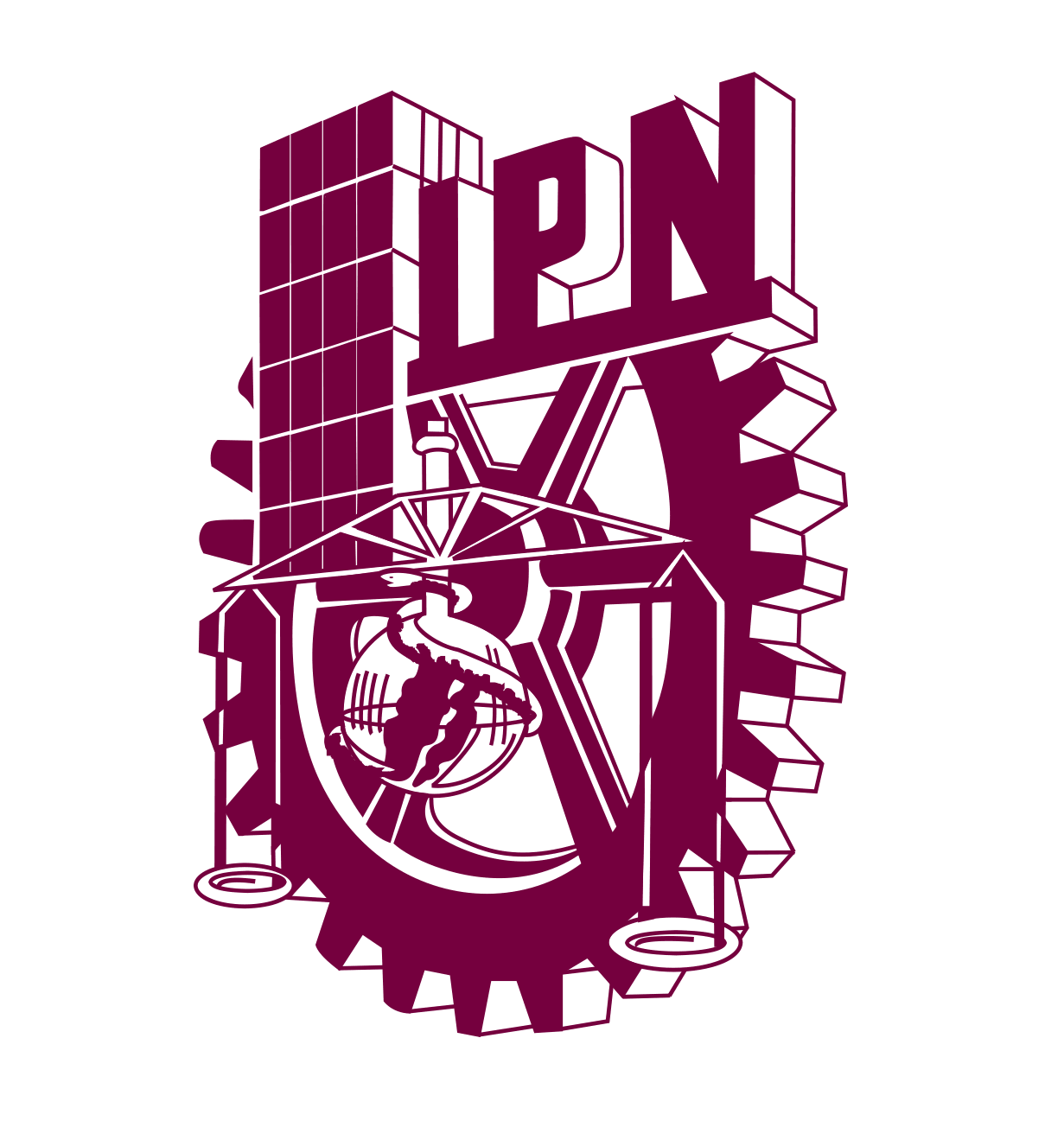
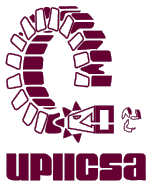
**INSTITUTO POLITÉCNICO NACIONAL**

**Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas**

**Ingeniería de Pruebas**

**Documento Técnico: Documento de Apoyo - Calculadora en Python con Tkinter**

**Autores:**

* Contreras Mosco Cristobal
* Díaz Pérez Diego
* Hernández Aguirre Ricardo
* Monroy Muñoz Angel Yael
* Salazar Rocha Any Jennifer

**Fecha:** [09/09/2025]

**Versión:** 1.0

**Índice**

1. Introducción
2. Objetivo
3. Estructura del sistema

3.1. Dependencias

1. Descripción General del Código
2. Código Fuente Completo
3. Observaciones
4. Conclusión

**Introducción**

El presente documento forma parte de los documentos de apoyo técnico del proyecto Calculadora con Interfaz Gráfica.  
Su propósito es incluir el código fuente completo del sistema, así como observaciones sobre su organización y dependencias, con el fin de garantizar su correcta instalación, ejecución y mantenimiento.

**Objetivo**

Brindar un respaldo técnico que permita:

* Contar con el código fuente íntegro y documentado del sistema.
* Facilitar el mantenimiento y posibles mejoras futuras.
* Permitir la verificación de que el software fue desarrollado con base en los requisitos y diseño definidos previamente.

**Estructura del Sistema**

La aplicación está contenida en un solo archivo ejecutable: calculadora.py

**Dependencias:**

* Python 3.8 o superior
* Tkinter (incluido en la instalación estándar de Python)

**Descripción General del Código**

El código está organizado en clases y métodos que manejan la lógica de la calculadora:

* Clase Calculadora: Contiene toda la lógica y la interfaz.
* Métodos principales:
  + boton\_presionado: Controla los eventos al presionar botones.
  + agregar\_a\_operacion: Añade números u operadores a la operación actual.
  + realizar\_calculo: Evalúa y muestra resultados.
  + borrar\_ultimo\_caracter: Borra el último carácter ingresado.
  + mostrar\_ocultar\_historial: Activa o desactiva el historial de operaciones.

**Código Fuente Completo**

import tkinter as tk

class Calculadora:

def \_\_init\_\_(self, raiz):

self.raiz = raiz

self.raiz.title("Calculadora")

self.raiz.resizable(False, False)

# ================= VARIABLES =================

self.operacion\_actual = ""

self.historial\_visible = False # Oculto al iniciar

self.resultado\_mostrado = False

# ================= FRAME PRINCIPAL =================

marco\_principal = tk.Frame(self.raiz, bg="#d9d9d9")

marco\_principal.pack(side=tk.LEFT, fill=tk.BOTH, padx=5, pady=5)

# ================= PANTALLA =================

self.pantalla = tk.Entry(marco\_principal, font=("Arial", 24), justify="right", bd=10, relief="sunken")

self.pantalla.grid(row=0, column=0, columnspan=4, pady=5)

# ================= HISTORIAL =================

self.marco\_historial = tk.Frame(self.raiz, bg="#f0f0f0", width=200)

self.etiqueta\_historial = tk.Label(self.marco\_historial, text="Historial", font=("Arial", 14, "bold"), bg="#f0f0f0")

self.etiqueta\_historial.pack(pady=5)

self.lista\_historial = tk.Listbox(self.marco\_historial, font=("Arial", 12), height=15, width=25)

self.lista\_historial.pack(padx=5, pady=5, fill=tk.BOTH)

# ================= BOTONES =================

botones = [

("H", 1, 0), ("CA", 1, 1), ("C", 1, 2), ("B", 1, 3),

("7", 2, 0), ("8", 2, 1), ("9", 2, 2), ("/", 2, 3),

("4", 3, 0), ("5", 3, 1), ("6", 3, 2), ("\*", 3, 3),

("1", 4, 0), ("2", 4, 1), ("3", 4, 2), ("-", 4, 3),

("0", 5, 0), (".", 5, 1), ("=", 5, 2), ("+", 5, 3)

]

for (texto, fila, columna) in botones:

tk.Button(marco\_principal, text=texto, width=6, height=2, font=("Arial", 14),

command=lambda t=texto: self.boton\_presionado(t)).grid(row=fila, column=columna, padx=2, pady=2)

