Clase 22: Interfaces gráficas y ventanas (módulo Tkinter)

Veremos cómo crear interfaces gráficas usando una clase especial llamada Tkinter. Para eso, haremos varios ejemplos, llendo desde los más básicos hasta los más complicados.

Todos los ejemplos de esta clase pueden ejecutarse y verse de dos maneras

- Usando IDLE: requiere instalación en Windows, Mac o Linux.
- Usando la plataforma Repl.it (ver video tutorial de uso en ucursos).

Ej.1: Programa que muestra una ventana con la palabra “hola”

```
# incluye clases predefinidas
from tkinter import *

# crear ventana (con barra de título)
ventana = Tk()

# crear componente (con texto) y agregar a la ventana
saludo = Label(ventana,text="Hola")

# acomodar componente en ventana
saludo.pack()

# mostrar ventana y esperar cierre (click en botón X)
ventana.mainloop()
```

```
# ventana_hola.py
from tkinter import*
ventana = Tk()
saludo = Label(ventana, text = "Hola")
saludo.pack()
ventana.mainloop()
```

```
from tkinter import *
```

deja disponibles (visibles) clases predefinidas: **Tk, Label**

```
ventana = Tk()
```

crea objeto de la clase predefinida **`Tk`** que:

- inicializa **`tkinter`** (interfaz con sistema operativo)
- representa una ventana en la pantalla (con una barra de título)

```
saludo = Label(ventana,text="Hola")
```

crea objeto de la clase predefinida **`Label`** que

- es una componente (widget) de la ventana
- contiene el texto “Hola”

```
saludo.pack()
```

invoca al método pack del objeto componente que

- acomoda la componente en la ventana para hacerla visible

```
ventana.mainloop()
```

invoca al método mainloop del objeto ventana que

- muestra la ventana en la pantalla
- ejecuta un **"loop"** de espera de eventos (ej: cerrar la ventana)

Ej.2: Ventana con 3 componentes: Programa para Saludar en Inglés y Español

- 2 botones (objetos de la clase Button)
- 1 componente de la clase Label

Idea?

- click en el primer botón debe mostrar “hola” en Label
- click en el 2° botón debe mostrar “hello” en Label
- Se necesitan métodos, asociados a los botones, que se ejecuten al hacer click con el mouse
- programa termina cuando se cierre ventana (botón externo X en Windows o botón rojo en Mac)

```
# ventana2botones.py
from tkinter import *

#metodos que se ejecutan en caso de click en botones
def saludar():
    saludo.config(text="hola")
```

```
def greet():
    saludo.config(text="hello")

#ventana
ventana = Tk()

#boton "espanol", al hacer click ejecuta saludar()
boton1 = Button(ventana,text="espanol",command=saludar)
boton1.pack()  #ubica boton en la ventana

#boton "english", al hacer click ejecuta greet()
boton2 = Button(ventana,text="english",command=greet)
boton2.pack()  #ubica boton en la ventana

#componente para mostrar saludo
saludo= Label(ventana)
saludo.pack() #ubica Label debajo
#loop para clicks en botones (hasta cerrar ventana)
ventana.mainloop()
```

Ej.3: Saludador personalizado ☺

Ventana con 3 componentes:

- 1 Label con pregunta “Cuál es tu nombre?
- 1 componente para recibir respuesta del usuario (de clase Entry)
- 1 Label para saludo personalizado

¿Qué queremos que haga?

- después de ingresar nombre se debe mostrar saludo **“hola xxx”**
- se necesita un método, asociado a la 2a componente, que se ejecute al terminar de ingresar un nombre (caracteres hasta tecla **<Return>**)
- programa saluda a todas las personas que quieran ☺
- programa termina cuando se cierre ventana (botón externo X/rojo)

```
# ventanaSaludoPersonalizado.py
from tkinter import *
#metodo que se ejecuta en caso de ingresar un nombre
def saludar(x):
    r=respuesta.get()  #obtener respuesta
    saludo.config(text="hola "+ r)

#programa que saluda
ventana = Tk()
pregunta = Label(ventana,text="Cual es tu nombre?")
pregunta.pack()
```

```
#componente para ingresar nombre
respuesta = Entry(ventana)
respuesta.pack()
respuesta.bind("<Return>",saludar) #espera un evento sobre Entry

#componente para mostrar saludo
saludo=Label(ventana)
saludo.pack()
ventana.mainloop()
```

Ej.4: Saludador personalizado bilingüe ☺

Ventana con 5 componentes:

- 2 botones (para elegir idioma)
- 1 Label para preguntar nombre
- 1 Entry para recibir nombre
- 1 Label para mostrar saludo

Idea?

