UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
COLEGIO DE CIENCIAS Y HUMANIDADES
Centro de Formación Continua

FORMATO DE PLANEACIÓN
Estrategia didáctica

DATOS GENERALES

Nombre del participante	**Martín Ramírez Fuentes**
Asignatura	Sistemas computacionales desarrollo de software
Año o semestre en que imparte	2024-1 (grupos de tercero y quinto semestre, ambos turnos)
Horas clase a la semana	6 horas
Unidad	MÓDULO VII. LENGUAJE VISUAL “INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A EVENTOS”
Aprendizajes	Soluciona problemas a través de elementos propios del lenguaje de programación diseñando una interfaz legible que muestre resultados de manera sencilla.
Problemática que se abordará a través del problema.	Realizar conversiones de unidades de magnitudes físicas longitudinales y sus equivalencias (millas a km, km a millas, yardas a metros, metros a yardas, pies a metros, metros a pies) aplicando una interfaz gráfica.
Justificación. **(porque considera que el programa en python o Julia puede apoyar al alumno a entender o lograr el**	*Porque el programa de python, cuenta con muchas bibliotecas de apoyo para programar, en especial con la interfaz gráfica, entre ellas utilizaremos la biblioteca de python Tkinter, esta biblioteca facilita la programación de ventanas,* botones, cajas de texto, etiquetas, colores de fondo o imagen de fondo, de tal forma que los estudiantes de manera colaborativa, desarrollen, cada uno, parte de las conversiones

aprendizaje)	de física y las integren en un programa de conversiones de unidades longitudinales de Física.
Producto esperado **(Después de haber explicado, haber realizado alguna actividad guiada y/o dejar una actividad extraclase, ¿Qué evidencia tiene que entregar para ser evaluada?**	Un menú de opciones que ejecuten las conversiones de magnitudes físicas longitudinales (millas a km, km a millas, yardas a metros, metros a yardas, pies a metros, metros a pies) aplicando una interfaz gráfica.
Recursos materiales /Herramientas TIC	• Computadora o laptop, • Software: Lenguaje de programación python • Conexión a internet, • Plataforma educativa: Teams • Videoproyector, • Pizarrón blanco, • Plumigis • Lista de cotejo para evaluar el desarrollo del proyecto.
Tiempos de realización.	10 horas 6 en clase y 4 extraclase
Forma de trabajo	En equipos de forma colaborativa

Secuencia didáctica

	**Presentación del problema a resolver** **Se solicita:** Un menú de opciones que ejecuten las conversiones de magnitudes físicas longitudinales (millas a km, km a millas, yardas a metros, metros a yardas,

Secuencia didáctica

pies a metros, metros a pies), aplicando una interfaz gráfica en python importando la biblioteca de Tkinter.

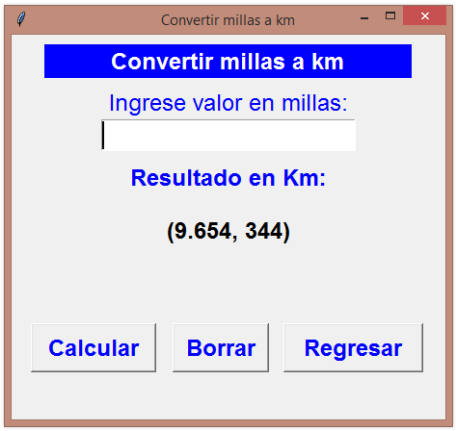

Imagen 1. Menú de opciones aplicando una interfaz gráfica

Inicio de la Sesión

Utilizando la **imagen 1**, se solicita a los alumnos que observen e identifiquen cuales son los elementos gráficos que se encuentran en la imagen.

Los alumnos observan e identifican cuales son los elementos gráficos que se encuentran en la imagen.

