**TALLER TKINTER Y GIT**
**Programación de Computadores**
**Grupos 9 y 12**

**Objetivos:**

1. Utilizar la librería Tkinter de Python para mostrar interfaces gráficas y de esta manera hacer amigable al usuario la aplicación. Seguir el tutorial que se presenta a continuación y mejorar con más widgets de acuerdo al sitio web recomendado.
2. Utilizar la herramienta Git para versionar la aplicación y Github para almacenar los archivos de la aplicación en un repositorio

**Ejercicio 1:**

Vamos a crear una aplicación de **calculadora** que le permita al usuario ingresar dos números reales y un operador (suma, resta, multiplicación, división y exponenciación) y le mostrará en pantalla el resultado de operar los dos números. Para ello, se debe seguir el tutorial descrito a continuación:

1. Crear un archivo .py e importar las librerías Tkinter y ttk:

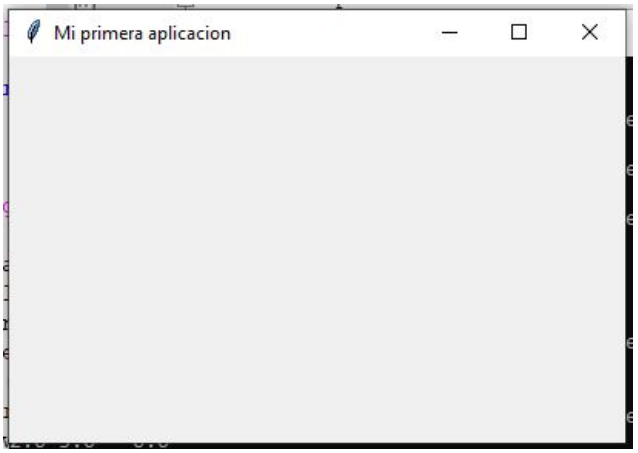
```
import tkinter as tk
from tkinter import ttk
```

2. Definir la función **init_window,** donde se creará la pantalla de la aplicación, para ello, inicializamos la variable **window** que será de tipo **tk.window,** propia de la librería Tkinter. Es importante que al final se llame a la función **mainloop()** que pertenece a la clase Tk, es decir, es propia de la ventana que creamos.

   La función **mainloop** permite que la ventana espere cualquier interacción del usuario hasta que éste la cierra.

```
def init_window():

    window = tk.Tk() #crear la pantalla
    window.title('Mi primera aplicacion') #agregar titulo a la pantalla
    # Establecer el tamanio de la pantalla (ancho: 400px y largo: 250px)
    window.geometry('400x250')

    window.mainloop()
```

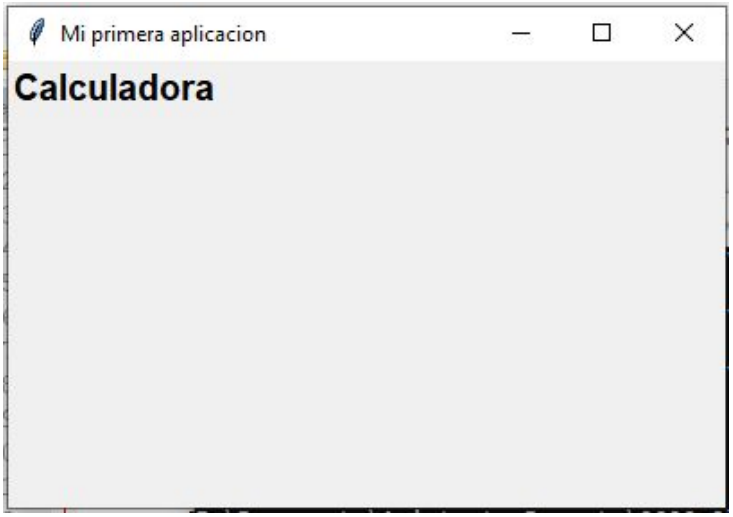
   Cuando se corra el programa, debe generarse la pantalla:




3. **Agregar una etiqueta:** Con la clase **Label** que hace parte de la librería Tkinter, se pueden agregar etiquetas, para ello, inicializamos una variable **label** la ubicamos dentro de la pantalla:

```
def init_window():

    window = tk.Tk() #crear la pantalla
    window.title('Mi primera aplicacion') #agregar titulo a la pantalla
    # Establecer el tamanio de la pantalla (ancho: 400px y largo: 250px)
    window.geometry('400x250')

    #crear una etiqueta con fuente Arial bold y tamanio 15
    label = tk.Label(window, text='Calculadora', font=('Arial bold', 15))
    #ubicar la etiqueta en la columna y fila 0 de la pantalla
    label.grid(column = 0, row = 0)

    window.mainloop()
```

Así debe verse la pantalla:



4. **Agregar campos de texto:** Gracias a la clase Entry de la librería Tkinter, es posible darle la posibilidad al usuario de ingresar valores a través de campos de texto, para ello, inicializamos dos variables de tipo **tk.Entry** y las ubicamos dentro de la pantalla:

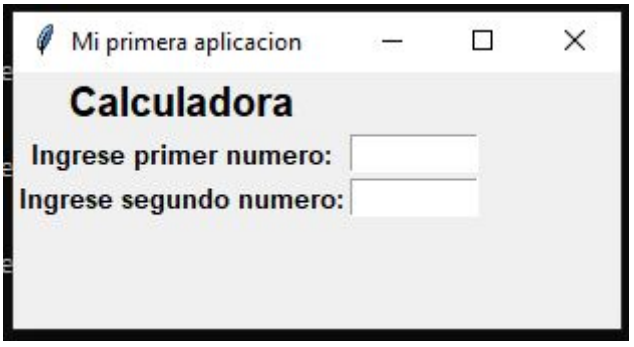
```
#Agregar dos campos de texto
entrada1 = tk.Entry(window, width=10)
entrada2 = tk.Entry(window, width=10)

entrada1.grid(column = 1, row = 1)
entrada2.grid(column = 1, row = 2)
```

Para que el usuario sepa qué valores debe ingresar, es importante agregar otras etiquetas como se hizo en el paso 3.

```
#Agregar dos etiquetas para indicarle al usuario los valores que debe ingresar
label_entrada1 = tk.Label(window, text = 'Ingrese primer numero:', font=('Arial bold', 10))
label_entrada1.grid(column = 0, row = 1)

label_entrada2 = tk.Label(window, text = 'Ingrese segundo numero:', font=('Arial bold', 10))
label_entrada2.grid(column = 0, row = 2)
```

**Nota:** Tener en cuenta en qué posición se ubican los widgets (etiquetas, campos de texto), ya que con una ubicación incorrecta, éstos se pueden superponer, por esta razón, las etiquetas de las entradas estarán en la columna 0 y en las filas 1 y 2 para que concuerden con la ubicación de los campos de texto, que estarán ubicados en las columnas 1 y filas 1 y 2.

Así debe verse la pantalla:



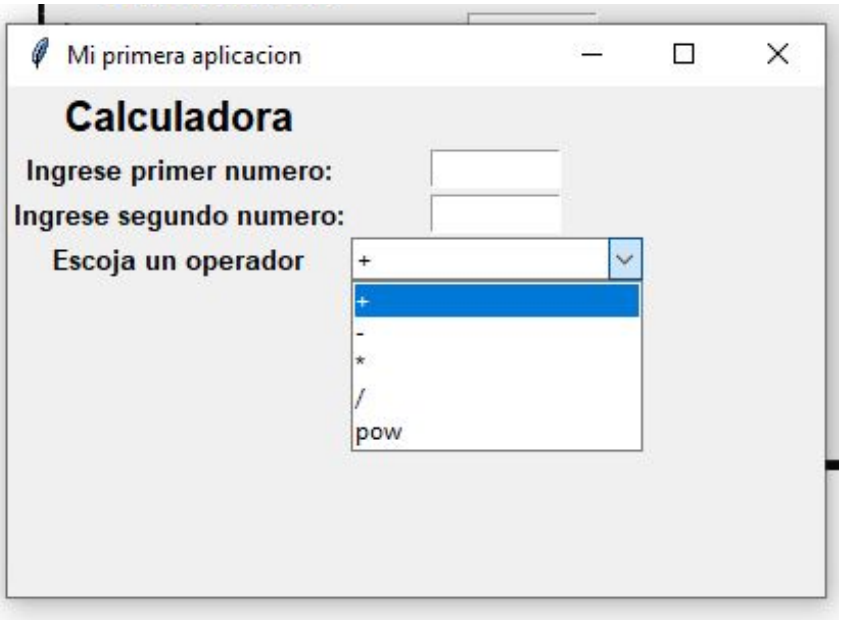

5. **Agregar un combobox:** La librería Tkinter.ttk nos da la posiblidad de agregar un seleccionador, de manera que se pueden mostrar varias opciones al usuario y éste escoge una de ellas, para ello creamos una variable de tipo **ttk.Combobox.**

