Proyecto Calculadora

Marzo 5 2025

Manuel Santiago Bastidas Gaona, Nicolas Palacios Quimbayo

I. Introducción

Este proyecto tiene como objetivo el desarrollo de una calculadora interactiva en Python que permita realizar operaciones matemáticas básicas como suma, resta, multiplicación y división, con una interfaz de usuario sencilla y un historial de operaciones que facilite el seguimiento de los cálculos realizados. La calculadora podrá ser utilizada tanto en su versión de línea de comandos como, opcionalmente, con una interfaz gráfica basada en Tkinter para una experiencia más amigable.

II.Propósito

El propósito de este proyecto es desarrollar una calculadora avanzada en Python que no solo permita realizar las operaciones matemáticas básicas (suma, resta, multiplicación y división), sino que también integre funciones científicas y especializadas para satisfacer las necesidades de usuarios más exigentes. Esta calculadora será capaz de realizar cálculos de sumatorias, integrales y operaciones con matrices, además de graficar funciones matemáticas y proporcionar el código de colores de resistencias. El objetivo es ofrecer una herramienta completa que combine la simplicidad de una calculadora básica con la potencia y versatilidad de una calculadora científica, todo ello mediante una interfaz accesible y fácil de usar. Además, este proyecto busca poner en práctica conocimientos avanzados de programación en Python, incluyendo manejo de bibliotecas matemáticas, visualización de datos y diseño de interfaces interactivas, con el fin de crear una aplicación robusta y multifuncional.

III.Alcance Del Producto

La calculadora avanzada es una aplicación diseñada para realizar operaciones matemáticas complejas y especializadas, orientada a estudiantes, ingenieros y profesionales en disciplinas científicas y técnicas. Su funcionalidad abarca desde cálculos básicos hasta herramientas avanzadas que facilitan la resolución de problemas matemáticos y eléctricos.

V.Clases y Caracteristicas de Usuarios

Clasificación de usuarios

1. Estudiantes
2. Profesionales e Ingenieros

características de usuarios

1. Características de estudiantes
    - Interfaz intuitiva y fácil de usar.
    - Funcionalidades de integración, sumatoria y manipulación de matrices para resolver ejercicios académicos.
    - Representación gráfica de funciones matemáticas en 2D.
    - Explicación detallada de ciertos cálculos para facilitar el aprendizaje.

2. Características de profesionales e Ingenieros::
    - Cálculos precisos de integrales, matrices y ecuaciones complejas.
    - Soporte para resolución de sistemas de ecuaciones mediante matrices.

- Herramienta de código de colores de resistencias para facilitar el diseño de circuitos eléctricos.
- Exportación de resultados en formatos compatibles con reportes técnicos.

V.Requerimientos Funcionales

La aplicación cuenta con distintas interfaces y funciones adicionales a la realizaciòn de operaciones matemáticas dependiendo de las necesidades del usuario

1. registro del usuario: La aplicación cuenta con un sistema de registro e inicio de sesiòn, cuando la app detecta que un usuario desea registrarse se despliega una ventana que pide los siguientes datos:
    - nombre de usuario
    - contraseña

2. inicio de sesión: si el usuario ya tiene una cuenta registrada en la base de datos del programa puede iniciar sesión con el nombre de usuario y la contraseña registrados en la base de datos

3. Historial del usuario: La aplicación registra todas las operaciones realizadas por el usuario y las registra en la base de datos, el usuario puede tener acceso a su historial desde la aplicación

4. Modo Oscuro: La aplicación cuenta con modo oscuro para que el usuario tenga la posibilidad de personalizar el color de la interfaz de la calculadora según sus preferencias.

5. Calculadora de matrices: la aplicación cuenta con una interfaz independiente para el cálculo de matrices, en este mismo se pedirá al usuario que ingrese el tamaño de la matriz y la operación que desea realizar con la misma.

6. Calculadora científica: la aplicación cuenta con las operaciones que una calculadora científica debe tener como la operación de integrales, derivadas, sumatorias etc.
.
7. Código de colores de resistencias: la aplicación cuenta con una interfaz especializada en el cálculo de resistencias según sus bandas de colores.