Una vez que los alumnos han observado e identificado cuales son los elementos gráficos que se encuentran en la imagen, el profesor explica el comportamiento de los elementos de la interfaz gráfica a utilizar:

Button
Label
Entry
Pack()
Grid()
Place()

Secuencia didáctica

	Importar la biblioteca o módulo que permita utilizar los elementos de la interfaz gráfica. **import tkinter as tk** Crear la ventana principal ventana = tk.Tk() Medidas de la ventana ventana.geometry("500x500")
	**Desarrollo de la sesión** El profesor modela la entrada de datos de **millas a km**, utilizando el elemento gráfico: **Entry** El elemento gráfico para etiquetas: **Label** Elemento gráfico para botones: **Button** Así como la ubicación de cada uno de ellos en la ventana utilizando los métodos: **Pack()**, **Grid()** y **Place()** También el botón calcular, borrar y regresar, así como las etiquetas de la conversión de magnitud física longitudinal convertir: **millas a km** Se solicita se integren en equipos de 5 alumnos. El profesor solicita a cada integrante del equipo, en forma colaborativa, realizar una de las conversiones de magnitud física longitudinal restantes (**km a millas, yardas a metros, metros a yardas, pies a metros, metros a pies**). Los alumnos se integran en equipos de 5. Cada integrante del equipo realiza una de las conversiones de magnitud física longitudinal (**km a millas, yardas a metros, metros a yardas, pies a metros, metros a pies**).

Secuencia didáctica

Cierre de la sesión

Reflexión

- Se conduce una discusión sobre lo aprendido.
- Se invita a los estudiantes a compartir sus experiencias, desafíos y soluciones al crear sus ventanas.
- Se anima a los estudiantes a considerar posibles mejoras o características adicionales que podrían agregar a la ventana en el futuro.

Presentación

- Se invita a los estudiantes a presentar sus interfaces gráficas y compartir sus logros con sus compañeros de clase.
- Se resumen los conceptos clave aprendidos durante la clase y se refuerza la importancia de las conversiones de unidades en la resolución de problemas reales.

Evaluación

Lista de Cotejo:	Cumple	
	SI	NO
Diseño de interfaz gráfica completo y funcional.		
Botones para todas las conversiones requeridas.		
Funciones de conversión implementadas y conectadas a los botones.		
Resultados de conversiones se muestran de manera adecuada en la interfaz.		
Discusión sobre la experiencia de creación y posibles mejoras.		

Evaluación

Secuencia didáctica

Referencias

TkDocs tutorial. (s/f). Tkdocs.com. Recuperado el 7 de agosto de 2023, de https://tkdocs.com/tutorial/

NeuralNine [@NeuralNine]. (2021, septiembre 29). *Tkinter Beginner Course - Python GUI Development*. Youtube. https://www.youtube.com/watch?v=ibf5cx221hk

Follow, S. (2019, mayo 20). *RadioButton in tkinter*. GeeksforGeeks. https://www.geeksforgeeks.org/radiobutton-in-tkinter-python/?ref=lbp

Alvarez, A. (2016). Guía de tkinter. https://buildmedia.readthedocs.org/media/pdf/guia-tkinter/latest/guia-tkinter.pdf

Stac Overflow contribuitors. (s/f). Aprendizaje tkinter Ebook gratis. https://riptutorial.com/Download/tkinter-es.pdf