Una vez creada, se le asigna los valores a seleccionar:

```
#Crear una etiqueta para el seleccionador (combobox)
label_operador = tk.Label(window, text = 'Escoja un operador', font=('Arial bold', 10))
label_operador.grid(column = 0, row = 3)

#Crear un seleccionador (combobox)
combo_operadores = ttk.Combobox(window)
#Asignar los valores del seleccionador a traves de su atributo values
combo_operadores['values'] = ['+', '-', '*', '/', 'pow']
#Asignar por defecto una opcion seleccionada: 0 es el indice de los valores
combo_operadores.current(0) #set the selected item
#Ubicar el seleccionador
combo_operadores.grid(column=1, row=3)
```

También se agrega una etiqueta para hacerle saber al usuario qué dato es que el va a seleccionar.

Así debe verse la pantalla:




6. **Agregar botón:** Se agregará un botón que permita calcular y mostrar el resultado de la operación, para ello vamos a crear una variable de tipo **tk.Button** de la librería Tkinter.

   6.1. **Agregar etiqueta resultado:** Se agregará una etiqueta o texto para mostrar el resultado, de manera que cada vez que el usuario presione el botón **calcular,** se muestre el resultado en la pantalla.

```
#agregar etiqueta para mostrar el resultado de la operacion en pantalla
label_resultado = tk.Label(window, text='Resultado: ', font=('Arial bold', 15))
label_resultado.grid(column = 0, row = 5)
```

**6.1. Definir la función calculadora:** Esta función es independiente de las interfaces gráficas que se tengas en el programa y su objetivo es calcular un valor a partir de dos números reales y un operador.

```
def calculadora(num1, num2, operador):

    if operador == '+':
        resultado = num1 + num2
    elif operador == '-':
        resultado = num1 - num2
    elif operador == '*':
        resultado = num1 * num2
    elif operador == '/':
        resultado = round(num1 / num2, 2)
    else:
        resultado = num1 ** num2

    return resultado
```

**6.2. Definir la función click_calcular:** Esta función recibirá como parámetros, la etiqueta **label_resultado** y los valores ingresados por el usuario (definidos en el paso 4 y 5). Se debe tener en cuenta que todo lo que ingrese el usuario es un **string,** por lo tanto, es importante hacer las conversiones de tipo necesarias. Así mismo, como en el paso anterior 6.1. se definió la función **calculadora,** se hará uso de ella:

```
def click_calcular(label, num1, num2, operador):

    #Conversion de valores
    valor1 = float(num1)
    valor2 = float(num2)

    #Calculo dados los valores y el operador
    res = calculadora(valor1, valor2, operador)

    #Actualizacion del texto en la etiqueta
    label.configure(text = 'Resultado: ' + str(res))
```

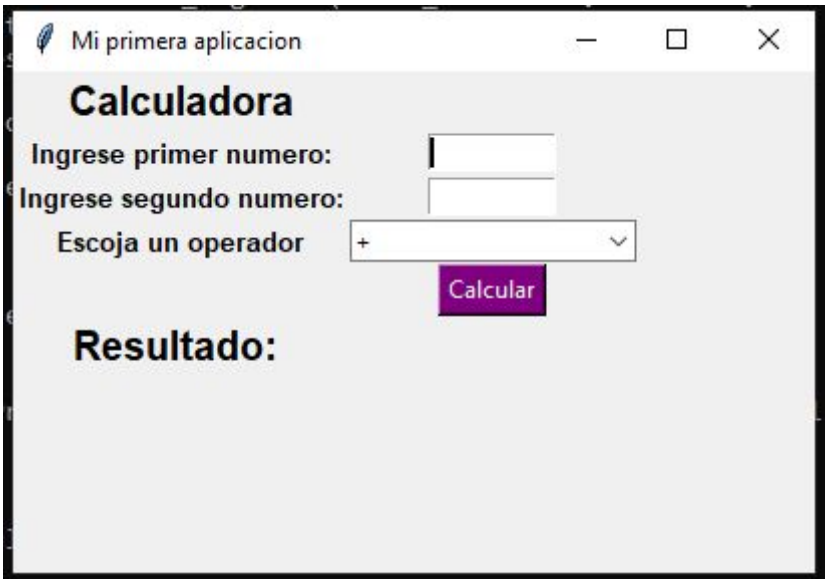
6.3. **Creación botón calcular:** Se creará el botón **calcular,** que será de la clase **Tk.button.** Un botón tiene ciertas propiedades, entre ellas, el color del fondo (bg), el color de la fuente (fg) y el comando (command) que se ejecutará cada vez que el usuario lo presione.