- seleccionar idioma con botones
- preguntar y saludar en el idioma seleccionado
- programa termina cuando se cierra ventana (botón externo X)

```
# ventanaSaludoPersBilingue.py
from tkinter import *

#metodo que se ejecuta en caso de ingresar un nombre
def saludar():
    respuesta.bind("<Return>",saludarEspanol)
    pregunta.config(text="Cual es tu nombre?")

def saludarEspanol(x):
    r=respuesta.get()  #obtener respuesta
    saludo.config(text="hola "+ r)

def greet():
    respuesta.bind("<Return>",greetEnglish)
    pregunta.config(text="What is your name?")

def greetEnglish(x):
    r=respuesta.get()  #obtener respuesta
    saludo.config(text="hello "+ r)

#saludador bilingue
ventana = Tk()
```

```
pregunta = Label(ventana,text="")
#boton "espanol", al hacer click ejecuta saludar()
boton1 = Button(ventana,text="espanol",command=saludar)
boton1.pack()  #ubica boton en la ventana
#boton "english", al hacer click ejecuta greet()
boton2 = Button(ventana,text="english",command=greet)
boton2.pack()  #ubica boton en la ventana
pregunta.pack()

#componente para ingresar nombre
respuesta = Entry(ventana)
respuesta.pack()

#componente para mostrar saludo
saludo=Label(ventana)
saludo.pack()
ventana.mainloop()
```

Propuesto: Haga las siguientes posibles mejoras:

- Intente borrar el saludo del label cuando haga click en cualquiera de los dos botones. **Pistas**:
  - Si saludo es un objeto Label, para borrar el texto desplegado debe hacer `saludo.config(text="")`.
  - Si respuesta es un objeto Entry, para borrar el texto adentro debe hacer `respuesta.delete(0,END)`.
- Agréguele un boton nuevo que permita salir. **Pista:** Para crear un boton cuya acción sea cerrar la ventana, debe asignarle `command=ventana.destroy` al crearla.

Ej.5: Rediseño de ventana del Personalizado Saludador Bilingue

Ventana con 3 componentes:

- un “marco” con los botones
- un “marco” con la pregunta y respuesta
- 1 Label para mostrar saludo

“marco”?

- objeto de clase Frame
- objeto para agrupar componentes
- la idea es agrupar componentes que tengan alguna relación lógica

```
# ventanaSaludoPersBilingueFrame.py
from tkinter import *
```

```
#metodo que se ejecuta en caso de ingresar un nombre
def saludar():
    respuesta.bind("<Return>",saludarEspanol)
    pregunta.config(text="Cual es tu nombre?")

def saludarEspanol(x):
    r=respuesta.get()  #obtener respuesta
    saludo.config(text="hola "+ r)

def greet():
    respuesta.bind("<Return>",greetEnglish)
    pregunta.config(text="What is your name?")

def greetEnglish(x):
    r=respuesta.get()  #obtener respuesta
    saludo.config(text="hello "+ r)

#programa que saluda
ventana = Tk()

marco1=Frame(ventana)
marco1.pack()

#boton "espanol", al hacer click ejecuta saludar()
boton1 = Button(marco1,text="espanol",command=saludar)
boton1.pack(side=LEFT)  #ubica boton en la ventana
#boton "english", al hacer click ejecuta greet()
boton2 = Button(marco1,text="english",command=greet)
boton2.pack()  #ubica boton en la ventana

marco2=Frame(ventana)
marco2.pack()

pregunta = Label(marco2)
pregunta.pack(side=LEFT)

#componente para ingresar nombre
respuesta = Entry(marco2)
respuesta.pack()

#componente para mostrar saludo
saludo=Label(ventana)
saludo.pack()
ventana.mainloop()
```

Ej.6: Dibujar diagonales de la ventana (en colores azul y rojo)

```
from tkinter import *
ventana=Tk()
# objeto (“tela”) para dibujar
ancho=200; alto=100 #pixeles
cv=Canvas(ventana,width=ancho,height=alto)
cv.pack()
# dibujar diagonales (azul y roja)
# cv.create_line(h1,v1,h2,v2,fill=”color”)
cv.create_line(0,0,ancho,alto,fill="blue")
cv.create_line(0,alto,ancho,0,fill="red")
ventana.mainloop()
```

Nota:

- coordenadas se expresan en la forma horizontal,vertical

```
# canvasDiagonales.py
from tkinter import *
ventana=Tk()
# objeto canvas para dibujar
ancho=200; alto=100 #pixeles
cv=Canvas(ventana,width=ancho,height=alto)
cv.pack()
#dibujar diagonales (azul y roja)
#cv.create_line(h1,v1,h2,v2,fill="color")
cv.create_line(0,0,ancho,alto,fill="blue")
cv.create_line(0,alto,ancho,0,fill="red")
ventana.mainloop()
```