Anexo. Código resultado del ejercicio:

```
# Presentación del problema a resolver con la biblioteca Tkinter de Python

# Se solicita: Un menú de opciones que ejecuten las conversiones de magnitudes físicas longitudinales
# (millas a km, km a millas, yardas a metros, metros a yardas, pies a metros, metros a pies), aplicando¶
# una interfaz gráfica en python importando la biblioteca de Tkinter.

# Importar biblioteca gráfica Tkinter
import tkinter as tk

# Diseño de la Interfaz Gráfica

# Clase de la ventana principal
class MainWindow(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Medidas de longitud") # título de la ventana
        self.geometry("450x400") # Medida de la ventana

# Etiquetas de la ventana principal y algunos de sus argumentos
```

```
    etiq_prin = tk.Label(self, text="FISICA I", # Texto
                bg="blue", # fondo de color
                fg="white", # color de la letra
                font=("Arial", 18, "bold"),  # Tipo y tamaño de letra
                height=1,   # altura de la letra
                width=10     # longitud del número de caracteres
                )
    etiq_prin.pack(padx = 10, pady =10) # Método de ubicación en pantalla

    etiq_princ = tk.Label(self, text="Equivalencia de Unidades",
                bg="blue",
                fg="white",
                font=("Arial", 18, "bold"),
                height=1,
                width=24
                )
    etiq_princ.pack(padx = 10) # Método de ubicación en pantalla

    etiq_prin = tk.Label(self, text="Longitudinales",
                bg="blue",
                fg="white",
                font=("Arial", 18, "bold"),
                height=1,
                width=14
                )
    etiq_prin.pack(padx = 10, pady =10) # Método de ubicación en pantalla

# Botones de la ventana principal y algunos de sus argumentos
    self.boton_km = tk.Button(self, text="millas a km",
                command=self.abrir_km,
                fg="#0000ff",
                relief="raised",
                font=("Arial", 18, "bold"),
                height=1,
                width=12
                )
    self.boton_km.place(x = 20, y = 160) # Método de ubicación en pantalla

    self.boton_pie = tk.Button(self, text="pies a m",
                command=self.abrir_pie,
                fg="#0000ff",
```

```
            relief="raised",
            font=("Arial", 18, "bold"),
            height=1,
            width=12
            )
self.boton_pie.place(x = 20, y = 280) # Método de ubicación en pantalla

self.boton_metro = tk.Button(self, text="m a pies",
            command=self.abrir_m,
            fg="#0000ff",
            relief="raised",
            font=("Arial", 18, "bold"),
            height=1,
            width=12
            )
self.boton_metro.place(x = 240, y = 280) # Método de ubicación en pantalla

self.boton_millas = tk.Button(self, text="km a millas",
            command=self.abrir_millas,
            fg="#0000ff",
            relief="raised",
            font=("Arial", 18, "bold"),
            height=1,
            width=12
            )
self.boton_millas.place(x = 20, y = 220) # Método de ubicación en pantalla

self.boton_yarda = tk.Button(self, text="m a yardas",
            command=self.abrir_yarda,
            fg="#0000ff",
            relief="raised",
            font=("Arial", 18, "bold"),
            height=1,
            width=12
            )
self.boton_yarda.place(x = 240, y = 160) # Método de ubicación en pantalla

self.boton_m = tk.Button(self, text="yardas a m",
            command=self.abrir_metro,
            fg="#0000ff",
            relief="raised",
```

```
                font=("Arial", 18, "bold"),
                height=1,
                width=12
                )
    self.boton_m.place(x = 240, y = 220) # Método de ubicación en pantalla

# función de la subventana convertir millas a km
def abrir_km(self):
    self.withdraw()
    km_window = tk.Toplevel(self)
    km_window.title("Convertir millas a km") # Título de la ventana
    km_window.geometry("450x400") # Medida de la ventana

# Etiquetas de la subventana y algunos de sus argumentos
    etiq_km = tk.Label(km_window, text="Convertir millas a km",
                bg="blue",
                fg="white",
                font=("Arial", 18, "bold"),
                height=1,
                width=25
                )
    etiq_km.pack(padx = 10, pady =10) # Método de ubicación en pantalla