# ================= EVENTOS DE BOTONES =================

def boton\_presionado(self, valor):

if valor == "C":

self.operacion\_actual = ""

self.actualizar\_pantalla()

elif valor == "CA":

self.operacion\_actual = ""

self.lista\_historial.delete(0, tk.END)

self.actualizar\_pantalla()

elif valor == "H":

self.mostrar\_ocultar\_historial()

elif valor == "=":

self.realizar\_calculo()

elif valor == "B":

self.borrar\_ultimo\_caracter()

else:

self.agregar\_a\_operacion(valor)

# ================= MANEJO DE OPERACION =================

def agregar\_a\_operacion(self, valor):

if self.resultado\_mostrado:

if valor in "+-\*/":

self.operacion\_actual = self.pantalla.get() + valor

else:

self.operacion\_actual = valor

self.resultado\_mostrado = False

else:

# Limite de caracteres en pantalla

if len(self.operacion\_actual) >= 12:

return

# Evitar dos puntos decimales en el mismo numero

if valor == ".":

partes = self.operacion\_actual.split(" ")[-1] if self.operacion\_actual else self.operacion\_actual

if "." in partes:

return

# Evitar operadores consecutivos, reemplazando el anterior

if valor in "+-\*/":

if self.operacion\_actual and self.operacion\_actual[-1] in "+-\*/":

self.operacion\_actual = self.operacion\_actual[:-1] + valor

self.actualizar\_pantalla()

return

self.operacion\_actual += valor

self.actualizar\_pantalla()

# ================= CALCULO =================

def realizar\_calculo(self):

if not self.operacion\_actual or self.operacion\_actual[-1] in "+-\*/":

self.pantalla.delete(0, tk.END)

self.pantalla.insert(tk.END, "ERROR: Incompleta")

self.operacion\_actual = ""

return

try:

resultado = eval(self.operacion\_actual)

if isinstance(resultado, float):

resultado = round(resultado, 2)

# Mostrar sin decimales si es entero

if resultado == int(resultado):

resultado = int(resultado)

# Actualizar historial

self.lista\_historial.insert(tk.END, f"{self.operacion\_actual} = {resultado}")

self.lista\_historial.yview(tk.END)

self.pantalla.delete(0, tk.END)

self.pantalla.insert(tk.END, resultado)

self.operacion\_actual = str(resultado)

self.resultado\_mostrado = True

except ZeroDivisionError:

self.pantalla.delete(0, tk.END)

self.pantalla.insert(tk.END, "ERROR: ÷0")

self.operacion\_actual = ""

except:

self.pantalla.delete(0, tk.END)

self.pantalla.insert(tk.END, "ERROR")

self.operacion\_actual = ""

# ================= BORRAR ULTIMO CARACTER =================

def borrar\_ultimo\_caracter(self):

if self.operacion\_actual:

self.operacion\_actual = self.operacion\_actual[:-1]

self.actualizar\_pantalla()

# ================= PANTALLA =================

def actualizar\_pantalla(self):

self.pantalla.delete(0, tk.END)

self.pantalla.insert(tk.END, self.operacion\_actual)

# ================= MOSTRAR/OCULTAR HISTORIAL =================

def mostrar\_ocultar\_historial(self):

if self.historial\_visible:

self.marco\_historial.pack\_forget()

self.historial\_visible = False

else:

self.marco\_historial.pack(side=tk.RIGHT, fill=tk.Y)

self.historial\_visible = True

# ================= MAIN =================

if \_\_name\_\_ == "\_\_main\_\_":

raiz = tk.Tk()

app = Calculadora(raiz)

raiz.mainloop()

**Observaciones**

* El código sigue las convenciones de estilo de Python (PEP8).
* Incluye manejo de errores comunes (división por cero, operaciones incompletas).
* El historial no se guarda entre sesiones, solo se conserva mientras la aplicación está en ejecución.

**Conclusión**

El presente documento concentra el código fuente completo de la calculadora, sirviendo como respaldo técnico para consultas, mantenimiento y futuras mejoras del sistema.