**Documentación del Proyecto: Calculadora Científica en Python**

**1. Introducción**

Este proyecto consistió en el desarrollo de una calculadora científica con interfaz gráfica utilizando **Python** y la librería **Tkinter**. El objetivo principal fue construir una herramienta capaz de realizar operaciones matemáticas básicas y avanzadas, incorporando también un historial de cálculos y la opción de trabajar en grados o radianes.

**2. Objetivo del Proyecto**

* Implementar una **calculadora completa** con funciones científicas.
* Desarrollar la interfaz con **Tkinter** para simular una calculadora física real.
* Aplicar conceptos de **POO (Programación Orientada a Objetos)**, manejo de excepciones y uso de librerías matemáticas.
* Añadir funcionalidades extra como **historial de operaciones** y **manejo de errores**.

**3. Herramientas Utilizadas**

* **Python**: Lenguaje de programación principal.
* **Tkinter**: Librería estándar de Python para interfaces gráficas.
* **Math**: Librería matemática para implementar funciones trigonométricas, logaritmos, potencias, etc.

**4. Desarrollo Paso a Paso**

**4.1. Creación de la Clase Principal**

Se creó la clase Calculator, heredando de tk.Tk, lo que permitió manejar toda la lógica y la interfaz en una sola estructura:

class Calculator(tk.Tk):

def \_\_init\_\_(self):

super().\_\_init\_\_()

# Configuración inicial de la ventana

Aquí se definieron:

* **Expresión actual** (self.expression).
* **Modo de ángulos** (self.deg\_mode).
* **Historial de cálculos** (self.history).

**4.2. Diseño de la Interfaz**

* Ventana principal de **400x600 píxeles**, con fondo oscuro.
* **Campo de entrada** para mostrar expresiones y resultados.
* **Botones** organizados en una cuadrícula (9 filas × 6 columnas).
* Diferenciación de colores para resaltar funciones especiales (ejemplo: DEL, AC, =).

**4.3. Implementación de Funcionalidades**

* **append()**: Añade caracteres a la expresión actual.
* **clear()**: Borra todo el contenido.
* **backspace()**: Elimina el último carácter.
* **sign()**: Cambia el signo del número actual.
* **toggle\_mode()**: Alterna entre **DEG** y **RAD**.
* **calculate()**: Evalúa la expresión matemática con seguridad, utilizando un diccionario restringido de funciones.

**4.4. Funciones Matemáticas**

Se implementaron funciones trigonométricas y logarítmicas adaptadas al modo de ángulos:

def sin(x):

return math.sin(math.radians(x)) if self.deg\_mode else math.sin(x)

También se añadieron:

* **log()**, **log10()**, **exp()**, **sqrt()**, **factorial()**
* Constantes: **π**, **e**

**4.5. Historial de Cálculos**

* Cada operación resuelta se almacena en self.history.
* Al presionar el botón *Historial*, se abre una nueva ventana mostrando todos los cálculos.
* Posibilidad de limpiar el historial.

**4.6. Manejo de Errores**

* Uso de try-except en calculate().
* Errores como división por cero, valores fuera de dominio o expresiones inválidas muestran "Error" sin interrumpir el programa.
* Se evita acceso no autorizado deshabilitando \_\_builtins\_\_ en eval().

**5. Pruebas y Resultados**

* **Operaciones Básicas**: 5+3=8, 10-4=6, 7\*6=42.
* **Trigonometría**: sin(30°)=0.5 en modo grados.
* **Logaritmos y potencias**: log(10)=2.3025, 2³=8.
* **Errores**: división por cero o entradas inválidas → "Error".

**6. Conclusiones**

* Se logró implementar una calculadora **científica funcional y segura**.
* El uso de **Tkinter** permitió crear una interfaz amigable y similar a las calculadoras físicas.
* El proyecto integró programación orientada a objetos, librerías externas y manejo de errores.
* El historial y el modo de ángulos aportaron **valor añadido** respecto a una calculadora básica.