Material Imprimible

Curso de Aplicaciones Prácticas de Python

---------- generamos el scrollbar siguiendo el codigo de textocomentario

scrollVert=Scrollbar(miFrame, command=textoComentario.yview) #se genera el scrollbar

scrollVert.grid(row=4, column=2) # lo posisionamos para mostrarlo

PERO NO TOMA FORMA DE SCROLL COMO SOLEMOS VER ADAPTADO AL CUADRO Y EL BOTON CENTRAL QUE TENGA MOVIMIENTO E INTERACCION

----------Solucion

scrollVert.grid(row=4, column=2, sticky="nsew")

TAMAÑO Y BOTON INTERATIVO SOLUCIONADO

----------Ajustamos el botón para que acompañe la línea de texto

textoComentario.config(yscrollcommand=scrollVert.set)

Link manual text online

https://www.tutorialspoint.com/python/tk_text.htm

----------generamos botón

botonEnvio=Button(raiz, text="Enviar")

botonEnvio.pack()

llamamos a la funcion

botonEnvio=Button(raiz, text="Enviar" , command=codigoBoton)

botonEnvio.pack()

generamos la función para que muestre en un entry

def codigoBoton():

    minombre.set("Juan") # funcion set para darle valor a una variable funcion get para obtener datos de una variable

----------creamos variable que usamos en la función y generamos en el entry un textvariable donde colocamos el return de la función

```
minombre=StringVar()

cuadroApellido=Entry(miFrame, textvariable=minombre)#asociamos el cuadro a la variable
```

----------Queda final----------

```
from tkinter import *

raiz =Tk()

miFrame=Frame(raiz, width=1200, height=600)
miFrame.pack()

minombre=StringVar()

cuadroApellido=Entry(miFrame, textvariable=minombre)
cuadroApellido.grid(row=0, column=1, padx=10, pady=10)
cuadroApellido.config(fg="red",justify="right")

cuadroNombre=Entry(miFrame)
cuadroNombre.grid(row=1, column=1, padx=10, pady=10)
cuadroNombre.config(fg="red",justify="right")

cuadroDireccion=Entry(miFrame)
cuadroDireccion.grid(row=2, column=1, padx=10, pady=10)
cuadroDireccion.config(fg="red",justify="right")

nombreLabel=Label(miFrame, text="Nombre:")
nombreLabel.grid(row=0, column=0, padx=10, pady=10)
```

```
apellidoLabel=Label(miFrame, text="Apellido:")
apellidoLabel.grid(row=1, column=0, padx=10, pady=10)

direccionLabel=Label(miFrame, text="Direccion:")
direccionLabel.grid(row=2, column=0, padx=10, pady=10)

cuadroPass=Entry(miFrame)
cuadroPass.grid(row=3, column=1, padx=10, pady=10)
cuadroPass.config(show="*",fg="red",justify="right")

PassLabel=Label(miFrame, text="Password:")
PassLabel.grid( row=3, column=0, padx=10, pady=10)

comentariosLabel=Label(miFrame, text="Comentario:")
comentariosLabel.grid( row=4, column=0, padx=10, pady=10)

textoComentario=Text(miFrame, width=16, height=5)
textoComentario.grid( row=4, column=1, padx=10, pady=10)


scrollVert=Scrollbar(miFrame, command=textoComentario.yview)
scrollVert.grid(row=4, column=2, sticky="nsew")
textoComentario.config(yscrollcommand=scrollVert.set)

def codigoBoton():

        minombre.set("Juan")

botonEnvio=Button(raiz, text="Enviar", command=codigoBoton)
```

botonEnvio.pack()

raiz.mainloop()

Calculadora entorno


```
from tkinter import *

raiz =Tk()

miFrame=Frame(raiz, width=1200, height=600)
miFrame.pack()

raiz.mainloop()
```

----------Generamos la pantalla o visor de números

Generamos divisiones porque vamos a usar varias líneas de código

#----------Pantalla----------

pantalla=Entry(miFrame)

pantalla.grid(row=1, column=1, padx=10, pady=10)

pantalla.config(background="black", fg="#03f943", justify="right")

Cambiar el state no permite al usuario ingresar datos al entry, aunque me quita atributos estéticos del mismo

pantalla=Entry(miFrame, textvariable=numeroPantalla, state='disabled')

Ahora ingresamos la primera fila de números

#----------Fila1----------

boton7=Button(miFrame, text="7", width=3)

boton7.grid(row=2, column=1)

boton8=Button(miFrame, text="8", width=3)

boton8.grid(row=2, column=2)

boton9=Button(miFrame, text="9", width=3)

boton9.grid(row=2, column=3)

botonDiv=Button(miFrame, text="/", width=3)

botonDiv.grid(row=2, column=4)