    eti_n1 = tk.Label(km_window, text="Ingrese valor en millas:",
                fg="blue",
                font=("Arial", 18),
                height=1,
                width=20
                )
    eti_n1.pack() # Método de ubicación en pantalla

# Entrada de datos
    ingresa_n1 = tk.Entry(km_window,
                fg="red",
                font=("Arial", 18, "bold")
                )
    ingresa_n1.pack() # Método de ubicación en pantalla

    etiq_n1 = tk.Label(km_window, text="Resultado en Km:",
                fg="#0000ff",
                font=("Arial", 18, "bold"),
```

```
                height=1,
                width=16
                )
    etiq_n1.pack(pady =10) # Método de ubicación en pantalla

    result_var = tk.StringVar()
    imprime_resultado = tk.Label(km_window,
                textvariable=result_var,
                fg="#000000",
                font=("Arial", 18, "bold"),
                height=1,
                width= 24
                )
    imprime_resultado.pack(pady =10) # Método de ubicación en pantalla

  # Funciones para: calcular, borrar y regresar
    def calcula_km():
        try:
            n1 = float(ingresa_n1.get())
            result = n1 * 1.609,344
            result_var.set(f"{result}")
        except ValueError:
            result_var.set("Error: Ingrese números válidos")

    def borra_datos():
        ingresa_n1.delete(0, tk.END)
        result_var.set("")

    def regresa():
        self.deiconify()
        km_window.withdraw()

  # Botones para: calcular, borrar y regresar
    boton_calcula = tk.Button(km_window, text="Calcular",
                    command=calcula_km,
                    fg="#0000ff",
                    relief="raised",
                    font=("Arial", 18, "bold"),
                    height=1,
                    width=8
                    )
```

```
    boton_calcula.place(x = 20, y = 300) # Método de ubicación en pantalla

    boton_borra = tk.Button(km_window, text="Borrar",
                    command=borra_datos,
                    fg="#0000ff",
                    relief="raised",
                    font=("Arial", 18, "bold"),
                    height=1,
                    width=6
                    )
    boton_borra.place(x = 167, y = 300) # Método de ubicación en pantalla

    boton_regresa = tk.Button(km_window, text="Regresar",
                    command=regresa,
                    fg="#0000ff",
                    relief="raised",
                    font=("Arial", 18, "bold"),
                    height=1,
                    width=9
                    )
    boton_regresa.place(x = 282, y = 300) # Método de ubicación en pantalla

# función de la subventana convertir km a millas
def abrir_millas(self):
    self.withdraw()
    sub_window = tk.Toplevel(self)
    sub_window.title("Convertir Km a Millas") # Título de la ventana
    sub_window.geometry("450x400") # Medidas de la ventana

# Etiquetas de la subventana y algunos de sus argumentos.
    etiq_km = tk.Label(sub_window, text="Convertir km a millas",
                    bg="blue",
                    fg="white",
                    font=("Arial", 18, "bold"),
                    height=1,
                    width=25
                    )
    etiq_km.pack(padx = 10, pady =10) # Método de ubicación en pantalla

    n1_label = tk.Label(sub_window, text="Ingrese valor en Km:",
                    fg="blue",
```

```
                font=("Arial", 18),
                height=1,
                width=20
                )
    n1_label.pack() # Método de ubicación en pantalla

# Entrada de datos
    n1_entry = tk.Entry(sub_window,
                fg="red",
                font=("Arial", 18, "bold")
                )
    n1_entry.pack() # Método de ubicacion en pantalla

    result_label = tk.Label(sub_window, text="Resultado en millas",
                fg="#0000ff",
                font=("Arial", 18, "bold"),
                height=1,
                width=16
                )
    result_label.pack(pady =10) # Método de ubicación en pantalla
    result_var = tk.StringVar()

    result_display = tk.Label(sub_window,
                textvariable=result_var,
                fg="#000000",
                font=("Arial", 18, "bold"),
                height=1,
                width= 24
                )
    result_display.pack(pady =10) # Método de ubicación en pantalla

# Funciones y botones para: calcular, borrar y regresar
    def calcula_millas():
