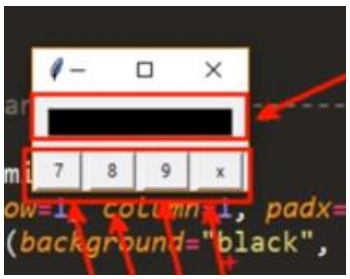
PROGRAMACION 2

Interfaz Gráfica

Esc. Sup. De Comercio N° 49 Cap. Gral. J.J. de Urquiza

Técnico Superior en Desarrollo de Software

En la clase de hoy vamos a integrar todo lo que vimos hasta ahora de interfaz gráfica realizando una aplicación de Calculadora.



La calculadora tendrá arriba el visor y luego abajo los botones de números y operaciones.

Usaremos una grilla de cuatro (4) columnas y 5 filas.

En la primera fila pondremos el visor. Para que el visor de la calculadora ocupe las 4 columnas y estén alineadas, sino lo pongo la primer columna (la del botón 7) va a tener el ancho del visor y quedaría de diferente tamaño que las otras tres

```
from tkinter import *

raiz=Tk()

miFrame=Frame(raiz)
miFrame.pack()

operacion=""
reset_pantalla=False
resultado=0

#-------------pantalla ---------------
numeroPantalla=StringVar()

pantalla=Entry(miFrame, textvariable=numeroPantalla)
pantalla.grid(row=1, column=1,padx=10, pady=10, columnspan=4)
pantalla.config(background="#FFB833", fg="#6133FF", justify="right")

#-------------teclas---------------
```

Columnspan 4 con esta instrucción le indicamos que agrupe las 4 columnas.

Fila 1 que ocupa las 4 columnas

El espacio que aparece desde el visor y el borde es el padx y pady de 10px que le hemos dejado

Fila 2 y siguientes con sus 4 columnas

```
#-------------fila 1 ---------------
boton7=Button(miFrame, text="7", width=3, command=lambda:numeroPulsado("7"))
boton7.grid(row=2, column=1)
boton8=Button(miFrame, text="8", width=3, command=lambda:numeroPulsado("8"))
boton8.grid(row=2, column=2)
boton9=Button(miFrame, text="9", width=3, command=lambda:numeroPulsado("9"))
boton9.grid(row=2, column=3)
botonDiv=Button(miFrame, text="/", width=3)
botonDiv.grid(row=2, column=4)
#-------------fila 2 ---------------
boton4=Button(miFrame, text="4", width=3, command=lambda:numeroPulsado("4"))
boton4.grid(row=3, column=1)
boton5=Button(miFrame, text="5", width=3, command=lambda:numeroPulsado("5"))
boton5.grid(row=3, column=2)
boton6=Button(miFrame, text="6", width=3, command=lambda:numeroPulsado("3"))
boton6.grid(row=3, column=3)
botonMult=Button(miFrame, text="*", width=3)
botonMult.grid(row=3, column=4)
#-------------fila 3 ---------------
boton1=Button(miFrame, text="1", width=3, command=lambda:numeroPulsado("1"))
boton1.grid(row=4, column=1)
boton2=Button(miFrame, text="2", width=3, command=lambda:numeroPulsado("2"))
boton2.grid(row=4, column=2)
boton3=Button(miFrame, text="3", width=3, command=lambda:numeroPulsado("3"))
boton3.grid(row=4, column=3)
botonRest=Button(miFrame, text="-", width=3)
botonRest.grid(row=4, column=4)
#-------------fila 4 ---------------
boton0=Button(miFrame, text="0", width=3,command=lambda:numeroPulsado("0"))
boton0.grid(row=5, column=1)
botonComa=Button(miFrame, text=".", width=3,command=lambda:numeroPulsado("."))
botonComa.grid(row=5, column=2)
botonIgual=Button(miFrame, text="=", width=3, command=lambda:igual())
botonIgual.grid(row=5, column=3)
botonSuma=Button(miFrame, text="+", width=3, command=lambda:suma(numeroPantalla.get()))
botonSuma.grid(row=5, column=4)

raiz.mainloop()
```

Una vez que diseñamos la pantalla tenemos que hacer que asocie lo que pulsamos en los botones y que vaya apareciendo en el visor.

Definimos una variable como stringVar y la asociamos al visor (como vimos en clases anteriores)

Luego definimos una función la cuál lo que debe hacer es escribir los números que pulsamos en el visor que voy a llamar mediante la opción command de Button.

Además la función debe ir agregando a lo que está en el visor lo que voy pulsando

Get() obtiene la información que hay en el cuadro

Set() asigna valor

`numeroPantalla.set(numeroPantalla.get()+"4")` obtenemos lo que ya hay en pantalla con get() y luego le agregamos el valor del botón pulsado

Funciones Lambda o anónimas

En *Python*, una función Lambda se refiere a una **pequeña función anónima**. Las llamamos "funciones anónimas" porque técnicamente carecen de nombre.

