RUTAS RELATIVAS, SYS, OS

Antes de continuar, quiero mencionar brevemente la ruta de búsqueda de bibliotecas. Python busca en varios sitios cuando intenta importar un módulo. Específicamente, busca en todos los directorios definidos en sys.path. Esto es simplemente una lista, y puede verla y/o modificarla fácilmente con los métodos estándar de una lista (conocerá las listas más adelante, en este capítulo).

    import sys

    import os

    print(sys.path)

['c:\\CURSOS\\python\\Seccion 56', 'C:\\Program Files\\Python39\\python39.zip', 'C:\\Program Files\\Python39\\DLLs', 'C:\\Program Files\\Python39\\lib', 'C:\\Program Files\\Python39', 'C:\\CURSOS\\python\\Seccion 56\\.venv', 'C:\\CURSOS\\python\\Seccion 56\\.venv\\lib\\site-packages']

Cuando se intentan incorporar imágenes o recursos, puede generar problemas al intentar ejecutar archivos desde fuera del ide.

Una forma es utilizar la lista **sys.path** con el índice 0 **sys.path[0]**  que indica la ruta absoluta del archivo actual y unirle la relativa para que se realice la importación correctamente.

Ejemplo:

    root.iconbitmap(os.path.join(sys.path[0], ' \\resources\py.ico'))

De este modo podré acceder correctamente al archivo ubicado en la ruta deseada.

Por su parte, la funcion **os.path.join**

*Help on function join in module ntpath:*

*join(path, \*paths)*

*# Join two (or more) paths.*

COMPONENTES Y PROPIEDADES EN TKINTER

Para editar componentes puede utilizarse **tkinter**  o el “tema” llamado **ttk**

Estas son sus importaciones:

import tkinter as tk

from tkinter import ttk

Esta sería una forma de crear un botón con ttk:

btn1 = ttk.Button(root, text="nuevo boton",

                  command=click\_btn1)

de igual modo, podría hacerlo con tkinter (en este caso tk, por como se le ha importado)

lbl = tk.Label(text=f"{variable}", bg="white", fg="blue")

los elementos creados con tk dan la posibilidad de personalización de ciertas propiedades (como bg y fg) que no están disponibles para elementos creados con ttk.

**COLOCANDO ELEMENTOS EN LA VENTANA**

Para colocar elementos en la ventana, puede utilizarse el método pack()

    btnClick = ttk.Button(root, text="click me outta me", command=click\_btn1)

    btnClick.pack()

Otra forma de organización es el grid. Un concepto muy similar al de CSS

Una forma de configurar el grid de nuestra ventana es con:

root.rowconfigure(0, weight=1)

root.rowconfigure(1, weight=5)

root.columnconfigure(0, weight=3)

root.columnconfigure(1, weight=7)

donde root es:

root = tk.Tk()

Ahora los elementos se colocan usando el método grid:

btn3 = ttk.Button(root, text="boton 3")

btn3.grid(row=0, column=1, sticky='NSWE')

**LA PROPIEDAD STICKY**

Esta propiedad se utiliza cuando los elementos se disponen con el método grid y determina “hacia donde” se orienta el elemento.

N -> North

S -> South

W -> West

E -> East

Los elementos no solo pueden tomar un valor sino que pueden representar, “ir de un lado a otro” con los valores “NS” y “WE”. Por ultimo cabe resaltar que se puede especificar “NSWE” para hacer que “ocupe todo el espacio”

**PADDING**

Para utilizar el padding en componentes, se utilizan las propiedades ipady, ipadx para el padding interno de los componentes y padx, pady para el externo.

El concepto es el mismo que el de css.

btn3 = ttk.Button(root, text="boton 3")

btn3.grid(row=0, column=1, sticky='NSWE', ipadx=10, ipady=10, padx=15, pady=15)

COMPONENTE ENTRY

El componente entry es un input. Se declara como:

input1 = ttk.Entry(root, width=20, justify=tk.LEFT)

Algunas propiedades a mencionar:

*# show="\*" -> hace que solo se muestren \* en el input*

*# state = tk.DISABLED / tk.NORMAL /readonly*

Una forma de modificar el contenido de un entry es:

    input1.insert(0, '')

donde 0 es el índice a insertar y el string a continuación es el texto a introducir.

Para eliminar el contenido puede usarse de igual forma:

    input2.delete(0, tk.END)

y para recuperar el content:

    entrada\_name = input1.get()

**ENLAZANDO UNA VARIABLE A UN INPUT**

Para manejar una variable enlazada a un Entry, se procede:

variable\_input1 = tk.StringVar(value='')

input1 = ttk.Entry(root, width=20, justify=tk.LEFT,

                   textvariable=variable\_input1)

Con la primera línea se define un objeto de <class 'tkinter.StringVar'> y con la propiedad textvariable se enlaza dicha variable al componente.

Ahora para recuperar o modificar el valor se procede:

    variable\_input1.set('valor default')

or

variable\_input1.get()

**MENSAJES DE ALERTA**

Para utilizar mensajes de alerta se importa messagebox:

from tkinter import ttk, messagebox, Menu

la forma de lanzar uno es

        messagebox.showinfo('info', 'Guardado correctamente')

donde el primer param es el titul ode la ventana del mensaje y el segundo es el texto de la ventana. El método que se invoca determina el icono y demás configuraciones (como el sonido)

También están disponibles

.showerror, .showwarning

**MENUS**

Para utilizar menus también es necesario importarle

En este caso se crea una funcion para crear el menu, la misma se llama justo antes de **root.mainloop()**

    menu\_principal = Menu(root)

Con esta línea se crea un menú y se asocia con la ventana principal.

El siguiente es un ejemplo de la creación de un submenú:

    submenu\_archivo = Menu(menu\_principal, tearoff=0)

como se vé, se asocia al menu principal, pero la clase que se instancia es la misma.

Lo siguiente crea un “comando” y un separador para este submenú.

En el command, con la propiedad **command** podría indicarse la funcion o declararse el lambda relacionado con ese menu.

    submenu\_archivo.add\_command(label='Nuevo')

    submenu\_archivo.add\_separator()

Por ultimo se agrega al menu principal:

    menu\_principal.add\_cascade(menu=submenu\_archivo, label='Archivo')

Y se agrega el menu al root.

    root.config(menu=menu\_principal)