        try:
            n1 = float(n1_entry.get())
            result = n1 / 1.609,344
            result_var.set(f"{result}")
        except ValueError:
            result_var.set("Error: Ingrese números válidos")

    def clear_data():
```

```
    n1_entry.delete(0, tk.END)
    result_var.set("")

  def regresa():
    self.deiconify()
    sub_window.withdraw()

  calc_button = tk.Button(sub_window, text="Calcular",
                command=calcula_millas,
                fg="#0000ff",
                relief="raised",
                font=("Arial", 18, "bold"),
                height=1,
                width=8
                )
  calc_button.place(x = 20, y = 300) # Método de ubicación en pantalla

  clear_button = tk.Button(sub_window, text="Borrar",
                command=clear_data,
                fg="#0000ff",
                relief="raised",
                font=("Arial", 18, "bold"),
                height=1,
                width=6
                )
  clear_button.place(x = 167, y = 300) # Método de ubicación en pantalla

  back_button = tk.Button(sub_window, text="Regresar",
                command=regresa,
                fg="#0000ff",
                relief="raised",
                font=("Arial", 18, "bold"),
                height=1,
                width=9
                )
  back_button.place(x = 282, y = 300) # Método de ubicación en pantalla

# función de la subventana convertir metros a yardas
def abrir_yarda(self):
  self.withdraw()
  yarda_window = tk.Toplevel(self)
```

```
    yarda_window.title("Convertir metros a yardas")
    yarda_window.geometry("450x400")

# Etiquetas de la subventana y algunos de sus argumentos.
    etiq_m = tk.Label(yarda_window, text="Convertir metros a yardas",
                bg="blue",
                fg="white",
                font=("Arial", 18, "bold"),
                height=1,
                width=25
                )
    etiq_m.pack(padx = 10, pady =10)

    n1_label = tk.Label(yarda_window, text="Ingrese valor en metros:",
                fg="blue",
                font=("Arial", 18),
                height=1,
                width=20
                )
    n1_label.pack()

 # Entrada de datos
    n1_entry = tk.Entry(yarda_window,
                fg="red",
                font=("Arial", 18, "bold")
                )
    n1_entry.pack()

    result_label = tk.Label(yarda_window, text="Resultado en yardas:",
                fg="#0000ff",
                font=("Arial", 18, "bold"),
                height=1,
                width=16
                )
    result_label.pack(pady =10)

    result_var = tk.StringVar()
    result_display = tk.Label(yarda_window,
                textvariable=result_var,
                fg="#000000",
                font=("Arial", 18, "bold"),
```

```
                height=1,
                width= 24
                )
    result_display.pack(pady =10)

# Funciones y botones para: calcular, borrar y regresar
    def calcula_yarda():
        try:
            n1 = float(n1_entry.get())
            result = n1 * 1.09361
            result_var.set(f"{result}")
        except ValueError:
            result_var.set("Error: Ingrese números válidos")

    def borra_dato():
        n1_entry.delete(0, tk.END)
        result_var.set("")

    def regresa():
        self.deiconify()
        yarda_window.withdraw()

    boton_calcula = tk.Button(yarda_window, text="Calcular",
                    command=calcula_yarda,
                    fg="#0000ff",
                    relief="raised",
                    font=("Arial", 18, "bold"),
                    height=1,
                    width=8
                    )
    boton_calcula.place(x = 20, y = 300)

    boton_borra = tk.Button(yarda_window, text="Borrar",
                    command=borra_dato,
                    fg="#0000ff",
                    relief="raised",
                    font=("Arial", 18, "bold"),
                    height=1,
                    width=6
                    )
    boton_borra.place(x = 167, y = 300)
```

```
    boton_regresa = tk.Button(yarda_window, text="Regresar",
                    command=regresa,
                    fg="#0000ff",
                    relief="raised",
                    font=("Arial", 18, "bold"),
                    height=1,
                    width=9
                    )
    boton_regresa.place(x = 282, y = 300)


# función de la subventana convertir yardas a metros
def abrir_metro(self):
