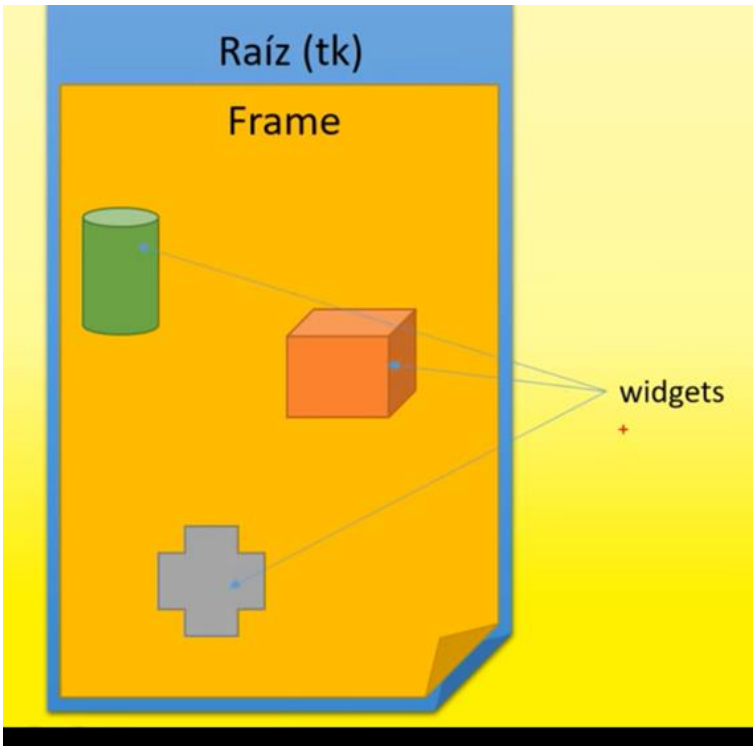
Material Imprimible

Curso de Aplicaciones Prácticas de Python



from tkinter import * #importar librería tkinter con todas las clases

raiz =Tk() #crear raíz llamando a la clase Tk

raiz.mainloop()#crear método para que la ventana este en estado de ejecución es necesario que esté en un bucle infinito

---

raiz.title("Trabajo Python")

raiz.config(bg="blue")#color de fondo

raiz.resizable(0,False)#ancho y alto 0 o 1 también

raiz.iconbitmap("logo.ico") # https://www.online-convert.com/es/

raiz.geometry("650x350") #ancho y alto predeterminado

En este paso ejecutamos desde el archivo directamente y veremos que abre la consola por detrás.
Para evitar eso cambiamos la extensión del archivo a .pyw

Interfaces gráficas de usuario con Tk

https://docs.python.org/3.3/library/tk.html

Opciones de empaquetador

Ancla / Anchor Maneja puntos cardinales Norte Sur Este Oeste

Tipo de anclaje. Indica dónde debe colocar cada esclavo en su parcela.

Ampliar / expand

Booleano, 0 o 1.

Llenar / fill

Valores jurídicos: 'x', 'y', 'both', 'none'.

ipadx y ipady

Una distancia - designando el relleno interno en cada lado del widget esclavo.

padx y pady

Una distancia - designando el relleno externo en cada lado del widget esclavo.

Lado / Side

Los valores legales son: 'izquierda', 'derecha', 'superior', 'inferior'.

```
miFrame=Frame() #crea el frame
miFrame.pack()#activa el método pack
miFrame.config(bg="red")

miFrame.config(width="650",height="350")#ancho y alto del frame
miFrame.pack(side="right")#lado derecho
miFrame.pack(fill="x")#expande en x
```

```
miFrame.pack(fill="y")#no expande en y
miFrame.pack(fill="y", expand="True")#expande en eje y
miFrame.pack(fill="both", expand="True") #Expande todo el frame
miFrame.config(relief="groove")#sunken borde de frame
miFrame.config(bd=35)#ancho de borde
miFrame.pack()#quitamos atributos así podemos observar bien el borde en el frame
miFrame.config(cursor="hand2")#cambio cursor pirate otro tipo
```