8. Gráficas de funciones: La aplicación cuenta con una función que le permite realizar gráficas de las funciones operadas en la misma.

VI.Requerimientos No Funcionales

1. La aplicación no permite ingresar a una cuenta con contraseñas incorrectas.

2. la calculadora debe resolver operaciones matemáticas sencillas y complejos en fracciones cortas de tiempo

3. La calculadora de matrices debe poder operar matrices de gran tamaño sin inconvenientes

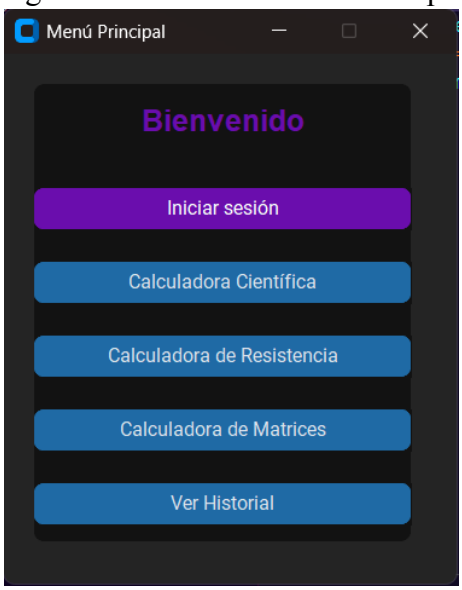
4. La interfaz debe ser intuitiva, fácil de usar y amigable con el usuario, con botones de fácil acceso y debe poder usarse con las teclas del teclado.

5. La aplicación debe contar con botones de fácil acceso y debe poder usarse con las teclas del teclado.

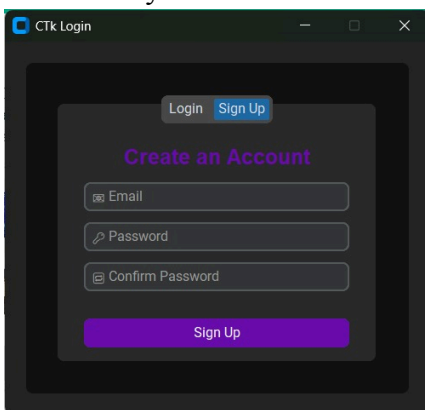
6. La aplicación debe almacenar automáticamente el historial al realizar cualquier operación

VII. Manual de Usuario

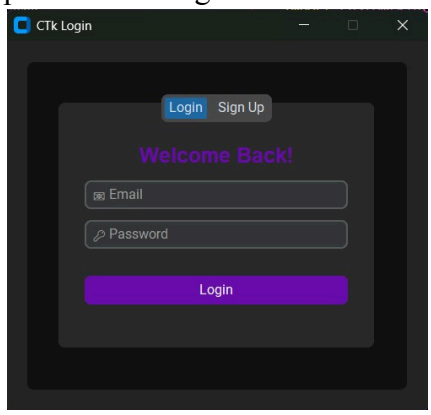
1. inicio: Al iniciar la aplicación el usuario puede acceder a la mayoría de las funciones de la misma a excepción de historial, para tener acceso a este el usuario debe registrarse o iniciar sesión en la aplicación



2. Registro: Si el usuario desea registrarse en la aplicación debe acceder a la ventana de registro desde el apartado de inicio de sesión ubicado en el menú y llenar los campos con la información requerida (correo electrónico, contraseña y confirmación de la contraseña)

