
pencere.mainloop()
```

```
from tkinter import *
#Modül içeri aktarılıyor.
pencere = Tk()
pencere.mainloop()
```

Bu pencere örnekleme ile birlikte oluşmuş olsa da, Tkinter'in iç işleyişi gereği, 'ana döngü' adlı bir mekanizma çalışmaya başlamadan görünür hale gelmez. İşte bu özel ana döngü mekanizmasını çalıştırmak ve böylece oluşturduğumuz pencereyi görünür hale getirmek için, `Tk()` sınıf örneklerinin `mainloop()` adlı bir metodunu çalıştırdık.

TKINTER PENCERE

Tkinter'le oluşturulan boş bir pencere öntanımlı olarak 200 piksel genişliğe ve 200 piksel yüksekliğe sahip olur.

Tk() sınıfının geometry() adlı metodunu kullanarak, pencere boyutunu ayarlayabiliriz

Tk() sınıfının hangi metotlara sahip olduğunu görmek için dir(pencere) komutunu yazabiliriz.

```
import tkinter as tk
pencere = tk.Tk()
pencere.geometry('200x70')
#Tk() sınıf örneklerinin geometry() metodunu kullanarak 200x200 yerine 200x70 boyutlarında bir pencere oluştu.
pencere.mainloop()
```

TKINTER NESNELERİ PROSEDÜREL YAKLAŞIMLA ÖRNEK

```
import tkinter as tk

pencere = tk.Tk()
pencere.geometry('200x70')

etiket = tk.Label(text='Merhaba Dünya’)
#Label() sınıfı ile etiket oluştu.
etiket.pack()
#pack() metodu ile etiket ve düğmeler pencere üzerine yerleşti.

düğme = tk.Button(text=‘Çıkış', command=pencere.destroy)
#Button() sınıfı ile düğme oluştu. Butonun üzerine tıklandığında pencere kapanacak.
düğme.pack()
#pack() metodu ile etiket ve düğmeler pencere üzerine yerleşti.

pencere.mainloop()
```

TKINTER NESNELERİ

düğme = tk.Button(text='Çıkış', command=pencere.destroy)

Düğmenin üzerine tıklandığında ne olacağını Button() sınıfının command parametresi aracılığıyla belirledik.

Bu parametreye, pencere örneğinin destroy() metodunu verdiğimizde pencereye kapatma sinyali gönderilecektir.

Bu metodu yazarken parantez işaretlerini kullanmadık. Eğer metodu pencere.destroy() şeklinde parantezli bir biçimde yazarsak, kapatma komutu daha düğmeye basmadan çalışacak ve bu durumda düğmemiz düzgün işlemeyecektir.

TKINTER NESNELERİ

```
import tkinter as tk

pencere = tk.Tk()

def çıkış():
    etiket['text'] = ‘Güle güle...'
    düğme['text'] = 'Bekleyin...'
    düğme['state'] = 'disabled’
    #Düğmeyi pasif yaptık, artık düğmeye tıklanamaz
    pencere.after(2000, pencere.destroy)
#2000 milisaniye (yani 2 saniye) sonra pencere.destroy() komutunu işleterek pencerenin kapanmasını sağladık.

etiket = tk.Label(text='Merhaba')
etiket.pack()
düğme = tk.Button(text='Çık', command=çıkış)
#Düğmeye basıldığında, command parametresinin değeri olan çıkış() fonksiyonu çalışmaya başlayacak ve fonksiyon gövdesinde tanımladığımız işlemler gerçekleşecek.
düğme.pack()

pencere.protocol('WM_DELETE_WINDOW', çıkış)
#‘X’ düğmesine basıldığında da pencere kapanmadan önce çıkış() fonksiyonunun çalışmasını sağladık

pencere.mainloop()
```

TKINTER NESNELERİ
NESNE TABANLI YAKLAŞIMLA ÖRNEK

```
import tkinter as tk

class Pencere(tk.Tk):
    def __init__(self):
        super().__init__()
        self.protocol('WM_DELETE_WINDOW', self.çıkış)

        self.etiket = tk.Label(text='Merhaba')
        self.etiket.pack()

        self.düğme = tk.Button(text='Çık', command=self.çıkış)
        self.düğme.pack()

    def çıkış(self):
        self.etiket['text'] = ‘Güle güle...'
        self.düğme['text'] = 'Bekleyin...'
        self.düğme['state'] = 'disabled'
        self.after(2000, self.destroy)

pencere = Pencere()
pencere.mainloop()
```

TKINTER NESNELERİ
NESNE TABANLI YAKLAŞIMLA ÖRNEK