VEMOS QUE LA PANTALLA OCUPA UN ANCHO QUE AGRANDA LA COLUMNA 1, PERO DEBERÍA OCUPAR TODO EL ANCHO Y LOS NÚMEROS ABAJO COMO EN LA CALCULADORA DEL EJEMPLO

columnspan=4 # atributo usado diseño web para que combine varias celdas en este caso 4

pantalla.grid(row=1, column=1, padx=10, pady=10 , columnspan=4)

Actividad:

Realizar las otras tres filas de botones

GENERAMOS LA SEGUNDA LINEA DE BOTONES

```
#---------------Fila2---------------
Boton4=Button(miFrame, text="4", width=3)
Boton4.grid(row=3, column=1)
Boton5=Button(miFrame, text="5", width=3)
Boton5.grid(row=3, column=2)
Boton6=Button(miFrame, text="6", width=3)
Boton6.grid(row=3, column=3)
botonMult=Button(miFrame, text="X", width=3)
botonMult.grid(row=3, column=4)

#---------------Fila3---------------
boton1=Button(miFrame, text="1", width=3)
boton1.grid(row=4, column=1)
boton2=Button(miFrame, text="2", width=3)
boton2.grid(row=4, column=2)
boton3=Button(miFrame, text="3", width=3)
boton3.grid(row=4, column=3)
botonMen=Button(miFrame, text="-", width=3)
botonMen.grid(row=4, column=4)

#---------------Fila4---------------
Boton0=Button(miFrame, text="0", width=3)
Boton0.grid(row=5, column=1)
BotonComa=Button(miFrame, text=",", width=3)
BotonComa.grid(row=5, column=2)
BotonIgual=Button(miFrame, text="=", width=3)
```

```
BotonIgual.grid(row=5, column=3)
botonSum=Button(miFrame, text="+", width=3)
botonSum.grid(row=5, column=4)
```

Funcionalidad de los botones

Usaremos funciones que sean capaz de escribir un texto en pantalla cada vez que la llamamos.

Para lograr que cuando apriete un botón aparezca un número.

Generamos una variable

```
numeroPantalla=StringVar() # se genera una variable del tipo Str
```

Ahora asociamos la variable creada a la pantalla.

En el constructor de Entry

```
pantalla=Entry(miFrame, textvariable=numeroPantalla)
```

Creamos función seguido de la pantalla pulsación de teclados.

```
#----------------------Pulsaciones teclado
def numeroPulsado():
        numeroPantalla.set("4") #muestra el número cuatro en pantalla
```

buscamos el botón 4 para asociar la función----------

```
Boton4=Button(miFrame, text="4", width=3, command=numeroPulsado)
```

Pero escribe un solo cuatro cada vez que llamamos a la función----------

Se pisa a sí mismo.

Debemos indicar que vea lo que hay en pantalla y que lo agregue.

Con get obtenemos lo que hay en pantalla----------

```
numeroPantalla.set(numeroPantalla.get() + "4") # coloca en pantalla agregando lo que ya tiene
```

Hasta acá la función llama a colocar el numero 4 deberíamos cambiar para que funciones en los otros botones.

Pasamos por parámetros a la función el valor que debería tener según la tecla presionada.

def numeroPulsado(num):

Y luego en el botón le pasamos el argumento en formato texto porque así definimos al principio en la variable numeroPantalla----------
Boton4=Button(miFrame, text="4", width=3, command=numeroPulsado("4"))

Pero dará un error

Aparece siempre el número cuatro, y si apretamos nuevamente el 4, no aparece nada más que un solo cuatro.

En Python cuando utilizamos los paréntesis en una función indican que ejecutamos el código de la función y lo almacenamos en command sin esperar presionar el botón. Es como una llamada automática.

En nuestro programa no queremos ejecutar la función, sino tener una referencia y que se ejecute cuando el usuario presione la tecla o llame a la función.

FUNCIONES LAMBDA

Lambda de Python

Una función lambda es una pequeña función anónima.
Una función lambda puede tomar cualquier número de argumentos, pero solo puede tener una expresión.

Sintaxis

lambda *arguments* : *expression*
Se ejecuta la expresión y se devuelve el resultado:

Ejemplo

Una función lambda que agrega 10 al número pasado como argumento e imprime el resultado:

```
x = lambda a: a + 10
print(x(5))
```

Las funciones de Lambda pueden tomar cualquier número de argumentos:

Ejemplo

Función lambda que multiplica el argumento a con el argumento b e imprime el resultado

```
x = lambda a, b: a * b
print(x(5, 6))
```

Función lambda que suma el argumento a, b y c e imprime el resultado:

```
x = lambda a, b, c: a + b + c
print(x(5, 6, 2))
```

¿Por qué utilizar Lambda Functions?

El poder de lambda se muestra mejor cuando se utilizan como una función anónima dentro de otra función.

