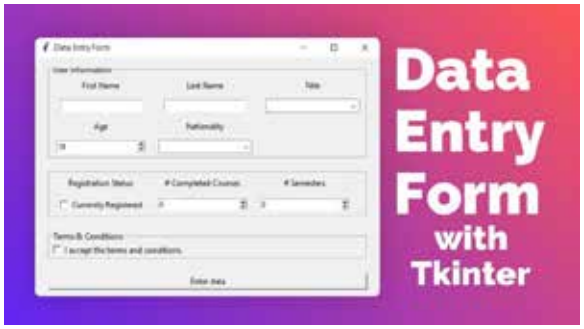
PYTHON TKINTER

F. Vargas Navarro
Catedrático UCENFOTEC
Auditor Certificado ISO/IEC 27000
Senior Member IEEE (francisco.vargas@cpic.cr)

INTRODUCCIÓN

Todos sabemos acerca de la popularidad del lenguaje de programación “PYTHON”, un lenguaje interpretado muy flexible y amigable. Una de las fortalezas más grandes es la versatilidad ante diferentes requerimientos y estos se logra gracias a la comunidad de soporte que está día a día publicando nuevas librerías, así como mejorando las actuales. Una de estas librerías se llama “TKINTER”. Esta permite el manejo de ventanas de manera simple y rápida. En este artículo presentaremos un breve resumen de las principales funcionalidades de “TKINTER”.



INSTALACIÓN

Iniciaremos el proceso de instalación con la carga del módulo, ya que esta librería no forma parte del “Core” de “PYTHON”, así que debemos utilizar el programa de instalación que se descarga al instalar el lenguaje, para cargar la libraría. El comando desde la consola es:

```
>pip install tkinter
```

```
Una vez instalado, podremos pro-
bar
agregando una línea de código a
nuestro
programa:
```

```
import tkinter as tk
```

Es importante recordar que el “as” es opcional. Con este comando, cargaremos el módulo y comprobaremos que todo está correcto.

COMANDOS

Las ventanas en un sistema “Microsoft WINDOWS” siguen un patrón que es muy conocido: tres botones en la parte superior derecha (minimizar, maximizar y cerrar). En la parte superior izquierda aparece un logo y alguna otra información pertinente. A esto llamaremos “TOP”. Debajo del “TOP”, encontramos el área de trabajo donde ponemos nuestro diseño de pantalla. A esta área del llamaremos el “Workspace WS”. “TKINTER” se apoya en el uso de “WIDGET” para el diseño en el WS.

Otro aspecto por considerar es que, a diferencia de la programación tradicional serial, TK se apoya en la programación por eventos, creando un ciclo sin fin (“Do forever”) y haciendo uso de un estado del sistema operativo conocido como “WAIT ON A EVENT”, estado en el cual no se consume procesador y el proceso en dicho estado puede responder a las interrupciones que se activen.

El primer paso es crear la ventana; usaremos una variable, “root”, que servirá de puntero al objeto en memoria, que representa el WS:

```
root=tk.Tk()
```

A partir de este momento, TK nos da una ventana configurada según el estándar de “Windows” antes descrito.

Podemos configurar algunos aspectos de visualización de la WS, colores, tamaño, título; vemos algunos:

ISSN: 2015-5392

```
root.geometry("1350x400")
root.title("Programa de Pruebas"
root.iconbitmap(r"I:\CPIC\Avanzado\
logo.ico")
```

- "geometry" nos permite dimensionar el WS.
- "title" se refiere al rótulo que aparece en TOP a la par del logo.
- "iconbitmap" permite poner una imagen personalizada en el espacio del logo. Es archivo debe ser tipo ".ico", es decir, tipo ícono.

Finalmente, debemos poner la instrucción que activa el ciclo a la espera de interrupciones:

```
root.mainloop()
```

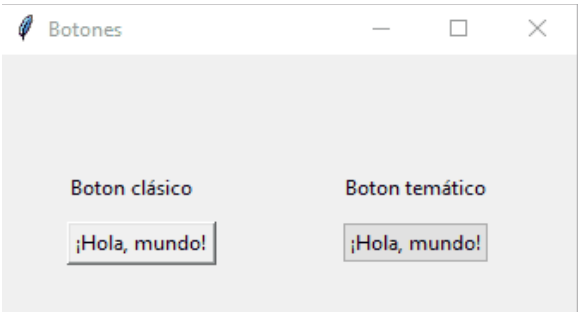
En resumen, este sería el código:

```
import tkinter as tk

root=tk.Tk()
root.geometry("1350x400")
root.title("Programa de Pruebas"
root.iconbitmap(r"I:\CPIC\Avanzado\
logo.ico")

# área de programación

root.mainloop()
```

En el área de programación ponemos todas aquellas instrucciones necesarias para el control de las interrupciones programadas.