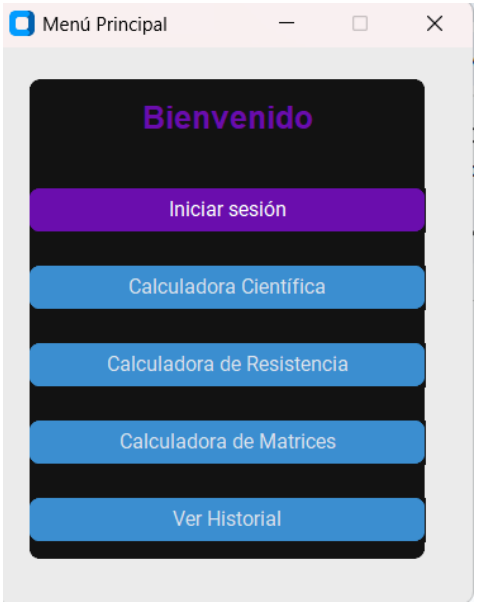


3. Inicio de sesión: Si el usuario ya posee una cuenta en la aplicación y desea iniciar sesión debe dirigirse al apartado de inicio de sesión ubicado en el menú y llenar los campos requeridos con la información de su cuenta previamente registrada.

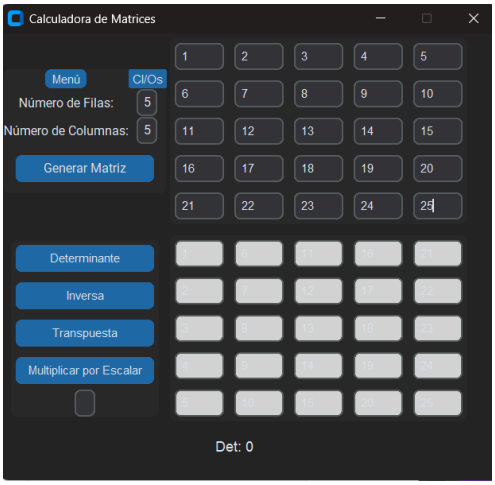


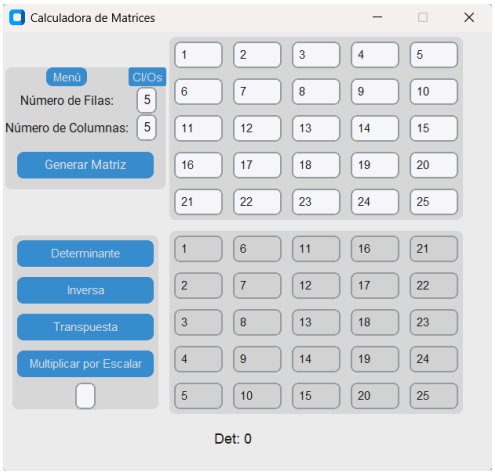
4. Calculadora científica: La calculadora viene configurada predeterminadamente como calculadora científica por lo que el usuario no requiere hacer nada para poder usarla en este modo.

5. Modo oscuro: Para alternar entre modo oscuro y modo claro en la calculadora hay un botón de fácil acceso que permite hacerlo en cualquiera de las calculadoras.

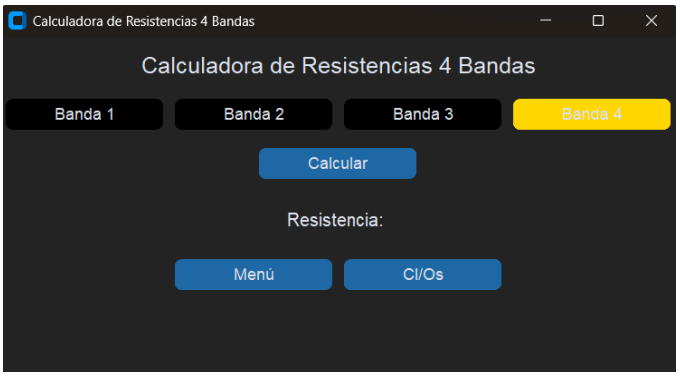


6. Calculadora de matrices: Para acceder a la calculadora de matrices debe hacerse desde el menú principal y eligiendo la opción correspondiente. para usarla se debe ingresar el tamaño de la matriz, llenar cada uno de los espacios y seleccionar la operación deseada