Supongamos que tiene una definición de función que toma un argumento y ese argumento se multiplicará por un número desconocido:

```
def myfunc(n):
return lambda a : a * n
```

Utilice esa definición de función para crear una función que siempre duplique el número que envíe:

```
def myfunc(n):
return lambda a : a * n
mydoubler = myfunc(2)
```

print(mydoubler(11))

O bien, utilice la misma definición de función para crear una función que siempre *triplica* el número que envía:

```
def myfunc(n):
return lambda a : a * n
mytripler = myfunc(3)
print(mytripler(11))
```

O bien, utilice la misma definición de función para realizar ambas funciones, en el mismo programa:

```
def myfunc(n):
return lambda a : a * n
mydoubler = myfunc(2)
mytripler = myfunc(3)
print(mydoubler(11))
print(mytripler(11))
```

Utilice funciones lambda cuando se requiera una función anónima durante un breve período de tiempo.

```
Boton4=Button(miFrame, text="4", width=3, command=lambda:numeroPulsado("4"))
```

Solo se realiza la ejecución de la función cuando se utiliza el command acción del botón.

Ejercicio

Realizar estas acciones en todos los botones, menos en los botones de calcular.

COMPLETO QUEDA ASÍ

```
from tkinter import *
```

```
raiz =Tk()

miFrame=Frame(raiz, width=1200, height=600)
miFrame.pack()

numeroPantalla=StringVar()
pantalla=Entry(miFrame, textvariable=numeroPantalla)
pantalla.grid(row=1, column=1, padx=10, pady=10, columnspan=4)
pantalla.config(background="black", fg="#03f943", justify="right")
#----------Pulsaciones teclado
def numeroPulsado(num):
        numeroPantalla.set(numeroPantalla.get() + num) #muestra cuatro en pantalla

#----------Fila1----------
boton7=Button(miFrame, text="7", width=13, command=lambda:numeroPulsado("7"))
boton7.grid(row=2, column=1)
boton8=Button(miFrame, text="8", width=13, command=lambda:numeroPulsado("8"))
boton8.grid(row=2, column=2)
boton9=Button(miFrame, text="9", width=13, command=lambda:numeroPulsado("9"))
boton9.grid(row=2, column=3)
botonDiv=Button(miFrame, text="/", width=13)
botonDiv.grid(row=2, column=4)

#----------Fila2----------
Boton4=Button(miFrame, text="4", width=13, command=lambda:numeroPulsado("4"))
Boton4.grid(row=3, column=1)
Boton5=Button(miFrame, text="5", width=13, command=lambda:numeroPulsado("5"))
Boton5.grid(row=3, column=2)
Boton6=Button(miFrame, text="6", width=13, command=lambda:numeroPulsado("6"))
Boton6.grid(row=3, column=3)
```

```
botonMult=Button(miFrame, text="X", width=13)
botonMult.grid(row=3, column=4)

#---------------Fila3---------------
boton1=Button(miFrame, text="1", width=13, command=lambda:numeroPulsado("1"))
boton1.grid(row=4, column=1)
boton2=Button(miFrame, text="2", width=13, command=lambda:numeroPulsado("2"))
boton2.grid(row=4, column=2)
boton3=Button(miFrame, text="3", width=13, command=lambda:numeroPulsado("3"))
boton3.grid(row=4, column=3)
botonMen=Button(miFrame, text="-", width=13)
botonMen.grid(row=4, column=4)

#---------------Fila4---------------
Boton0=Button(miFrame, text="0", width=13, command=lambda:numeroPulsado("0"))
Boton0.grid(row=5, column=1)
BotonComa=Button(miFrame, text=",", width=13, command=lambda:numeroPulsado(","))
BotonComa.grid(row=5, column=2)
BotonIgual=Button(miFrame, text="=", width=13)
BotonIgual.grid(row=5, column=3)
botonSum=Button(miFrame, text="+", width=13)
botonSum.grid(row=5, column=4)

raiz.mainloop()
```

Funcionalidad de la calculadora

Sumamos

Colocamos el número y muestra en pantalla, cuando presionamos SUMA, el valor sigue en pantalla y después tecleo, desaparece el anterior y aparece el nuevo número.

En algún lugar debe almacenar el valor anterior y reseteo de pantalla.

Variable global operación

Almacena la función que el usuario quiere realizar, suma, resta, etc.

Además, permite reset pantalla

operacion="" # **cuando inicia el programa operación es una cadena vacía se crea al inicio del programa**

En esta variable se almacena la palabra SUMA si apretamos SUMA.

Creamos una función o método para que almacene debajo de numeroPulsado

```
#----------Funcion suma
def suma():
    global operacion
    operacion="suma"
```

Ahora ubicamos la tecla suma.

```
botonSum=Button(miFrame, text="+", width=3, command=lambda:suma())
```

Ahora debemos resetear la pantalla para colocar el número siguiente.

Dentro de la función numeroPulsado

Declaro en la función que utilizare la variable global operación.