Es importante considerar que nos enfrentamos ante un cambio de paradigma de programación, pues ya no tenemos comandos como: "INPUT", "READ", que detienen la ejecución del programa hasta presionar la tecla "ENTER".

Ahora nuestro programa está en un estado de espera sin consumir procesador y responderá ante una interrupción del sistema.



A una de estas interrupciones básicas le llamamos botones.

Al presionar el botón, se provoca una interrupción del sistema que causa que el programa se active según el siguiente comando:

```
boton1=tk.Button(root, text=
"Proceso A", command=funcion1,
fg="blue",bg="#2BFC70", width=
20, height=2)
```

Esta instrucción crea un botón con el texto indicado en "TEXT" en su interior y lo ubica en el WS indicado en "ROOT". "COMMAND" especifica qué función se ejecutará al presionar el botón y al producirse la interrupción.

La función es del tipo:

```
def funcion1 ()
.....
return
```

Los otros parámetros se relacionan con la estética del botón.

Como se puede observar, la secuencia de ejecución se controla a través de botones asociados con funciones.

Otro comando que es importante conocer es el relacionado con las ventanas adicionales o "POPUP Windows". El comando es:

```
pw=tk.Toplevel(root)
```

Este indica que se abre una nueva ventana asociada a la variable pw y que depende del WS root. Esta será una nueva área de trabajo que podremos cerrar al finalizar con:

```
pw.destroy()
```

Los mensajes se controlan con una librería del módulo llamada "messagebox". Podemos incorporarla directamente así:

```
from tkinter import messagebox
```

Y se usa de la siguiente forma:

```
messagebox.showerror ("ERROR","-
Falla al detectar BD")
messagebox.showinfo("AVISO","BD
Cerrada")
messagebox.showwarning("AVI-
SO","BD está cerrada")
```


ISSN: 2015-5392

El primer parámetro es el título y el segundo es el mensaje. Se abrirá una ventana emergente con un botón de “ACEPTAR”.

Solo queda pendiente la aceptación de los datos y su presentación.

Para leer un dato, debe usar un “WIDGET” de la siguiente manera:

```
cedulaS=tk.Entry(pw, width=50)
cedulaS.place(x=20,y=20)
```

Mediante “Entry”, se captura de pantalla para, luego, extraer el valor del objeto con:

```
cedula=cedulaS.get()
```

Es importante entender que el programa no detiene su ejecución en el “Entry”, simplemente es una zona en la cual se pueden digitar datos en cualquier momento.

Además, siempre debemos indicar en qué WS vamos a desplegar el “WIDGET”

Para mostrar algún rótulo, podemos usar lo siguiente:

```
etiqueta1=tk.Label(root, text=
”CPIC Prueba”)
etiqueta1.pack()
```

“LABEL” nos permite mostrar un texto en la WS y “pack” lo acomoda automáticamente. Si queremos una ubicación específica, deberemos usar:

```
“etiqueta1.place(x=25, y=220)”
```

En vez de “etiqueta1.pack()”

CONCLUSIONES

Este artículo no pretende ser un manual de referencia de TK, es simplemente una introducción. Cada “Widget” aquí mencionado tiene muchas más opciones (parámetros) y también hay una cantidad amplia de “Widgets” que podemos utilizar.

REFERENCIAS

Python Software Foundation. (2024). *El tutorial de Python. https://docs.python.org/es/3/tutorial/*

Python Software Foundation. (2024). Interfaces gráficas de usuario con Tk. *https://docs.python.org/es/3/library/tk.html*

Refsnes Data. (2024). Python Tutorial. *https://www.w3schools.com/python/*

ACERCA DEL AUTOR

El autor es auditor internacional ISO/IEC 27000, catedrático de UCENFOTEC, consultor con experiencia nacional e internacional en diferentes áreas de tecnología, incluyendo la ciberseguridad, telecomunicaciones, robótica, analítica de datos y mecatrónica. Es profesor universitario desde 1985 y consultor internacional por más de 20 años. Es miembro del comité de ética del Colegio de Profesionales en Informática y Computación (CPIC), presidente de IEEE COMSOC Costa Rica y miembro Senior de IEEE; además, es instructor CISCO certificado CCNA.

ISSN: 2015-5392