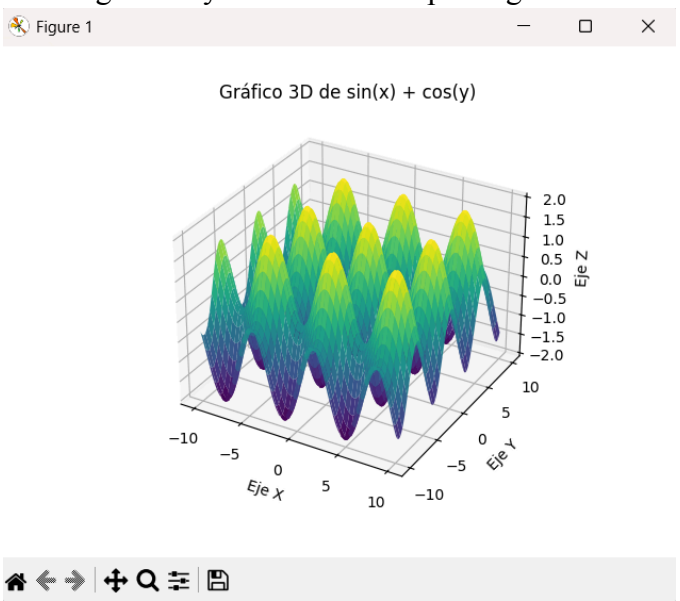


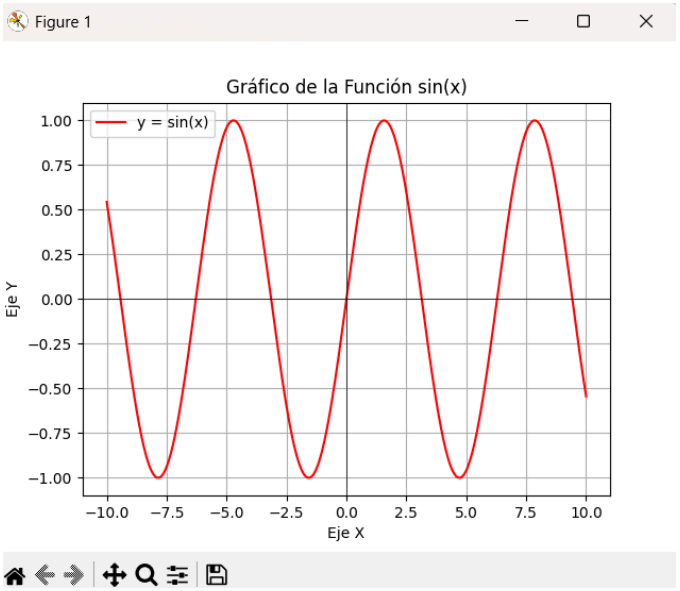


7. Calculadora de resistencias: Para acceder a la calculadora de resistencias debe hacerse desde el menú principal y eligiendo la opciòn correspondiente



8. Gráficas: Para generar una gráfica de una función debe darse click en el botón de graficar y seleccionar el tipo de gráfico





9) Historial de operaciones: Para acceder al historial de operaciones es necesario que el usuario haya iniciado sesión previamente. El usuario debe hacer click en el botón de “historial” y se desplegara una ventana emergente con su historial de operaciones

VIII. Implementación y Librerías

Firebase: Firebase de Google es una plataforma en la nube para el desarrollo de aplicaciones web y móviles. Está disponible para distintas plataformas (iOS, Android y web), con lo que es más rápido trabajar en el desarrollo.
Su función esencial es hacer más sencilla la creación de tanto aplicaciones webs como móviles y su desarrollo, procurando que el trabajo sea más rápido, pero sin renunciar a la calidad requerida.

En este proyecto fue utilizado como base de datos para almacenar los datos de registro y del historial de cada usuario

CustomTkinter: CustomTkinter es una biblioteca de interfaz de usuario de escritorio basada en Tkinter que ofrece widgets de aspecto moderno y totalmente personalizables. Con CustomTkinter, obtendrá una apariencia uniforme en todas las plataformas de escritorio (Windows, macOS, Linux).

