Ximena Céspedes Quesada

Tutorial interfaz grafica para una calculadora sencilla en Python.

En el siguiente tutorial, se creara una interfaz gráfica para una calculadora sencilla, mediante la biblioteca “customtkinter” en Python 3. Seguidamente, se explicara paso a paso como realizar el código:

1. **Importaciones:**

   Primeramente en la terminal de comandos de Windows, instale el paquete “pip install customtkinter”. Una vez instalado, se importa la biblioteca “customtkinter”, la cual nos ayudara a crear una interfaz mas personalizada utilizando Tkinter, por lo tanto también es necesario importar los módulos de Tkinter y messagebox, tal y como se observa en la figura 1.

```
import customtkinter
from tkinter import *
from tkinter import messagebox
```

Figura 1: Importaciones.

2. **Creación de la clase:**

   Ahora bien, para representar a la calculadora, se define la clase “Calculator”,se llama el método “$_{init}$,, el cual se conoce como el método constructor de la clase y este se va a utilizar para inicializar la calculadora y poder configurar la interfaz.

```
class Calculator:
    def __init__(self):
```

Figura 2: Definición de la clase.

3. **Inicialización de la interfaz:**

   En tercer lugar, se creara la ventana principal de la aplicación mediante la creación de la instancia “customtkinter.CTk”. Seguidamente, como se muestra en la figura 3, se establece el titulo, el tamaño y el color del fondo de la ventana.

```
self.app = customtkinter.CTk()
self.app.title("Calculadora")
self.app.geometry("250x350")
self.app.config(bg="#000000")
```

Figura 3: Inicialización de la interfaz.

4. **Variables:**

   Como se muestra en la figura 4. Para lograr almacenar los datos numéricos introducidos por el usuario, se inicializa una variable llamada “equation” como una cadena vacía. Seguidamente, se inicializa la variable “i”, para poder realizar el seguimiento del índice de inserción en la caja de resultados. Por ultimo, se tiene que en la variable “letra1”, se define la fuente que se va a utilizar en la interfaz.

```
self.equation = ""
self.i = 0
self.letra1 = ('Arial', 20, 'bold')
self.create_widgets()
```

Figura 4: Variables.

5. **Creación de widgets:**

   Se define la función “$create_widgets$”, la cual sera la encargada de crear los elementos de la interfaz, tales como la caja de resultados y los botones. Esta función se llamara en el constructor de la clase para así poder crear los widgets al momento que se inicia la calculadora. Por lo tanto, el código que se tiene que tener hasta el momento se observa en la figura 5.

```
1  import customtkinter
2  from tkinter import *
3  from tkinter import messagebox
4
5
6  class Calculator:
7      def __init__(self):
8          #Para crear la ventana.
9          #Ajustar la fuente de la letra
10         self.app = customtkinter.CTk()
11         self.app.title("Calculadora")
12         self.app.geometry("250x350")
13         self.app.config(bg="#000000")
14         self.equation = ""
15         self.i = 0
16         self.letra1 = ('Arial', 20, 'bold')
17         self.create_widgets()
18
19     def create_widgets(self):
```

Figura 5: Primera parte del código.

Seguidamente, en la misma función definida anteriormente se realiza la creación de la caja de resultados. Para esto es necesario crear un objeto mediante “customtkinter.CTkEntry”, el cual va a representar la caja de resultados. Para esto se configuran los atributos de la misma, tales como la fuente, el ancho, el color del texto y el borde. Por ultimo, para posicionar la caja de resultados en la ventana de la interfaz, se coloca en la posición (0, 10), como se observa en la figura 6.

```
19      def create_widgets(self):
20          #Para crear la cajita de resultados
21          #(La ventana donde queremos la nueva cajita,lo que queremos que este escrito,
22          #la fuente, tamaño, colorbg,borde)
23          self.result_entry = customtkinter.CTkEntry(
24              self.app,
25              textvariable="",
26              font=self.letra1,
27              width=250,
28              fg_color="#FFFFFF",
29              border_color="#FFFFFF"
30          )
31          #posicionar la cajita
32          self.result_entry.place(x=0,y=10)
```

Figura 6: Creación de la caja de resultados.

6. **Creación de los botones:**

Ahora bien, en la misma función anterior, se definen los botones.Para crear los botones de la calculadora, se realiza mediante “customtkinter.CTkButton”.