Y si operación es SUMA, o sea distinto de vacío, muestre el numero en pantalla indicado sin que concatene, de lo contrario que concatene los valores como antes.

```
def numeroPulsado(num):
    global operacion
    if operacion != "":
        numeroPantalla.set(num)
    else:
        numeroPantalla.set(numeroPantalla.get() + num)
```

pero no concatena

Debemos volver la variable operación a vacío para que vuelva a concatenar los números del siguiente valor numérico.

```
def numeroPulsado(num):
        global operacion
        if operacion != "":
                numeroPantalla.set(num)
                operacion=""#ahora a sumar!!!!
        else:
                numeroPantalla.set(numeroPantalla.get() + num)
```

Necesito una variable para que vaya almacenando los resultados de las sumas sucesivas número + número + (muestra resultado) + número + (muestra nuevo resultado)…

Inicio la variable global en cero "0"

```
resultado=0
```

Ahora a la función SUMA.

Esta función también debe contar con la variable global resultado

```
global resultado
```

A esta variable resultado le incrementé el valor numérico anterior, pero este cálculo se realizará cuando presione sobre el botón MÁS. Debo indicar que sume lo que hay en pantalla, por eso debemos enviar un parámetro como lo que hay en pantalla para que se sume en resultado anteriormente.

```
def suma(valor):
        global operacion
        global resultado
        resultado += int (valor) # los cuadros de texto se consideran texto en Python, por eso ponemos int
        operacion="suma"
```

Para que los valores que vayan apareciendo en pantalla, como pasa con las calculadoras, debemos realizar:

```
def suma(valor):
        global operacion
        global resultado

        resultado += int (valor)
        operacion="suma"
        numeroPantalla.set(resultado)
```

Ahora esta función SUMA está preparada para recibir un parámetro y def suma (num). Tiene que ser lo que hay escrito en pantalla en la llamada de la función. Debemos pasarle ese parámetro:

```
botonSum=Button(miFrame, text="+", width=13,
command=lambda:suma(numeroPantalla.get()))
```

Ahora aplicar la última suma con el último número ingresado utilizando el signo igual

Luego de la función SUMA podriamos crear la función el_resultado

```
#----------el_resultado
def el_resultado():
        global resultado# Utilizamos la variable global resultado
#y ahora colocaremos en pantalla (set) el cálculo de resultado, más, tomando lo que hay en pantalla (get)
        numeroPantalla.set(resultado+int(numeroPantalla.get()))
ahora llamar a la función en el botón del igual
BotonIgual=Button(miFrame, text="=", width=3, command=lambda:(el_resultado()))
```

Ahora sí funciona el IGUAL.

Pero si presionamos nuevamente SUMA, me suma el valor almacenado en resultado.

Deberíamos resetear resultado en la función el_resultado

```
resultado=0
```

Actividad

1) Lograr que sume valores decimales
2) No permitir colocar más de una coma
3) Realizar el resto de las operaciones
4) Cambiar características a su gusto: colores, estructuras, etc.
5) Botón volver a cero todo
6) Botón retroceso
7) Validar, no dividir por cero
8) No permitir que siga sumando lo que hay en pantalla más lo acumulado.

```
from tkinter import *

raiz =Tk()

miFrame=Frame(raiz, width=1200, height=600)
miFrame.pack()

operacion=""
resultado=0

numeroPantalla=StringVar()
pantalla=Entry(miFrame, textvariable=numeroPantalla)
pantalla.grid(row=1, column=1, padx=10, pady=10, columnspan=4)
pantalla.config(background="black", fg="#03f943", justify="right")
#----------Pulsaciones teclado
#def numeroPulsado(num):
#        numeroPantalla.set(numeroPantalla.get() + num) #muestra cuatro en pantalla
```

```
def numeroPulsado(num):
        global operacion
        if operacion != "":
                numeroPantalla.set(num)
                operacion=""#ahora a sumar!!!!
        else:
                numeroPantalla.set(numeroPantalla.get() + num)


#--------------------Funcion suma
def suma(num):
        global operacion
        operacion="suma"
        global resultado
        resultado += int (num) # los cuadro de texto se considera texto en Python por eso
ponemos int
        numeroPantalla.set(resultado)

#---------------el_resultado
def el_resultado():
        global resultado# Utilizamos la variable global resultado
                                        #y ahora colocaremos en pantalla (set) el cálculo de
resultado, más, tomando lo que hay en pantalla (get)
        numeroPantalla.set(resultado+int(numeroPantalla.get()))
        resultado=0

#---------------Fila1---------------
operacion=1
boton7=Button(miFrame, text="7", width=13, command=lambda:numeroPulsado("7"))
boton7.grid(row=2, column=1)
```

```
boton8=Button(miFrame, text="8", width=13, command=lambda:numeroPulsado("8"))
boton8.grid(row=2, column=2)
boton9=Button(miFrame, text="9", width=13, command=lambda:numeroPulsado("9"))
boton9.grid(row=2, column=3)
botonDiv=Button(miFrame, text="/", width=13)
botonDiv.grid(row=2, column=4)

#---------------Fila2---------------
Boton4=Button(miFrame, text="4", width=13, command=lambda:numeroPulsado("4"))
Boton4.grid(row=3, column=1)
Boton5=Button(miFrame, text="5", width=13, command=lambda:numeroPulsado("5"))
Boton5.grid(row=3, column=2)
Boton6=Button(miFrame, text="6", width=13, command=lambda:numeroPulsado("6"))
Boton6.grid(row=3, column=3)
botonMult=Button(miFrame, text="X", width=13)
botonMult.grid(row=3, column=4)

#---------------Fila3---------------
boton1=Button(miFrame, text="1", width=13, command=lambda:numeroPulsado("1"))
boton1.grid(row=4, column=1)
boton2=Button(miFrame, text="2", width=13, command=lambda:numeroPulsado("2"))
boton2.grid(row=4, column=2)
boton3=Button(miFrame, text="3", width=13, command=lambda:numeroPulsado("3"))
boton3.grid(row=4, column=3)
botonMen=Button(miFrame, text="-", width=13)
botonMen.grid(row=4, column=4)

#---------------Fila4---------------
Boton0=Button(miFrame, text="0", width=13, command=lambda:numeroPulsado("0"))
Boton0.grid(row=5, column=1)
```