En este proyecto fue utilizado para crear la interfaz gráfica de la calculadora, decidimos usar CustomTkinter y no Tkinter porque queríamos tener más opciones de personalización en cuanto a la interfaz de la aplicación.

CTkMessagebox: Es un módulo basado en CustomTkinter que permite crear ventanas emergentes con un diseño moderno y personalizable.

En este proyecto, se utilizó CTkMessagebox para mostrar mensajes de advertencia, información y confirmación, ya que proporciona una mejor integración con la estética de CustomTkinter y permite personalizar colores, iconos y botones.

NumPy: NumPy es el paquete fundamental para la computación científica en Python. Es una biblioteca de Python que proporciona un objeto de matriz multidimensional, varios objetos derivados (como matrices y matrices enmascaradas) y una variedad de rutinas para operaciones rápidas en matrices, incluidas operaciones matemáticas, lógicas, manipulación de formas, ordenamiento, selección, E/S, transformadas discretas de Fourier, álgebra lineal básica, operaciones estadísticas básicas, simulación aleatoria y mucho más.

En este proyecto fue utilizado para todas las funciones que tienen que ver con operar números reales y dar resultados numéricos.

Datetime: El módulo datetime proporciona clases para manipular fechas y horas. Si bien la implementación permite operaciones aritméticas con fechas y horas, su principal objetivo es poder extraer campos de forma eficiente para su posterior manipulación o formateo.

En este proyecto fue utilizado para registrar la fecha y la hora en el historial de firebase.

SymPy: SymPy es una biblioteca de Python para matemáticas simbólicas. Su objetivo es convertirse en un sistema de álgebra computacional (CAS) con todas las funciones, manteniendo el código lo más simple posible para que sea comprensible y fácilmente extensible. SymPy está escrito completamente en Python.

En este proyecto fue utilizado para operaciones más complejas como derivadas, integrales, sumatorias y matrices

Matplotlib: Matplotlib es una biblioteca completa para crear visualizaciones estáticas, animadas e interactivas en Python. Matplotlib hace que las cosas fáciles sean fáciles y las difíciles, posibles.

En este proyecto esta librería fue utilizada para la generación de gráficas de las funciones de la calculadora científica

Re: Este módulo proporciona operaciones de coincidencia de expresiones regulares similares a las que se encuentran en Perl. Tanto los patrones como las cadenas que se van a buscar pueden ser cadenas Unicode (str) así como cadenas de 8 bits (bytes).

En este proyecto fue izada esta librería interpretar operaciones y símbolos matemáticos para que el programa pudiera perarlos

Sys: Este módulo proporciona acceso a algunas variables utilizadas o mantenidas por el intérprete y

a funciones que interactúan fuertemente con el intérprete. Está siempre disponible. A menos que se indique explícitamente lo contrario, todas las variables son de solo lectura.

IX. Funciones y Código

Conexionfb:

En este archivo se encuentra el código necesario para que firebase pueda guardar datos de inicio de sesión de usuario e historiales de usuario

```
cred = credentials.Certificate("C:\\Users\\basti\\OneDrive\\Documentos\\programacion2024-2\\supercalculadoratilininsano-firebase-adminsdk-fbsvc-4b60c20dce.json")
firebase_admin.initialize_app(cred, {
    'databaseURL': 'https://supercalculadoratilininsano-default-rtdb.firebaseio.com/'})
```

Este código configura Firebase y genera las credenciales de autenticación de inicio de sesión

```
current_user_id = None

def set_current_user(user_id):
    global current_user_id
    current_user_id = user_id
```

Se define la variable global current_user_id y una función que la actualiza

```

def registrar_historial(tipo, operacion, resultado):
    if current_user_id:
        historial_ref = db.reference(f'users/{current_user_id}/history')
        nueva_entrada = {
            'action': f"{operacion} = {resultado}",
            'timestamp': datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
            'type': tipo
```

