Capítulo 1

Analizador Avanzado de Funciones Matemáticas

Datos Generales

Curso: Programación Numérica
Docente: Ing. Fred Torres Cruz
Alumno: Wily Calib Caira Huancullo
Universidad Nacional del Altiplano – Puno

Descripción del problema

El presente trabajo consiste en el desarrollo de un programa en Python con interfaz gráfica utilizando **Tkinter**. El objetivo es analizar una expresión matemática ingresada por el usuario, identificando sus variables, operaciones y determinando si la relación es una función o no. Este analizador utiliza **SymPy** para el procesamiento simbólico y **expresiones regulares (re)** para estructurar correctamente los términos.

Objetivos específicos

- Identificar las variables dentro de una expresión matemática.
- Contar el número de operaciones aritméticas.
- Verificar si la expresión representa una función.
- Mostrar los resultados mediante una interfaz amigable.

Código en Python

Listing 1.1: Analizador Avanzado de Funciones Matemáticas con Tkinter y SymPy.

```
1 import tkinter as tk
2 from tkinter import ttk, messagebox
3 import sympy as sp
4 import re
```

```
5
6  def analizar():
7      expresion = entrada.get().strip()
8
9      if not expresion:
10         messagebox.showwarning("Error", "Por favor ingresa una
               expresi n matem tica.")
11         return
12
13     # Reemplazar ^ por ** para potencias
14     expresion = expresion.replace("^", "**")
15
16     # Agregar * donde falta (5x -> 5*x, xy -> x*y)
17     expresion = re.sub(r'(\d)([a-zA-Z])', r'\1*\2', expresion)
18     expresion = re.sub(r'([a-zA-Z])(\d)', r'\1*\2', expresion)
19     expresion = re.sub(r'([a-zA-Z])([a-zA-Z])', r'\1*\2', expresion
           )
20
21     try:
22         if "=" in expresion:
23             izquierda, derecha = expresion.split("=")
24             lhs = sp.sympify(izquierda.strip())
25             rhs = sp.sympify(derecha.strip())
26             eq = sp.Eq(lhs, rhs)
27         else:
28             lhs, rhs = sp.symbols("y"), sp.sympify(expresion)
29             eq = sp.Eq(lhs, rhs)
30
31         variables = list(eq.free_symbols)
32         num_vars = len(variables)
33
34         y = sp.Symbol("y")
35         if y in variables:
36             soluciones = sp.solve(eq, y)
37         else:
38             var_dep = variables[0]
39             soluciones = sp.solve(eq, var_dep)
40
41         operaciones = sum(expresion.count(op) for op in "+-*/**")
42
43         resultado_texto = f"Expresi n: {expresion}\n"
44         resultado_texto += f"Variables encontradas: {', '.join(str(
               v) for v in variables)}\n"
45         resultado_texto += f"N mero de variables: {num_vars}\n"
46         resultado_texto += f"N mero de operaciones: {operaciones}\
               n"
47
48         if len(soluciones) == 1:
49             resultado_texto += f"Es funci n: {soluciones[0]}"
50         elif len(soluciones) > 1:
```

```
            resultado_texto += f"No es funci n (m ltiples salidas
                : {soluciones})"
        else:
            resultado_texto += "No se pudo determinar si es
                funci n."

        resultado.set(resultado_texto)

    except Exception as e:
        messagebox.showerror("Error", f"No se pudo analizar la
            expresi n.\n{e}")

# ----------------------------
ventana = tk.Tk()
ventana.title("Analizador Avanzado de Funciones")
ventana.geometry("500x400")
ventana.configure(bg="#f4f6f7")

estilo = ttk.Style()
estilo.configure("TButton", font=("Arial", 12), padding=6)
estilo.configure("TLabel", font=("Arial", 12))

titulo = tk.Label(
    ventana, text="Analizador Avanzado de Funciones",
    font=("Arial", 16, "bold"), bg="#f4f6f7", fg="#2c3e50"
)
titulo.pack(pady=10)

entrada = ttk.Entry(ventana, font=("Consolas", 14))
entrada.pack(pady=10, ipadx=20, ipady=5)

btn = ttk.Button(ventana, text="Analizar", command=analizar)
btn.pack(pady=10)

resultado = tk.StringVar()
label_resultado = tk.Label(
    ventana, textvariable=resultado,
    font=("Consolas", 12), bg="#ecf0f1", fg="#2c3e50",
    relief="groove", justify="left", anchor="w"
)
label_resultado.pack(fill="both", expand=True, padx=20, pady=10)

ventana.mainloop()
```

Ejemplo de ejecución

Entrada:

```
x^2 + 3x - 4 = 0
```

Salida esperada:

```
Expresión: x**2 + 3*x - 4 = 0
Variables encontradas: x
Número de variables: 1
Número de operaciones: 3
Es función: [-3.0, 1.0]
```

Interpretación

El programa reconoce automáticamente las variables, identifica las operaciones y evalúa si la expresión ingresada representa una función. En este ejemplo, la ecuación cuadrática $x^2 + 3x - 4 = 0$ tiene dos soluciones reales, por lo tanto no es una función unívoca. El uso combinado de **SymPy** y **Tkinter** permite integrar análisis simbólico con una interfaz sencilla y visualmente ordenada.

Conclusión

El **Analizador Avanzado de Funciones Matemáticas**, desarrollado por **Wily Calib Caira Huancullo** para el curso de **Programación Numérica**, demuestra la integración entre el análisis simbólico y la programación visual. Esta herramienta facilita la comprensión de las propiedades de una función y promueve el uso de Python en el análisis matemático interactivo.