    self.withdraw()
    m_window = tk.Toplevel(self)
    m_window.title("Convertir yardas a metros") # Título de la ventana
    m_window.geometry("450x400") # Medidas de la ventana

# Etiquetas de la subventana y algunos de sus argumentos.
    etiq_y = tk.Label(m_window, text="Convertir yardas a metros",
                    bg="blue",
                    fg="white",
                    font=("Arial", 18, "bold"),
                    height=1,
                    width=25
                    )
    etiq_y.pack(padx = 10, pady =10)

    etiq_n1 = tk.Label(m_window, text="Ingrese valor en yardas:",
                    fg="blue",
                    font=("Arial", 18),
                    height=1,
                    width=20
                    )
    etiq_n1.pack()

# Entrada de datos
    ingresa_n1 = tk.Entry(m_window,
                    fg="red",
                    font=("Arial", 18, "bold")
```

```
                )
    ingresa_n1.pack()

    etiq_resultado = tk.Label(m_window, text="Resultado en metros:",
                fg="#0000ff",
                font=("Arial", 18, "bold"),
                height=1,
                width=16
                )
    etiq_resultado.pack(pady =10)

    result_var = tk.StringVar()
    imprime_resultado = tk.Label(m_window,
                textvariable=result_var,
                fg="#000000",
                font=("Arial", 18, "bold"),
                height=1,
                width= 24
                )
    imprime_resultado.pack(pady =10)

# Funciones y botones para: calcular, borrar y regresar
    def calcula_datos():
        try:
            n1 = float(ingresa_n1.get())
            result = n1 / 1.09361
            result_var.set(f"{result}")
        except ValueError:
            result_var.set("Error: Ingrese números válidos")

    def borra_dato():
        ingresa_n1.delete(0, tk.END)
        result_var.set("")

    def regresa():
        self.deiconify()
        m_window.withdraw()

    boton_calcula = tk.Button(m_window, text="Calcular",
                    command=calcula_datos,
                    fg="#0000ff",
```

```
                relief="raised",
                font=("Arial", 18, "bold"),
                height=1,
                width=8
                )
    boton_calcula.place(x = 20, y = 300)

    boton_borra = tk.Button(m_window, text="Borrar",
                command=borra_dato,
                fg="#0000ff",
                relief="raised",
                font=("Arial", 18, "bold"),
                height=1,
                width=6
                )
    boton_borra.place(x = 167, y = 300)

    boton_regresa = tk.Button(m_window, text="Regresar",
                command=regresa,
                fg="#0000ff",
                relief="raised",
                font=("Arial", 18, "bold"),
                height=1,
                width=9
                )
    boton_regresa.place(x = 282, y = 300)

# función de la subventana convertir pies a metros
def abrir_pie(self):
    self.withdraw()
    pie_window = tk.Toplevel(self)
    pie_window.title("Convertir pies a metros")
    pie_window.geometry("450x400")

# Etiquetas de la subventana y algunos de sus argumentos.
    etiq_pie = tk.Label(pie_window, text="Convertir pies a metros",
                bg="blue",
                fg="white",
                font=("Arial", 18, "bold"),
                height=1,
                width=25
```

```
                )
    etiq_pie.pack(padx = 10, pady =10)

    etiq_n1 = tk.Label(pie_window, text="Ingrese valor en pies:",
                       fg="blue",
                       font=("Arial", 18),
                       height=2,
                       width=20
                       )
    etiq_n1.pack()

# Entrada de datos
    ingresa_n1 = tk.Entry(pie_window,
                          fg="red",
                          font=("Arial", 18, "bold")
                          )
    ingresa_n1.pack()

    etiq_resulta = tk.Label(pie_window, text="Resultado en metros:",
                       fg="#0000ff",
                       font=("Arial", 18, "bold"),
                       height=1,
                       width=16
                       )
    etiq_resulta.pack(pady =10)
    result_var = tk.StringVar()
    imprime_resultado = tk.Label(pie_window,
                    textvariable=result_var,
                    fg="#000000",
                    font=("Arial", 18, "bold"),
                    height=1,
                    width= 24
                    )