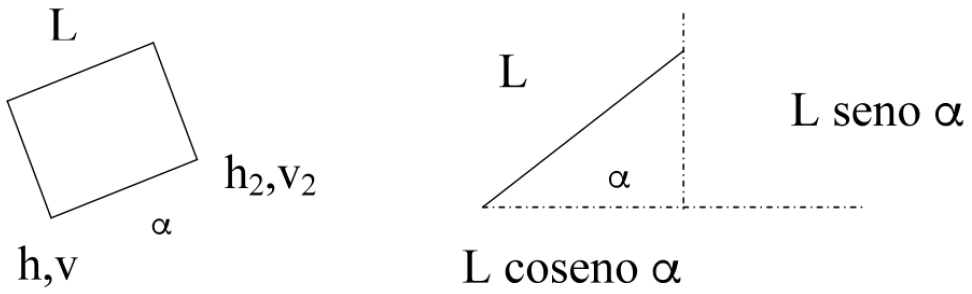
```
# canvas2ovalos.py
from tkinter import *

ventana=Tk()
cv=Canvas(ventana,width=200,height=200)
cv.pack()
cv.create_oval(20,40,100,100)
cv.create_oval(100,100,200,200,fill="black")
ventana.mainloop()
```

Nota. create_rectangle(...) dibuja rectángulo

```
#circunferencias-al-azar.py
from tkinter import *
import random
def dibujar():
```

```
def dibujar():
    x=random.randint(0,ancho)
    y=random.randint(0,alto)
    r=random.randint(1,min(alto,ancho))/2
    cv.create_oval(x-r,y-r,x+r,y+r)
v=Tk()
b=Button(v,text="circunferencia",command=dibujar)
alto=200; ancho=200
cv=Canvas(v,width=ancho,height=alto)
b.pack(); cv.pack()
v.mainloop()
```

Problema: Dibujar la siguiente figura (la de la izquierda, el cuadrado girado en α)




```
cv=Canvas(…)
h2=h+int(L*math.cos(alfa*math.pi/180))
v2=v-int(L*math.sin(alfa*math.pi/180))
cv.create_line(h,v,h2,v2)   #primera línea
h3=...; v3=...
cv.create_line(h2,v2,h3,v3) #segunda línea
h4=...; v4=...
cv.create_line(h3,v3,h4,v4) #tercera línea
cv.create_line(h4,v4,h,v)   #cuarta línea
```

Solución 2. Usando la geometría de la “tortuga” (de Logo).

Suponemos que tenemos una clase Tortuga con métodos girar y avanzar.

Luego, creamos un boton que ejecute la siguiente función como acción:

```
def dibujarCuadrado()
  #ubicar tortuga al centro de cv
  t=Tortuga(W/2,H/2,cv)
```

```
    #girar la tortuga en angulo alfa
    t.girar(45)
    #repetir 4 veces
    for i in range(4):
        #avanzar tortuga en L(dibujando linea)
        t.avanzar(50)
        #girar tortuga en 90º
        t.girar(90)
```

Problema propuesto (opcional).

1. Completar el ejemplo anterior, agregando lo que falta para crear la ventana y el boton.
2. Escribir y probar la clase Tortuga. (Parta del ejemplo de más abajo).

```
from tkinter import *
from Tortuga import *

def dibujarCuadrado()
  # ubicar tortuga al centro de cv
  t=Tortuga(W/2,H/2,cv)
  # girar la tortuga en angulo alfa
  t.girar(45)
  # repetir 4 veces
  for i in range(4):
      #avanzar tortuga en L(dibujando linea)
      t.avanzar(50)
      #girar tortuga en 90º
      t.girar(90)

# OJO: COMPLETAR
# Crear ventana
# Crear Boton, asociarle acción dibujarCuadarado()
# Crear Canvas
# Concluir llamando a ventana.mainloop()


# Modulo Tortuga.py

#h,v: int coordenadas horizontal y vertical
#cv: Canvas
#angulo: float (en radianes)
class Tortuga:
    #__init__: int int Canvas -> Tortuga
    #crea tortuga en coords(x,y)de canvas cv con angulo 0
    #ej: t=Tortuga(W/2,H/2,cv)
    def __init__(self,x,y,cv):
```

```
def __init__(self,x,y,cv):
    self.__h = x; self.__v = y
    self.__cv = cv
    self.__angulo = 0

#girar: int -> None
#girar tortuga en angulo x (sumar x al angulo)
#ej: t.girar(45)
def girar(self,x):
    # Completar




#avanzar: int -> None
#avanzar tortuga x pixeles (dibujando línea)
# ej: t.avanzar(50)
def avanzar(self,x):
    # Completar
```