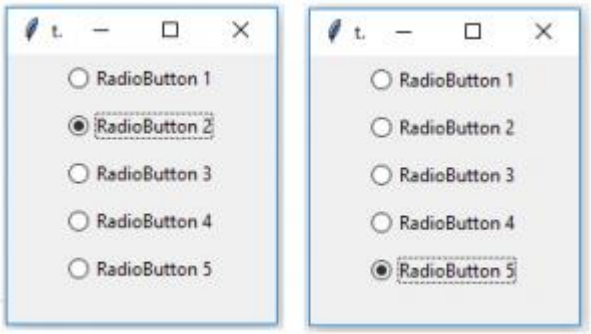
```
BotonComa=Button(miFrame, text=",", width=13, command=lambda:numeroPulsado(","))
BotonComa.grid(row=5, column=2)
BotonIgual=Button(miFrame, text="=", width=13, command=lambda:(el_resultado()))
BotonIgual.grid(row=5, column=3)
BotonSum=Button(miFrame, text="+", width=13,
command=lambda:suma(numeroPantalla.get()))
BotonSum.grid(row=5, column=4)

raiz.mainloop()
```

Botones de radio

El botón de radio es un widget de Tkinter estándar que se utiliza para implementar selección de una o de muchas opciones. Los botones de radio pueden contener texto o imágenes, y pueden asociar una función o método de Python con cada botón.

Cuando se presiona el botón, Tkinter llama automáticamente a esa función o método.



Generamos la plantilla

```
from tkinter import *
raiz=Tk()
raiz.mainloop()
```

colocamos los radio button ubicadas y empaquetadas en la raíz.

```
Radiobutton(raiz, text="Masculino").pack()
Radiobutton(raiz, text="Femenino").pack()
```

Generamos los radio, pero aparecen seleccionadas por defecto las dos y no nos permite deseleccionar.
Hay que asignarle una variable, como así también un valor como en los entry

varOpcion=IntVar() #al igual que en los Entry pero en lugar de str es un int porque almacenara un valor numérico
Radiobutton(raiz, text="Masculino", variable=varOpcion, value=1).pack()
Radiobutton(raiz, text="Femenino", variable=varOpcion, value=2).pack()

Ya le asignamos una variable y un valor a cada uno indicando qué le corresponde a cada radiobutton
Ahora debemos tomar o capturar la selección del usuario.
Colocamos una etiqueta para indicar mejor lo que debe realizar.

Actividad

Realizar una etiqueta encima de los radiobutton, ubicada en la raíz, con el nombre Genero y la empaquetamos.

```
Label(raiz, text="Genero:").pack()
```

Ahora realizamos una función que se ejecuta cuando presionamos en uno de estos button

```
def imprimir():
        print(varOpcion.get())
```

de esta manera la función toma lo que hay en pantalla y lo muestra en consola del sublime

Actividad

Hacer la llamada a la función

Radiobutton(raiz, text="Masculino", variable=varOpcion,value=1, command=imprimir).pack()
Radiobutton(raiz, text="Femenino", variable=varOpcion, value=2, command=imprimir).pack()

Actividad

Generar una manera que me muestre en pantalla y no en consola la opción seleccionada.

Generamos una etiqueta donde será el espacio donde mostrará la selección.

```
etiqueta=Label(raiz)
etiqueta.pack()#debajo de los button
#print(varOpcion.get()) sacamos este print porque no nos va ser útil.
```

Actividad

Crear la función donde cambiamos el valor numérico por el texto a mostrar.