    imprime_resultado.pack(pady =10)

# Funciones y botones para: calcular, borrar y regresar
    def calcula_pie():
        try:
            n1 = float(ingresa_n1.get())
            result = n1 * 0.3048
            result_var.set(f"{result}")
```

```
    except ValueError:
        result_var.set("Error: Ingrese números válidos")

def borra_datos():
    ingresa_n1.delete(0, tk.END)
    result_var.set("")

def regresa():
    self.deiconify()
    pie_window.withdraw()

boton_calcula = tk.Button(pie_window, text="Calcular",
                command=calcula_pie,
                fg="#0000ff",
                relief="raised",
                font=("Arial", 18, "bold"),
                height=1,
                width=8
                )
boton_calcula.place(x = 20, y = 300)

boton_borra = tk.Button(pie_window, text="Borrar",
                command=borra_datos,
                fg="#0000ff",
                relief="raised",
                font=("Arial", 18, "bold"),
                height=1,
                width=6
                )
boton_borra.place(x = 167, y = 300)

boton_regresa = tk.Button(pie_window, text="Regresar",
                command=regresa,
                fg="#0000ff",
                relief="raised",
                font=("Arial", 18, "bold"),
                height=1,
                width=9
                )
boton_regresa.place(x = 282, y = 300)
```

```
# función de la subventana convertir metros a pies
def abrir_m(self):
    self.withdraw()
    metro_window = tk.Toplevel(self)
    metro_window.title("Convertir metros a pies")
    metro_window.geometry("450x400")

# Etiquetas de la subventana y algunos de sus argumentos.
    etiq_metro = tk.Label(metro_window, text="Convertir metros a pies",
                bg="blue",
                fg="white",
                font=("Arial", 18, "bold"),
                height=1,
                width=25
                )
    etiq_metro.pack(padx = 10, pady =10)

    etiq_n1 = tk.Label(metro_window, text="Ingrese valor en metros:",
                    fg="blue",
                    font=("Arial", 18),
                    height=2,
                    width=20
                    )
    etiq_n1.pack()

# Entrada de datos
    ingresa_n1 = tk.Entry(metro_window,
                    fg="red",
                    font=("Arial", 18, "bold")
                    )
    ingresa_n1.pack()

    etiq_resulta = tk.Label(metro_window, text="Resultado en pies:",
                    fg="#0000ff",
                    font=("Arial", 18, "bold"),
                    height=1,
                    width=16
                    )
    etiq_resulta.pack(pady =10)
    result_var = tk.StringVar()
    imprime_resultado = tk.Label(metro_window,
```

```
                textvariable=result_var,
                fg="#000000",
                font=("Arial", 18, "bold"),
                height=1,
                width= 24
                )
    imprime_resultado.pack(pady =10)

# Funciones y botones para: calcular, borrar y regresar
    def calcula_metro():
        try:
            n1 = float(ingresa_n1.get())
            result = n1 / 0.3048
            result_var.set(f"{result}")
        except ValueError:
            result_var.set("Error: Ingrese números válidos")

    def borra_datos():
        ingresa_n1.delete(0, tk.END)
        result_var.set("")

    def regresa():
        self.deiconify()
        metro_window.withdraw()

    boton_calcula = tk.Button(metro_window, text="Calcular",
                command=calcula_metro,
                fg="#0000ff",
                relief="raised",
                font=("Arial", 18, "bold"),
                height=1,
                width=8
                )
    boton_calcula.place(x = 20, y = 300)

    boton_borra = tk.Button(metro_window, text="Borrar",
                command=borra_datos,
                fg="#0000ff",
                relief="raised",
                font=("Arial", 18, "bold"),
                height=1,
```

```
                width=6
                )
    boton_borra.place(x = 167, y = 300)

    boton_regresa = tk.Button(metro_window, text="Regresar",
                command=regresa,
                fg="#0000ff",
                relief="raised",
                font=("Arial", 18, "bold"),
                height=1,
                width=9
                )
    boton_regresa.place(x = 282, y = 300)

 # Ejecutar ventana principal
if __name__ == "__main__":
   root = MainWindow()
   root.mainloop()
```