UNIVERSIDAD TECNICA DEL NORTE

Nombre: Steven Vinueza

Fecha:9/1/2024

Materia: Propagacion de Ondas

Código de Phyton Proyecto de Propagación de Ondas

```
import math
import tkinter as tk
from tkinter import ttk


class CalculadoraPropagacion(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Calculadora de Propagación")
        self.geometry("400x300")


        self.Parametros()


    def Parametros(self):
        # Crear un cuadro combinado para elegir el tipo de cálculo
        self.calculo_var = tk.StringVar()
        self.calculo_combobox = ttk.Combobox(self, textvariable=self.calculo_var, values=["Espacio Libre", "Okumura Hata"])
        self.calculo_combobox.set("Espacio Libre")
        self.calculo_combobox.grid(row=0, column=0, padx=10, pady=10)


        # Crear etiquetas y campos de entrada
        self.distancia_label = ttk.Label(self, text="Distancia:")
        self.distancia_entry = ttk.Entry(self)
```

```
        self.distancia_label.grid(row=1, column=0, padx=10, pady=5)
        self.distancia_entry.grid(row=1, column=1, padx=10, pady=5)

        self.frecuencia_label = ttk.Label(self, text="Frecuencia:")
        self.frecuencia_entry = ttk.Entry(self)
        self.frecuencia_label.grid(row=2, column=0, padx=10, pady=5)
        self.frecuencia_entry.grid(row=2, column=1, padx=10, pady=5)

        self.altura_transmisor_label = ttk.Label(self, text="Altura Transmisor:")
        self.altura_transmisor_entry = ttk.Entry(self)
        self.altura_transmisor_label.grid(row=3, column=0, padx=10, pady=5)
        self.altura_transmisor_entry.grid(row=3, column=1, padx=10, pady=5)

        self.altura_receptor_label = ttk.Label(self, text="Altura Receptor:")
        self.altura_receptor_entry = ttk.Entry(self)
        self.altura_receptor_label.grid(row=4, column=0, padx=10, pady=5)
        self.altura_receptor_entry.grid(row=4, column=1, padx=10, pady=5)

        self.tipo_ciudad_label = ttk.Label(self, text="Tipo Ciudad:")
        self.tipo_ciudad_combobox = ttk.Combobox(self, values=["URBANO", "SUBURBANO",
"RURAL"])
        self.tipo_ciudad_combobox.set("URBANO")
        self.tipo_ciudad_label.grid(row=5, column=0, padx=10, pady=5)
        self.tipo_ciudad_combobox.grid(row=5, column=1, padx=10, pady=5)

        # Crear botón de calcular
        self.calcular_button = ttk.Button(self, text="Calcular", command=self.calcular)
        self.calcular_button.grid(row=6, column=0, columnspan=2, pady=10)
```

```
        # Crear etiqueta para mostrar el resultado
        self.resultado_label = ttk.Label(self, text="")
        self.resultado_label.grid(row=7, column=0, columnspan=2, pady=10)

    def calcular(self):
        distancia = float(self.distancia_entry.get())
        frecuencia = float(self.frecuencia_entry.get())
        altura_transmisor = float(self.altura_transmisor_entry.get())
        altura_receptor = float(self.altura_receptor_entry.get())
        tipo_ciudad = self.tipo_ciudad_combobox.get()

        if self.calculo_var.get() == "Espacio Libre":
            resultado = self.ModeloEspacioLibre(distancia, frecuencia)
        elif self.calculo_var.get() == "Okumura Hata":
            resultado = self.ModeloOkumuraHata(distancia, frecuencia, altura_transmisor,
altura_receptor, tipo_ciudad)
        else:
            resultado = "Error"

        self.resultado_label.config(text=f"Resultado: {resultado} dB")

    def ModeloEspacioLibre(self, distancia, frecuencia):
        velocidad_luz = 3 * 10**8  # velocidad de la luz en metros/segundo
        longitud_onda = velocidad_luz / (frecuencia * 10**6)  # longitud de onda en metros
        perdida_espacio_libre = 20 * math.log10(distancia) + 20 * math.log10(frecuencia) + 20 *
math.log10(4 * math.pi / longitud_onda)
        return perdida_espacio_libre

    def ModeloOkumuraHata(self, distancia, frecuencia, altura_transmisor, altura_receptor,
tipo_ciudad):
```

```
    if tipo_ciudad == "URBANO":
        q, r = 3.2, 11.75
    elif tipo_ciudad == "SUBURBANO":
        q, r = 3.2, 9.25
    elif tipo_ciudad == "RURAL":
        q, r = 3.2, 8.5
    else:
        raise ValueError("Tipo de ciudad no válido")

    C = 0  # para frecuencias menores a 300 MHz
    if frecuencia > 300:
        C = 3.2 * (math.log10(11.75 * altura_receptor))**2 - 4.97

    altura_efectiva = min(30, altura_transmisor)  # altura efectiva de la antena transmisora

    perdida_urbano = 69.55 + 26.16 * math.log10(frecuencia) - 13.82 * math.log10(altura_transmisor) - q + (44.9 - 6.55 * math.log10(altura_transmisor)) * math.log10(distancia) + C
    perdida_suburbano = 69.55 + 26.16 * math.log10(frecuencia) - 13.82 * math.log10(altura_transmisor) - r + (44.9 - 6.55 * math.log10(altura_transmisor)) * math.log10(distancia) + C

    if tipo_ciudad == "URBANO":
        return perdida_urbano
    elif tipo_ciudad == "SUBURBANO":
        return perdida_suburbano
    elif tipo_ciudad == "RURAL":
        return perdida_suburbano  # En el modelo original, se usa la misma fórmula para suburbano y rural

if __name__ == "__main__":
```

```
app = CalculadoraPropagacion()
app.mainloop()
```