Para este caso, el comando será la función **click_calcular,** como esta función es requiere parámetros, una manera de enviarlos será a través de una función **lambda (**que se estudiará en clase).

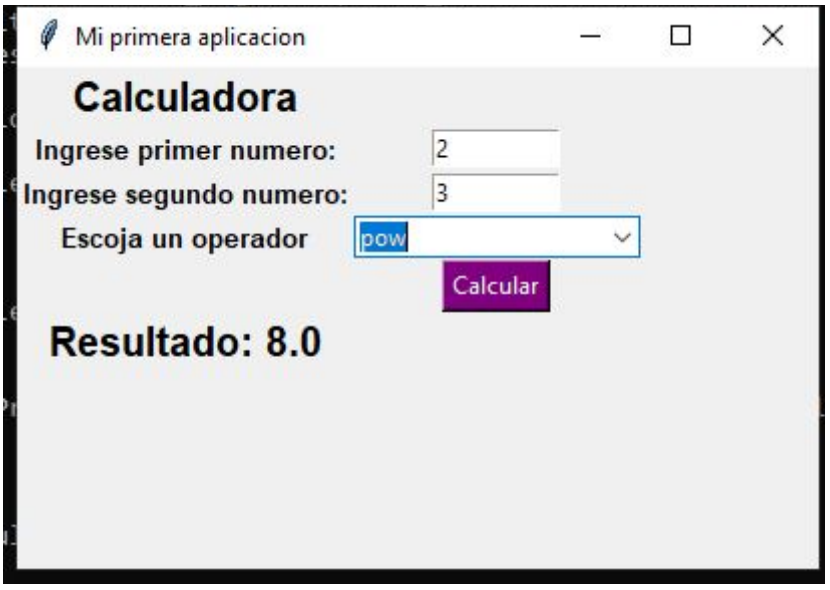
```
#Boton calcular
boton = tk.Button(window,
                  command = lambda: click_calcular(
                          label_resultado,
                          entrada1.get(),
                          entrada2.get(),
                          combo_operadores.get()),
                  text='Calcular',
                  bg="purple",
                  fg="white")

boton.grid(column = 1, row = 4)
```

**Nota:** Como parámetros de la función **click_calcular,** se enviará la etiqueta de resultado (para actualizarla), los valores ingresados por el usuario en el paso 4, y para obtenerlo se tiene la función **get** de la clase **Entry,** así mismo, se enviará el operador seleccionado en el paso 5 con la función **get** de la clase **Combobox.**

Así tiene que verse la pantalla:



**Ejemplo:** Con los valores 2, 3 y pow, se obtendrá:




7. **Agregar función main:** Para darle orden al programa y seguir buenas prácticas de programación se implementará la función **main,** que hasta el momento sólo llamará a la función **init_window:**

```
def main():
    init_window()

main()
```

8. **Agregar 3 widgets a la aplicación:** Se sugiere el tutorial de este sitio web: https://likegeeks.com/python-gui-examples-tkinter-tutorial/