Al contrario que una función normal, no la definimos con la palabra clave estándar *def* que utilizamos en *Python*. En su lugar, las funciones Lambda se definen como una **línea que ejecuta una sola expresión**. Este tipo de funciones pueden tomar cualquier número de argumentos, pero solo pueden tener una expresión.

Sintaxis básica

Todas las **funciones Lambda** en *Python* tienen exactamente la misma sintaxis:

```
#Escribo p1 y p2 como parámetros 1 y 2 de la función.lambda p1, p2: expresión
```

Como mejor te lo puedo explicar es enseñándote un ejemplo básico, vamos a ver una **función normal** y un ejemplo de **Lambda**:

```
#Aquí tenemos una función creada para sumar.
def suma(x,y):
    return(x + y)#Aquí tenemos una función Lambda que también suma.
lambda x,y : x + y#Para poder utilizarla necesitamos guardarla en una variable.
suma_dos = lambda x,y : x + y
```

Cuando presiona un botón de operación tengo que guardar el valor en pantalla y hacer que desaparezca el valor que había hasta ahora.
Necesito una variable global para ir guardando la operación que quiero realizar y también resetear la pantalla.
Si la variable global está vacía quiere decir que pulse el botón de operación y por lo tanto no tengo que seguir concatenando en el visor sino guardar el valor y poner el visor en blanco.

También necesito una variable de tipo global para ir almacenar los valores de las operaciones. Por eso definimos las variables globales `operacion resultado y reset_pantalla.`

Tenemos que tener en cuenta que todo lo que se ingresa por un cuadro de texto es considerado string por lo que para operar debemos usar la función int() o float para convertirlo a número.

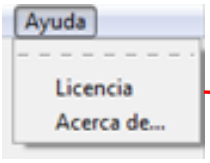
```
#-------------teclas---------------

def numeroPulsado(num):
    global operacion
    global reset_pantalla

    if reset_pantalla!=False:
        numeroPantalla.set(num)
        reset_pantalla=False
    else:
        numeroPantalla.set(numeroPantalla.get()+ num)

#------------- Suma ---------------
def suma(num):
    global operacion
    global resultado
    global reset_pantalla
    resultado+=int(num)
    operacion="suma"
    reset_pantalla=True
    numeroPantalla.set(resultado)

#------------- Igual ---------------
def igual():
    global resultado
    global operacion
    if operacion=="suma":
        numeroPantalla.set(resultado+int(numeroPantalla.get()))
        resultado=0
```

Podríamos modificar para que opere con decimales no sólo con enteros, Y que controle que una vez que puse una coma no me deje volver a poner otra.

Menu

```
from tkinter import *

root=Tk()

barraMenu=Menu(root)
root.config(menu=barraMenu, width=300, height=300)

archivoMenu=Menu(barraMenu, tearoff=0)
archivoMenu.add_command(label="Nuevo")
archivoMenu.add_command(label="Guardar")
archivoMenu.add_command(label="Guardar Como")
archivoMenu.add_separator()
archivoMenu.add_command(label="Cerrar")
archivoMenu.add_command(label="Salir")

archivoEdicion=Menu(barraMenu, tearoff=0)
archivoEdicion.add_command(label="Copiar")
archivoEdicion.add_command(label="Cortar")
archivoEdicion.add_command(label="Pegar")

archivoHerramientas=Menu(barraMenu, tearoff=0)

archivoAyuda=Menu(barraMenu, tearoff=0)
archivoAyuda.add_command(label="Licencia")
archivoAyuda.add_command(label="Acerca de...")

barraMenu.add_cascade(label="Archivo", menu=archivoMenu)
barraMenu.add_cascade(label="Edición", menu=archivoEdicion)
barraMenu.add_cascade(label="Herramientas", menu=archivoHerramientas)
barraMenu.add_cascade(label="Ayuda", menu=archivoAyuda)

root.mainloop()
```

Definimos un objeto (variable) al que le asignamos el widget(clase) Menu , lo y le damos ancho y alto



Por defecto nos agrega un barra horizontal discontinua dentro del submenú para que no lo haga dentro del constructor del menú ponemos la opción **tearoff=0**

Para agregar un submenú dentro de archivoMenu y lo vamos a hacer el el método add_command

Creamos los elementos del menú. Y tenemos que decirle cuál es el texto que tienen estos elementos del Menu. Para eso tenemos que usar el método add_cascade y le vamos asignando los elementos

Si queremos agregar una separación o agrupamiento dentro de las opciones. Nos situamos en el código antes de la opción donde queremos agregar el separador lo hacemos con el método **add_separator()**

*args y **kwargs en Python.

https://j2logo.com/args-y-kwargs-en-python/

https://python-intermedio.readthedocs.io/es/latest/args_and_kwargs.html

¿Qué es if __name__ == "__main__":?

https://es.stackoverflow.com/questions/32165/qu%C3%A9-es-if-name-main/32185