Función

```
_def imprimir():
        if varOpcion.get()==1:
                etiqueta.config(text="Has elegido Masculino")
        else:
                etiqueta.config(text="Has elegido Femenino")
```

Actividad

Agregar un radiobutton más, con las opciones de otros.

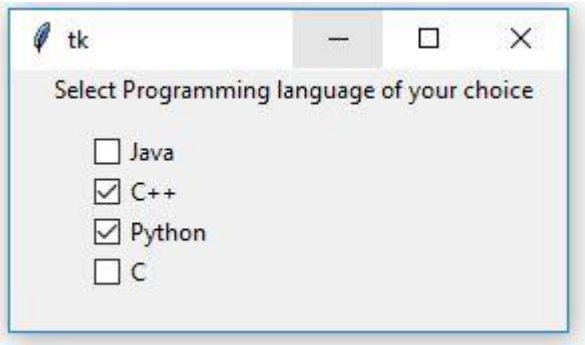
Código final

```
from tkinter import *
raiz=Tk()
varOpcion=IntVar()
def imprimir():
        #print(varOpcion.get())
        if varOpcion.get()==1:
                etiqueta.config(text="Has elegido Masculino")
```

```
        elif varOpcion.get()==2:
                etiqueta.config(text="Has elegido Femenino")
        else:
                etiqueta.config(text="Has elegido Otros / Otras")
Label(raiz, text="Genero:").pack()
Radiobutton(raiz, text="Masculino", variable=varOpcion,value=1, command=imprimir).pack()
Radiobutton(raiz, text="Femenino", variable=varOpcion, value=2, command=imprimir).pack()
Radiobutton(raiz, text="Otros/Otras", variable=varOpcion, value=3, command=imprimir).pack()
etiqueta=Label(raiz)
etiqueta.pack()
raiz.mainloop()
```

CheckButtons

También conocido en la jerga como «Checkbox», es un tipo de botón que permite representar dos estados opuestos (activado/desactivado, encendido/apagado, sí/no, etc.) o bien un estado indeterminado. Y tenemos la posibilidad de seleccionar respuestas múltiples.



Generamos la plantilla

```
from tkinter import *
raiz= Tk()
raiz.title("Lugares Turísticos")
raiz.mainloop()
```

Colocamos el primer Checkbutton

Checkbutton(raiz, text="Brasil").pack()

Agregamos los otros Checkbutton

Checkbutton(raiz, text="Chile").pack()
Checkbutton(raiz, text="Francia").pack()
Checkbutton(raiz, text="España").pack()

Agregamos una imagen
foto=PhotoImage(file="ARG.png") # generamos una variable para almacenar la foto. Solo acepta png o gif. Habría que descargar otra librería
Label(raiz, image=foto).pack() # en una etiqueta podríamos colocar texto o imagen y lo empaquetamos

Actividad

Generar un frame donde colocaremos los checkbutton que creamos y que se manejen independiente de la imagen.
Y colocaremos una etiqueta donde diga “elige destinos”, que esté ubicada en el frame.

Así quedaría hasta el momento nuestra app

```
from tkinter import *
raiz= Tk()
raiz.title("Lugares Turísticos")
foto=PhotoImage(file="ARG.png")
Label(raiz, image=foto).pack()
frame=Frame(raiz)
frame.pack()
```

```
Label(frame, text="elige tu destino", width=50).pack()
Checkbutton(frame, text="Brasil").pack()
Checkbutton(frame, text="Chile").pack()
Checkbutton(frame, text="Francia").pack()
Checkbutton(frame, text="España").pack()
raiz.mainloop()
```

Generar funcionalidad

Ejemplo similar a los radiobutton, que me muestre en pantalla qué destinos o destino elegí. Tenemos que generar varias variables del tipo entero, indicando que si está seleccionado vale 1 y sin seleccionar vale 0.

Creando variables

```
brasil=IntVar()
chile=IntVar()
francia=IntVar()
españa=IntVar()
```

Generamos la función para las opciones

```
def opciones():
    opcionElegida="" #Inicio vacío donde se almacenera el texto al final de la interfaz grafica
    if (brasil.get()==1):
        opcionElegida+="Brasil"
    if (chile.get()==1):
        opcionElegida+="Chile"
    if (francia.get()==1):
        opcionElegida+="Francia"
    if (españa.get()==1):
        opcionElegida+="España"
```