```
import tkinter as tk


class Pencere(tk.Tk):
    def __init__(self):
        super().__init__()
```

- Pencere oluşur oluşmaz işletilecek kodları tanımlamak için bir __init__() metoduna ihtiyacımız var.
- Ancak kendi __init__() metodumuzu tanımlarken, Tk()sınıfının kendi __init__() metodundaki işlemleri de gölgelememiz lazım.
- Dolayısıyla orijinal __init__() metodunu kendi __init__() metodumuza aktarmak için super() fonksiyonundan yararlandık.

PROTOCOL

```
self.protocol('WM_DELETE_WINDOW', self.çıkış)
```

- protocol() metodunun öntanımlı davranışı, pencerenin ‘X’ düğmesine basıldığında programı sonlandırmaktır.
- Bu öntanımlı davranışı değiştirmek için protocol() metodunu içeren kodu tekrar tanımladık ve ‘X’ düğmesine basıldığında çıkış() fonksiyonunun çalışmasını sağladık.

PYTHON MESSAGEBOX

```
 from tkinter import *
from tkinter import messagebox
pencere = Tk()
pencere.title("Tkinter penceresi")
pencere.geometry("600x300")

#formu grid olarak çizdirme /layout düzeni
uygulama = Frame(pencere)
uygulama.grid()
```

```
def dialog():
    var = messagebox.showinfo("Uyarı" , "Mesajınız var")

button1 = Button(uygulama, text = " Uyarı Ver " , width=20, command=dialog)
button1.grid(padx=110, pady=80)

#formu çiz
pencere.mainloop()
```

TKINTER NESNELERİ
ETİKET

```
from tkinter import *

from tkinter import messagebox

pencere = Tk()

pencere.title("Tkinter")
pencere.geometry("400x200")

#formu grid olarak çizdirme /layout düzeni
uygulama = Frame(pencere)
uygulama.grid()

#label nesnesini çiz
etiket = Label(uygulama,text="Merhaba")
etiket.grid(padx=110, pady=10)

#formu çiz
pencere.mainloop()
```

TKINTER NESNELERİ ETİKET VE BUTON

```
from tkinter import *

from tkinter import messagebox

pencere = Tk()

pencere.title("Tkinter")
pencere.geometry("400x300")

uygulama = Frame(pencere)
uygulama.grid()

#mesaj fonksiyonu
def dialog():
    var = messagebox.showinfo("Uyarı" , "Dikkat")

#buton nesnesini çiz ve fonksiyonu bağla
button1 = Button(uygulama, text = " Uyarı Ver " , width=20, command=dialog)
button1.grid(padx=110, pady=70)

#label nesnesini çiz
etiket = Label(uygulama,text="Merhaba")
etiket.grid(padx=110, pady=10)

#formu çiz
pencere.mainloop()
```

TKINTER NESNELERİ CHECKBOX

```
from tkinter import *

from tkinter import messagebox

pencere = Tk()

pencere.title("Tkinter")
pencere.geometry("400x300")

#grid form çizdirme
uygulama = Frame(pencere)
uygulama.grid()

chek1=Checkbutton(uygulama, text = "Kitap Okuma", onvalue = 1, offvalue = 0, height=5, width = 20)
chek1.grid(padx=110, pady=10)

chek2=Checkbutton(uygulama, text = "Spor Yapma", onvalue = 1, offvalue = 0, height=5, width = 20)
chek2.grid(padx=110, pady=5)

#formu çiz
pencere.mainloop()
```

TKINTER NESNELERİ
ENTRY

```
from tkinter import *

from tkinter import messagebox

pencere = Tk()

pencere.title("Tkinter")
pencere.geometry("400x300")

#grid form çizdirme
uygulama = Frame(pencere)
uygulama.grid()

L1 = Label(uygulama, text="Adınızı Girin")
L1.grid(padx=110, pady=10)

E1 = Entry(uygulama, bd =2)
E1.grid(padx=110, pady=3)

#formu çiz
pencere.mainloop()
```

TKINTER NESNELERİ LISTBOX

```
from tkinter import *

from tkinter import messagebox

pencere = Tk()

pencere.title("Tkinter")
pencere.geometry("400x300")

#grid form çizdirme
uygulama = Frame(pencere)
uygulama.grid()

Lb1 = Listbox(uygulama)
Lb1.insert(1, "Python")
Lb1.insert(2, "C#")
Lb1.insert(3, "JAVA")
Lb1.insert(4, "JAVASCRIPT")
Lb1.grid(padx=110, pady=10)

#formu çiz
pencere.mainloop()
```

KAYNAKLAR

- https://belgeler.yazbel.com/python-istihza/nesne_tabanli_programlama6.html , Erişim: 24.08.2019
- https://www.yazilimbilisim.net/python/python-tkinter-ornekleri/ , Erişim: 24.08.2019