Si hay una sesión iniciada se actualiza el historial del usuario

```
def mostrar_historial_calculadora(entry):
    if current_user_id is None:
        respuesta = CTkMessagebox(title="Iniciar Sesión",message="Necesitas iniciar sesión para ver tu historial. ¿Deseas iniciar sesión ahora?")
        if respuesta:
            login_window = LoginApp()
            login_window.mainloop()
    else:
        historial_window = ctk.CTk()

historial_window.title("Historial de Cálculos")

historial_window.geometry("400x300")
```

si no hay una sesión iniciada se le pide al usuario que inicie sesión para poder acceder al historial, si si hay una sesion iniciada el usuario puede acceder a su historial de operaciones.

main.py

```
from menu import MainMenu

if __name__ == "__main__":
    app = MainMenu()
    app.mainloop()
```

Este codigo es el de iniciacion de la calculadora, lo que hace es importar la clase MainMenu, si el

archivo es ejecutado directamente crea un menu y su bucle de interfaz grafica

diccionarios:

En este archivo principalmente en los diccionarios para crear botones de la calculadora cientifica, las equivalencias de cada operacion tanto en numpy como en sympy y el codigo de colores para la calculadora de resistencias.

```
# Diccionario calculadora cientifica y grafica

def crear_diccionario_funciones(entry,etiqueta,cientifica_obj):
    return {
        "□": lambda: funciones_principales.informacion(),
        "AYUDA": lambda:
```

interfaces:

En este archivo se encuentran las 3 grandes clases de todo el documento:

1) Calculadora cientifica:

```
class Cientifica(ctk.CTk):
    def __init__(self):
        super().__init__()

        self.title("Calculadora Científica")
        self.geometry("414x595")
        self.resizable(False, False)

        self.modo_oscuro = True

        self.resultado = ctk.CTkLabel(self
```

Tiene una entrada de texto para poder ingresar las opciones matemáticas, usa un diccionario de botones para no tener que ocupar tanto codigo, posee 2 frames, uno para poder las funciones matematicas avanzadas en columnas de 6 y otro para los botones basicos de columnas de 5 para operaciones sencillas y los numeros, cuenta con dos funciones personalizada para cambiar entre modo claro y oscuro

```
    def cambiar_tema(self):
        self.modo_oscuro = not self.modo_oscuro
        nuevo_modo = "dark" if self.modo_oscuro else "light"

        ctk.set_appearance_mode(nuevo_modo)
        self.actualizar_colores()
```

la primera lo que hace es alternar entre modo claro u oscuro, determina el nuevo modo,lo aplica y actualiza a traves de la segunda funcion.

```
   def actualizar_colores(self):
        color_texto = "white" if self.modo_oscuro else "black"
        for widget in self.frame_funciones.winfo_children():
            if isinstance(widget, ctk.CTkButton):
                texto = widget.cget("text")
                if texto in ["□", "AYUDA", "HISTORIAL",
```

la segunda funcion define el color del texto, se mira que todo sean botones, de acuerdo a su texto se escoge su color, se actualizan y luego se actualizan los botones numericos.

2)Calculador Resistencias

```
class InterfazResistencia:
    def __init__(self, ventana_principal):
        self.ventana_principal = ventana_principal  # Guarda una referencia a la ventana principal
        self.ventana = ctk.CTk()

        self.ventana.geometry("600x300")  # Tamaño inicial
```

permite a traves de botones que cambian de color escoger los colores de las bandas de las resistencias de 4 bandas y calcula el valor de la resistencia de acuerdo a estos valores.

3)Calculadora Matrices

```
class Interfaz:
    def __init__(self, ventana):
        self.ventana = ventana
        self.ventana.title("Calculadora
de Matrices")

self.ventana.geometry("500x450")
        self.ventana.resizable(False,
False)

        # Frame izquierdo para botones
de operaciones
        self.frame_botones =
ctk.CTkFrame(self.ventana)
        self.frame_botones.grid(row=1,
column=0, padx=10, pady=10)
```

Se organiza en 4 frames, el primero posee la generacion de una matriz, el segundo la matriz en entradas, la tercera es las operaciones que se pueden hacer con esta matriz y la cuarta es como se veria la matriz.