Actividad

Generar una etiqueta al final para mostrar la opción elegida, empaquetamos y hacemos que se configure textoFinal, dependiendo de la opción elegida al final de la función, pero dentro de la misma. Y realizar las llamadas a la función en cada check.

```
textofinal=Label(frame)
textofinal.pack()

textofinal.config(text=opcionElegida)# dentro de la función para retornar texto
Checkbutton(frame, text="Brasil", variable=brasil, onvalue=1, offvalue=0,
command=opciones).pack()
Checkbutton(frame, text="Chile", variable=chile, onvalue=1, offvalue=0,
command=opciones).pack()
Checkbutton(frame, text="Francia", variable=francia, onvalue=1, offvalue=0,
command=opciones).pack()
Checkbutton(frame, text="España", variable=españa, onvalue=1, offvalue=0,
command=opciones).pack()
```

Final del código

```
from tkinter import *
raiz= Tk()
raiz.title("Lugares Turísticos")

brasil=IntVar()
chile=IntVar()
francia=IntVar()
españa=IntVar()

def opciones():
        opcionElegida="" #Inicio vacío donde se almacenerá el texto al final de la interfaz grafica
```

```
        if (brasil.get()==1):
        opcionElegida+=" Brasil"

        if (chile.get()==1):
        opcionElegida+=" Chile"

        if (francia.get()==1):
        opcionElegida+=" Francia"

        if (españa.get()==1):
        opcionElegida+=" España"

        textofinal.config(text=opcionElegida)

foto=PhotoImage(file="ARG.png")
Label(raiz, image=foto).pack()

frame=Frame(raiz)
frame.pack()

Label(frame, text="elige tu destino", width=50).pack()

Checkbutton(frame, text="Brasil", variable=brasil, onvalue=1, offvalue=0,
command=opciones).pack()
Checkbutton(frame, text="Chile", variable=chile, onvalue=1, offvalue=0,
command=opciones).pack()
Checkbutton(frame, text="Francia", variable=francia, onvalue=1, offvalue=0,
command=opciones).pack()
```

```
Checkbutton(frame, text="España", variable=españa, onvalue=1, offvalue=0,
command=opciones).pack()
textofinal=Label(frame)
textofinal.pack()
raiz.mainloop()
```

Menú

Para crear se utiliza el widget Menú

Se genera la estructura principal o base:

```
from tkinter import *

raiz=Tk()

raiz.mainloop()
```

generamos una variable donde almacenaremos nuestro menú

```
barraMenu=Menu(raiz)
```

ahora usamos la raíz para que nos construya este menú

```
raiz.config(menu=barraMenu)
```

Ahora vamos a establecer cuántos elementos va a tener nuestro menú, archivo, edición, ayuda, herramientas, etc.

```
archivoMenu=Menu(barraMenu) # se crea el nombre y a que menú corresponde
EdicionMenu=Menu(barraMenu)
HerramientaMenu=Menu(barraMenu)
AyudaMenu=Menu(barraMenu)
```

Si ejecutamos el programa acá, no hace nada ya que hay que indicarle el texto que tendrán esos menús

barraMenu.add_cascade(label="Archivo", menu=archivoMenu)

A la barra menú, le agregamos el evento cascada, con la etiqueta e indicando a qué elemento le corresponde del menú.

Ahora, gestionamos todos los menús:

barraMenu.add_cascade(label="Archivo", menu=archivoMenu)

barraMenu.add_cascade(label="Edicion", menu=EdicionMenu)

barraMenu.add_cascade(label="Herramientas", menu=HerramientaMenu)

barraMenu.add_cascade(label="Ayuda", menu=AyudaMenu)

Si ahora ejecutamos el programa, vamos a ver los menús están un poco desfasados porque no asignamos tamaño a la raíz.

Al config de la raiz le agregamos:

raiz.config(menu=barraMenu, width=300, height=300)

Ahora vamos a los submenús

archivoMenu=Menu(barraMenu)

archivoMenu.add_command(label="Nuevo")

Actividad

Realizar los submenús del menú archivo con las etiquetas “Guardar, Guardar como, Cerrar y Salir”.,

Luego, las etiquetas del menú Edición con las etiquetas “Copiar, Cortar, Pegar, Deshacer, Rehacer”.

En el menú “Herramienta” las etiquetas “Ajuste de Línea, Fuente”

Y, por último, en “Ayuda”, “Licencia, Acerca de..”

El código hasta acá quedaría así:

```
from tkinter import *
```

```
raiz=Tk()

barraMenu=Menu(raiz)

raiz.config(menu=barraMenu, width=300, height=300)

archivoMenu=Menu(barraMenu)
archivoMenu.add_command(label="Nuevo")
archivoMenu.add_command(label="Guardar")
archivoMenu.add_command(label="Guardar como")
archivoMenu.add_command(label="Cerrar")
archivoMenu.add_command(label="Salir")

EdicionMenu=Menu(barraMenu)
EdicionMenu.add_command(label="Cortar")
EdicionMenu.add_command(label="Copiar")
EdicionMenu.add_command(label="Pegar")
EdicionMenu.add_command(label="Deshacer")
EdicionMenu.add_command(label="Rehacer")

HerramientaMenu=Menu(barraMenu)
HerramientaMenu.add_command(label="Ajuste de linea")
HerramientaMenu.add_command(label="Fuente")

AyudaMenu=Menu(barraMenu)
AyudaMenu.add_command(label="Licencia")
AyudaMenu.add_command(label="Acerca de...")
barraMenu.add_cascade(label="Archivo", menu=archivoMenu)
barraMenu.add_cascade(label="Edicion", menu=EdicionMenu)
```

```
barraMenu.add_cascade(label="Herramientas", menu=HerramientaMenu)
barraMenu.add_cascade(label="Ayuda", menu=AyudaMenu)

raiz.mainloop()
```