**Resumen:** La función **init_window** tiene que verse similar a la siguiente:

```
def init_window():

    window = tk.Tk() #crear la pantalla
    window.title('Mi primera aplicacion') #agregar titulo a la pantalla
    # Establecer el tamanio de la pantalla (ancho: 400px y largo: 250px)
    window.geometry('400x250')

    label = tk.Label(window, text='Calculadora', font=('Arial bold', 15))
    label.grid(column = 0, row = 0)


    label_entrada1 = tk.Label(window, text = 'Ingrese primer numero:', font=('Arial bold', 10))
    label_entrada1.grid(column = 0, row = 1)

    label_entrada2 = tk.Label(window, text = 'Ingrese segundo numero:', font=('Arial bold', 10))
    label_entrada2.grid(column = 0, row = 2)

    entrada1 = tk.Entry(window, width=10)
    entrada2 = tk.Entry(window, width=10)

    entrada1.focus()
    entrada2.focus()

    entrada1.grid(column = 1, row = 1)
    entrada2.grid(column = 1, row = 2)

    label_operador = tk.Label(window, text = 'Escoja un operador', font=('Arial bold', 10))
    label_operador.grid(column = 0, row = 3)
```

```
    combo_operadores = ttk.Combobox(window)
    combo_operadores['values']= ('+', '-', '*', '/', 'pow')
    combo_operadores.current(0) #set the selected item
    combo_operadores.grid(column=1, row=3)

    #agregar etiqueta para mostrar el resultado de la operacion en pantalla
    label_resultado = tk.Label(window, text='Resultado: ', font=('Arial bold', 15))
    label_resultado.grid(column = 0, row = 5)

    #Boton calcular
    boton = tk.Button(window,
                      command = lambda: click_calcular(
                              label_resultado,
                              entrada1.get(),
                              entrada2.get(),
                              combo_operadores.get()),
                      text='Calcular',
                      bg="purple",
                      fg="white")

    boton.grid(column = 1, row = 4)

    window.mainloop()
```

**Ejercicio 2:**

**Github** es una plataforma donde nosotros podemos:

1. Almacenar el código fuente de nuestros proyectos en **repositorios.**
2. Versionar nuestros proyectos, de manera que si cometemos algún error en una versión, podemos devolvernos a una versión anterior que sea funcional o hacer nuevos ajustes a nuestros proyectos.
3. Trabajar colaborativamente en un mismo proyecto con otras personas, de manera que los cambios de todos los integrantes de un equipo se vean reflejados en el proyecto.
4. Trabajar colaborativamente en proyectos reconocidos a nivel mundial. De esta manera Github es conocida como una red social para desarrolladores.

**Git** es un sistema de control de versiones desarrollado por Linus Torvalds (el hombre que creó Linux). Un **sistema de control de versiones** ayuda a registrar los cambios realizados en el código fuente de nuestros proyectos, así como el seguimiento acerca de quién hizo un cambio y a qué hora, esto resulta ser muy útil, ya que es usual que nuestros proyectos necesitan de nuevas funcionalidades, corrección de errores, actualizar una versión o devolverse a una versión funcional, etc.

**Repositorio:** Es un directorio donde se almacenan todos los archivos de código fuente de nuestros proyectos. Puede estar almacenado en Github y/o en un repositorio local en el computador personal del desarrollador.

**Rama (branch):** Es una copia del repositorio que recibe los cambios de uno o varios desarrolladores con el objetivo de **unir** todos los cambios del proyecto.

Para esta sección del taller, se subirá el mini-proyecto que se realizó en la sección anterior a un repositorio, para ello siga las instrucciones:

1. Crear una cuenta en **Github (https://github.com/)** con el correo institucional, si se quiere acceder a los beneficios del programa **GitHub Student Developer Pack** (**https://education.github.com/pack**).

2. Para esta actividad se debe seguir el tutorial descrito en este sitio web: https://docs.github.com/es/free-pro-team@latest/github/getting-started-with-github/create-a-repo

3. Suba su mini-proyecto de la calculadora a un repositorio en Github, para ello siga las instrucciones que se enuncian en el siguiente tutorial: https://blog.nubecolectiva.com/como-crear-un-repositorio-github-y-subir-los-archivos-de-tu-proyecto-modo-facil/.

**Sugerencia:**

Dado que el código fuente del proyecto debe estar almacenado a un repositorio de Github y cada integrante del equipo debe ser un colaborador, es decir, que cada uno debe subir sus cambios y aportes al repositorio, se sugiere que siga las instrucciones de este videotutorial, que explica en detalle algunos de los comandos de **git** para ser usados desde nuestro computador: https://www.youtube.com/watch?v=_MIxdqAGGYQ