Menú:

En este archivo se encuentra la interfaz del menú principal de la calculadora que se encarga de abrir las diferentes calculadoras y las ventanas emergentes de Inicio de sesión e historial.

```
def mostrar_historial_calculadora():
    def insertar_resultado(resultado):
```

esta parte del código se encarga de almacenar todos los resultados de operaciones obtenidos directamente en el el historial del usuario y los muestra en una ventana emergente generada por CustomTkinter

```
class MainMenu(ctk.CTk):
    def __init__(self):
        super().__init__()
```

En esta parte del código se encuentra la interfaz gráfica del menú principal.

Funciones_Principales

1)funcion agregar a pantalla

```
## Pone el caracter en las entradas
def agregar_a_pantalla(entry, valor):
    actual = entry.get()
    entry.delete(0, tk.END)
    entry.insert(0, actual +
str(valor))
```

Como su nombre lo indica esta funcion agrega a pantalla los valores de los botones que se presionen, esto a través de obtener que dice en la entrada en una variable, borrar lo que tenga esta y agregar el caracter a la variable que se habia guardado y volverlo a poner en la etiqueta.

2)manejar teclado

```
def manejar_teclado(event, entry,
etiqueta):
    tecla = event.keysym  # Obtener la
tecla presionada
    caracter = event.char
```

Esta función se encarga de permitir que se ingresen valores numéricos a la calculadora directamente desde el teclado y los muestra en la interfaz gráfica

3)Calcular

```
#funcion para dar resultado
def calcular(expresion, etiqueta,
entry):
    print("Expresión ingresada:",
expresion)  # Para depuración
    from conexionfb import
current_user_id
    try:
```

Esta funcion lo que hace es primero identificar si en la etiqueta ingresada hay caracteres especificos como sumatoria,derivada,integral o productoria, para redirijirlos a esta funcion, si no guarda el valor de esta y lo transforma en una cadena de caracteres donde la mayoría tiene formato numpy, para luego evaluarla a través de un eval, se actualiza la etiqueta y la entrada con el valor que da esta funcion y si hay usuario registrado lo envia a la base de datos, si no se puede desarrollar manda una ventana de error

4)Obtener expresión

```
def obtener_expresion(texto):

    x = sp.Symbol('x')  # Solo consideramos la variable 'x'

    # Convertir los operadores usando diccionario opersp
    for simbolo, reemplazo in diccionarios.opersp.items():
        if isinstance(reemplazo, str):
            texto = texto.replace(simbolo, reemplazo)
```

esta función se encarga de obtener los valores de derivada, limite, integral, sumatoria o gráfica y las convierte utilizando el diccionario de SymPy

5)Limpiar expresión

```
def limpiar_expresion(expresion):

    # Eliminamos los símbolos de derivada (d/dx, d/dy, d/dz)
    expresion = expresion.replace("d/dx", "").replace("d/dy",
```

lo que hace es borrar los símbolos de derivada, integral,sumatoria,productoria,limite.

6)Derivar:

```
def derivar(expresion):
    try:
        expresion = limpiar_expresion(expresion)  # Limpiar la expresión de símbolos no válidos
        expr, variables = obtener_expresion(expresion)
        if expr:
            derivada = sp.diff(expr, sp.symbols('x'))
            return f"Derivada: {derivada}"
```

Este código define una función llamada que calcula la derivada de una expresión matemática con respecto a la variable x, utilizando la biblioteca SymPy.

7)Integrar

```
def integrar(expresion):
    try:
        expresion = limpiar_expresion(expresion)
```

primero limpia la expresion, luego obtiene la expresion en valores de sympy y evalua la integral de la expresion.En caso de error manda un mensaje

8)Graficas

```
def graficar_funcion(entry):

    from conexionfb import current_user_id

    expr, variables = obtener_expresion(entry.get())  # Obtener la expresión y las variables

    if expr:
```

Toma una expresión ingresada por el usuario y la grafica en 2D o 3D según la cantidad de variables

presentes. Si la expresión tiene una sola variable, se grafica en 2D, convirtiendo la expresión de SymPy a una función de NumPy y evaluándose en un rango de valores de x. Si la expresión tiene dos variables, se genera una gráfica 3D con plot_surface, evaluando la función en una malla de valores de x y y. Además, si el usuario ha iniciado sesión, se registra la operación en Firebase. En caso de error, se muestra un mensaje de error.