Sacamos la barra horizontal cuando se expanden los submenús, agregando un parámetro al elemento de Menú de cada uno de ellos:

```
archivoMenu=Menu(barraMenu, tearoff=0) # colocamos esa Lagrima como le dicen en 0 cero
EdicionMenu=Menu(barraMenu, tearoff=0)
HerramientaMenu=Menu(barraMenu, tearoff=0)
AyudaMenu=Menu(barraMenu, tearoff=0)
```

Agregamos líneas divisoras por grupos de mismas características

```
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")
```

El código quedaría así hasta ahora:

```
from tkinter import *

raiz=Tk()

barraMenu=Menu(raiz)

raiz.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")

EdicionMenu=Menu(barraMenu, tearoff=0)
EdicionMenu.add_command(label="Copiar")
EdicionMenu.add_command(label="Cortar")
EdicionMenu.add_command(label="Pegar")
EdicionMenu.add_separator()
EdicionMenu.add_command(label="Deshacer")
EdicionMenu.add_command(label="Rehacer")

HerramientaMenu=Menu(barraMenu, tearoff=0)
HerramientaMenu.add_command(label="Ajuste de linea")
HerramientaMenu.add_command(label="Fuente")

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

barraMenu.add_cascade(label="Archivo", menu=archivoMenu)
barraMenu.add_cascade(label="Edicion", menu=EdicionMenu)
barraMenu.add_cascade(label="Herramientas", menu=HerramientaMenu)
barraMenu.add_cascade(label="Ayuda", menu=AyudaMenu)

raiz.mainloop()
```

Ventana emergente

Son las ventanas que nos permiten informar, avisar o permitir realizar tareas al usuario.

No están incluidas en la librería Tkinter, por tal motivo debemos importar una nueva librería

```
from tkinter import messagebox
```

Vamos a construir la ventana emergente.

Luego de crear la raíz, vamos a definir una función que será la encargada de construir la ventana emergente:

```
def infoAdicional():
    messagebox.showinfo("Procesador de texto", "Version 2020 Python 3.x")
```

Dentro de la función, contiene dos parámetros: el primero corresponde al título de la ventana emergente y el segundo al texto que contiene la misma.

Llamamos a la función en la opción de Acerca de…

```
AyudaMenu.add_command(label="Acerca de...", command=infoAdicional)
```

Se pueden modificar las estructuras de las ventanas para que sean de alerta o aviso, información.

Ejemplo

En la licencia crearemos un aviso del estado de la licencia, debajo de la función anterior:

```
def avisoLicencia():
    messagebox.showwarning("Licencia", "Licencia Trial")
```

Llamamos a la función

```
AyudaMenu.add_command(label="Licencia", command=avisoLicencia)
```

Otro ejemplo para que nos consulte si queremos salir o no

Generamos la función

```
def salirAplicacion():
```

```
    messagebox.askquestion("Salir", "¿Deseas salir de al aplicacion?")
```

Llamamos a la función:

```
archivoMenu.add_command(label="Salir", command=salirAplicacion)
```

Acá generamos la acción, el método askquestion nos devuelve un valor uno por sí "YES" y otro por no "NO". Por este motivo debemos almacenar este valor en una variable para poder determinar distintas acciones según la elección:

```
def salirAplicacion():
    valor=messagebox.askquestion("Salir", "¿Deseas salir de al aplicación?")
    if valor =="yes":
    raiz.destroy()
```

Podríamos cambiar. askquestion por askokcancel y, de esta manera, veríamos un "aceptar" y "cancelar", pero no devuelve "yes" y "no", sino que devuelve "True" y "False"

También en "reintentar" o "cancelar" se puede modificar el evento mensaje.

```
def cerrarDocumento():
    valor=messagebox.askretrycancel("Reintentar", "No es posible cerrar. Documento bloqueado")
```

Llamamos a la función en "cerrar"

```
archivoMenu.add_command(label="Cerrar", command=cerrarDocumento)
```

Y ahora agregamos la función "si"

```
def cerrarDocumento():
    valor=messagebox.askretrycancel("Reintentar", "No es posible cerrar. Documento bloqueado")
    if valor == False:
    raiz.destroy()
```