Primeramente se crean los botones de las operaciones, los cuales van a seguir una misma linea y en este caso solo cambiaran ciertos comandos dependiendo de cual botón se este creando. Tal y como se muestra en la figura 7, se le configuran los atributos, tales como el comando asociado, el texto que se quiere mostrar en el botón, la fuente, el ancho, el alto, los colores del texto y del botón, por ultimo se coloca el botón en la ventana principal de la calculadora en la posición que se desee.

```
37          self.Button1 = customtkinter.CTkButton(
38              self.app,
39              command=self.clear,
40              text="C",
41              font=self.letra1,
42              width=50,
43              height=2,
44              fg_color="#83838B",
45              hover_color="#83838B"
46          )
47          self.Button1.place(x=10,y=60)
```

Figura 7: Creación de los botones de operación.

Seguidamente, se procede a crear los botones para los números, al igual que en el caso anterior, se definen diferentes atributos para el mismo y se cambian según el numero que se este definiendo, tal y como se muestra en la imagen 8.

```
147         self.Button9 = customtkinter.CTkButton(
148         self.app,
149         command=lambda:self.show("1"),
150         text="1",
151         font=self.letra1,
152         width=50,
153         height=2,
154         fg_color="#2e2a27",
155         hover_color="#2e2a27"
156         )
157         self.Button9.place(x=10,y=240)
```

Figura 8: Creación de los botones numéricos

7. **Funcionamiento de los botones:**

Para que los botones funcionen al momento que se llaman al hacerle clic, se definen varias funciones.

Primeramente se define la función “show”, la cual se va a llamar al momento de darle clic a un botón numérico o de operación, y este va a mostrar el valor que le corresponde al botón que fue presionado en la caja de resultados. En la figura 9, se muestra el código empleado para la creación de la función “show”.

```
268     def show(self, value):
269         if(value=="%"):
270             value="/100"
271         elif(value=="√"):
272             value="**0.5"
273         self.equation+=value
274         self.result_entry.insert(self.i,value)
275         self.i=self.i+1#añado el nuevo valor a el nuevo indice
276         self.result_entry.delete(0, END)
277         self.result_entry.insert(0, self.equation)
```

Figura 9: Creación de la función “show”

Seguidamente, como se observa en la figura 10, se define la función “clear”, la cual se llama en el momento que se hace clic en el botón “C” y esta borra los valores que se tienen en la caja de resultados.

```
279     def clear(self):
280         self.equation = ""
281         self.result_entry.delete(0, END)
```

Figura 10: Creación de la función “clear”

Por ultimo, se define la función “calculate”, la cual nos va a ayudar a evaluar la expresión matemática al momento de hacer clic en el botón “=”, en esta misma función, se toma en cuenta el manejo de errores, previniendo que el usuario pueda ingresar una expresión errónea. Se puede observar en la figura 11.

```
283     def calculate(self):
284         try:
285             self.result = eval(self.equation)
286             self.equation = str(self.result)
287             self.result_entry.delete(0, END)
288             self.result_entry.insert(0, self.result)
289         except ZeroDivisionError:
290             messagebox.showerror("Error", "No se puede dividir entre cero")
291             self.equation = ""
292             self.equation = str(self.result)
293         except:
294             messagebox.showerror(title="Error", message="Operación inválida")
295
```

Figura 11: Creación de la función “calculate”

8. **Creación del bucle principal de la aplicación:**

Como se observa en la figura 12, se define la función “mainloop”, la cual esta encargada de iniciar el bucle principal de la aplicación. Por ultimo, se crea una instancia llamada “Calculator” y se llama a la función “mainloop” para así iniciar la calculadora.

```
297         def mainloop(self):
298             self.app.mainloop()
299
300     if __name__ == '__main__':
301         calculator = Calculator()
302         calculator.mainloop()
```

Figura 12: Creación del bucle principal

9. **Resultado Final:**

Por ultimo, en la figura 13, se observa el resultado final de la interfaz de la calculadora.

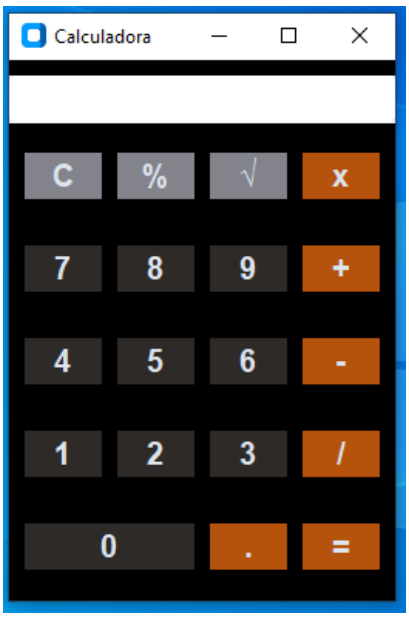

Figura 13: Resultado final