Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática
Docente: Fred Torres Cruz
Autor : Ronald Junior Pilco Nuñez

Trabajo Encargado - N° 001

Graficador de Funciones

1. Introducción

Se desarrolló una aplicación para graficar funciones matemáticas de una variable usando Python, con Tkinter para la interfaz gráfica y Matplotlib para la visualización.

2. Tecnologías

- **Python**: Lenguaje multiparadigma, ideal por su simplicidad y amplio ecosistema.
- **Tkinter**: Biblioteca de python para construir la interfaz de usuario.
- **Matplotlib**: Biblioteca de python para generación de gráficas personalizables.
- **NumPy**: Biblioteca de python para anejo eficiente de datos numéricos.

3. Arquitectura

- **Interfaz**: Permite ingresar la función y el intervalo.
- **Cálculo**: Evalúa la función en el rango indicado.
- **Visualización**: Genera la gráfica con cuadrícula, etiquetas y límites definidos.

4. Características

- Entrada de funciones matemáticas (ej. x^2).
- Configuración del rango de la variable.
- Gráfica con cuadrícula, etiquetas y leyenda.

5. Código

TRABAJO NRO 001.py - C:\Users\nalfj\OneDrive\Freed\TRABAJO NRO 001.py (3.13.1)

File Edit Format Run Options Window Help

```python
import tkinter as tk
from tkinter import ttk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import matplotlib.pyplot as plt
import numpy as np

def graficar_funcion():
    funcion = entrada_funcion.get()
    try:
        x_min = float(entrada_xmin.get())
        x_max = float(entrada_xmax.get())
        x = np.linspace(x_min, x_max, 400)
        y = eval(funcion)
        ax.clear()
        ax.plot(x, y, label=f"f(x) = {funcion}")
        ax.set_xlim([x_min, x_max])
        ax.grid(True, which='both', linestyle='--', linewidth=0.5)
        ax.set_title('Gráfica de la Función')
        ax.set_xlabel('X')
        ax.set_ylabel('Y')
        ax.legend()

        canvas.draw()
    except Exception as e:
        print(f"Error al graficar: {e}")

ventana = tk.Tk()
ventana.title("Graficadora de Funciones")
ventana.geometry("800x600")
ventana.config(bg="#87CEEB")

ttk.Label(ventana, text="Función f(x):", background="#87CEEB").pack(pady=5)
entrada_funcion = ttk.Entry(ventana, width=50)
entrada_funcion.pack(pady=5)

ttk.Label(ventana, text="Límite X mínimo:", background="#87CEEB").pack(pady=5)
entrada_xmin = ttk.Entry(ventana, width=10)
entrada_xmin.pack(pady=5)

ttk.Label(ventana, text="Límite X máximo:", background="#87CEEB").pack(pady=5)
entrada_xmax = ttk.Entry(ventana, width=10)
entrada_xmax.pack(pady=5)

boton_graficar = tk.Button(ventana, text="Graficar", command=graficar_funcion, bg="#4682B4", fg="white")
boton_graficar.pack(pady=10)

fig, ax = plt.subplots()
canvas = FigureCanvasTkAgg(fig, master=ventana)
canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)

ventana.mainloop()
```

Ln: 52 Col: 0

Figura 1: Código en Python

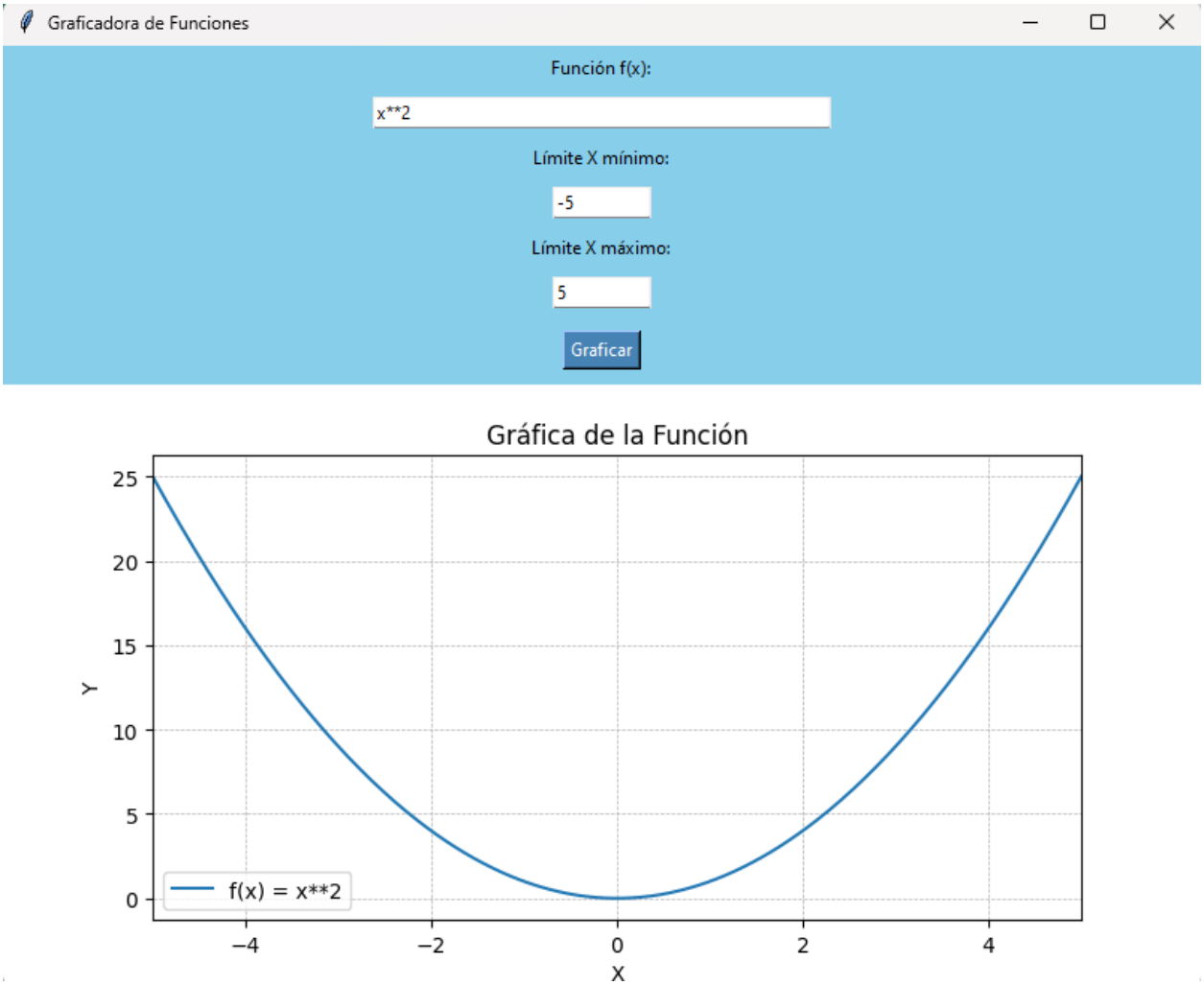

Figura 2: Ejemplo de gráfica generada por la aplicación.

6. Conclusión

Python y sus bibliotecas permiten crear herramientas intuitivas y funcionales para la visualización matemática. Futuras mejoras podrían incluir validación más robusta y exportación de gráficas.

Github

Repositorio Git