9) Actualizar pantalla

```
# Funciones para actualizar la pantalla
def actualizar_pantalla(valor,entrada):
    entrada.delete(0, tk.END)
    entrada.insert(0, valor)
```

Actualiza la pantalla cuando se da una res

10)Clear all

```
# Funciones para los botones AC, DEL y
Ans
def clear_all(entrada,etiqueta):
    entrada.delete(0, tk.END)
    etiqueta.configure(text=" ")
```

Borra tanto lo que haya en la etiqueta como en la entrada

11)Delete last

```
def delete_last(entrada):
    current_text = entrada.get()
    entrada.delete(0, tk.END)
    entrada.insert(0,
current_text[:-1])
```

Borra el ultimo valor de la entrada

12) Insert answer

```
def insert_answer(entry,etiqueta):

agregar_a_pantalla(entry,etiqueta.cget(
"text"))
```

Ingresa el valor de la etiqueta

13)Toggle sing

```
def toggle_sign(entrada):
    current_text = entrada.get()   #
Obtener el texto actual del Entry
    if current_text:  # Verifica que no
esté vacío
```

Ingresa un signo - en el final si no hay y si lo hay lo quita

14)Cerrar app

```
def cerrar_aplicacion():
    sys.exit()
```

Cierra la app

Otras Calculadoras:

1)Reistencias:

```
def calcular_resistencia(banda1,
banda2, banda3, banda4,etiqueta):
    try:
        # Obtener los valores de las
bandas
        banda1_valor =
valores[banda1.get()]
        banda2_valor =
valores[banda2.get()]
        multiplicador = 10 **
valores[banda3.get()]
        tolerancia =
tolerancias[banda4.get()]
```

Este código define una función que calcula el valor de una resistencia de 4 bandas de colores, utilizando diccionarios que almacenan los valores de cada banda.

Matrices:

```
def crear_matriz(filas, columnas,
frame, editable=True):
    matriz = []
    for i in range(filas):
        fila = []
        for j in range(columnas):
            entrada =
ctk.CTkEntry(frame, width=50,
font=("Arial", 12))
```

```
            entrada.grid(row=i,
column=j, padx=5, pady=5)
```

Este código define una función que crea una matriz de entradas en una interfaz gráfica utilizando CustomTkinter

```
def obtener_matriz(matriz_entrada):
    try:
        return
sp.Matrix([[sp.sympify(entry.get()) for
entry in fila] for fila in
matriz_entrada])
    except:
        CTkMessagebox(title="Error",
message="Valores inválidos en la
matriz", icon="cancel")
        return None
```

La función obtener_matriz(matriz_entrada) convierte una matriz de entradas de CustomTkinter en una matriz simbólica de SymPy, lo que permite realizar operaciones algebraicas con ella.

```
def mostrar_resultado(matriz_resultado,
frame_resultado):
    for widget in
frame_resultado.winfo_children():
        widget.destroy()
```

Esta función muestra una matriz en un frame de la interfaz gráfica.
Primero limpia el frame, eliminando cualquier contenido previo.

```
def calcular_operacion(operacion,
interfaz):
    matriz =
obtener_matriz(interfaz.matriz_entrada)
    if matriz is None:
        return
```

Esta función se encarga de realizar una operación matemática sobre una matriz ingresada por el usuario en la interfaz gráfica.

X. Bibliografía

- https://digital55.com/blog/que-es-firebase-funcionalidades-ventajas-conclusiones/
- https://customtkinter.tomschimansky.com/
- https://docs.python.org/es/3.13/library/tkinter.html
- https://numpy.org/doc/2.2/
- https://docs.python.org/es/3.13/library/datetime.html
- https://www.sympy.org/en/index.html
- https://matplotlib.org/
- https://docs.python.org/3/library/re.html