Agregamos bordes ancho y tipo de cursor a la Raiz

```
from tkinter import * #importar librería tkinter con todas las clases
raiz =Tk() #crear raíz llamando a la clase Tk
raiz.title("Aplicación de Escritorio")
raiz.resizable(1,True)#ancho y alto 0 o 1 también
raiz.iconbitmap("logo.ico")
#raiz.geometry("650x350") #ancho y alto predeterminado
raiz.config(bg="blue")#color de fondo
raiz.config(relief="groove")
raiz.config(bd=25)
raiz.config(cursor="pirate")

miFrame=Frame()
miFrame.pack()
miFrame.config(bg="red")
miFrame.config(width="650",height="350")
miFrame.config(relief="sunken")
miFrame.config(bd=35)
miFrame.config(cursor="hand2")

raiz.mainloop()#crear método para que la ventana este en estado de ejecución es necesario que
este en un bucle infinito
```

Label o etiquetas

Es un elemento estático donde no podemos interactuar (Borrar, arrastrar, etc)

Sintaxis

variableLabel=Label(contenedor, opciones)

Opción	Descripción
Text	Texto que se muestra en el Label
Anchor	Controla la posición del texto si hay espacio suficiente para él (center por defecto)
Bg	Color de fondo
Bitmap	Mapa de bits que se mostrará como gráfico
Bd	Grosor del borde (2 px por defecto)
Font	Tipo de fuente a mostrar
Fg	Color de la fuente
Width	Ancho de Label en caracteres (no en píxeles)
Height	Altura de Label en caracteres (no en píxeles)
Image	Muestra imagen en el Label en lugar de texto
justify	Justificación del texto del Label

```
from tkinter import *
raiz=Tk()
miFrame=Frame(raiz, width=500, height=400)
miFrame.pack()

raiz.mainloop()
despues del pack del frame
miLabel=Label(miFrame, text="Hola mundo") #generamos la variable contenedora del label
miLabel.pack()#empaquetamos el label dentro del frame, pero no respeta el ancho y el alto del frame
miLabel.place()#remplazamos pack por place para que tome el ancho y alto fijo, pero no muestra el texto
miLabel.place(x=100, y=200)#le damos coordenadas de ubicación en el frame
Label(miFrame, text="Hola mundo").place(x=100, y=200)#Si no tenemos, trabajamos con la etiqueta se puede abreviar de esta manera
```

Label(miFrame, text="Hola mundo", fg="red", font=("Comic Sans MS",18)).place(x=100, y=200)#caracteristicas de color de letra, tipo y tamaño.

Las imágenes que admite Tkinter son png y gif, si queremos otros formatos debemos importar otros módulos.

miImagen=PhotoImage(file="move.gif") # sin colocar la ruta cuando está dentro de la carpeta de trabajo.

Label(miFrame, image=miImagen).place(x=100, y=200) #cambiamos la propiedad text por la de image

Entry o entrada

Todas las opciones antes dadas como características de Label pueden utilizarse para los Entry

```
from tkinter import *
raiz=Tk()
raiz.mainloop()
cuadroTexto=Entry(raiz) #Variable con metodo Entry que le decimos que pertenece al elemento Raiz
cuadroTexto.pack()# empaquetamos pero sin dimensiones
```

Para generar dimensiones en la ventana

```
Creamos el frame
miFrame=Frame(raiz,width=1200, height=600)
dándole dimensiones
miFrame.pack()
lo empaquetamos
cuadroTexto=Entry(raiz)
cambiamos el contenedor raíz por el contenedor miFrame
cuadroTexto=Entry(miFrame)
y modificamos el empaquetado del cuadro de texto por place
cuadroTexto.place(x=100, y=100)
```

generando nuevas dimensiones en px dentro del frame

GENERANDO LABEL AL CUADRO DE TEXTO

nombreLabel=Label(miFrame,text="Nombre:")

nombreLabel.place(x=100, y=100) # no es aconsejable, la etiqueta empuja al cuadro, pero si le damos otra dimensión ejemplo x=120 vean qué hace…

Se podría trabajar con el método pack() al igual que los label, pero mejor sería trabajar con el método Grid() crea una tabla con filas y columnas como queramos dividimos nuestra interfaz con grillas

reemplazamos la sintaxis

cuadrtoTexto.place(x=100, y=100)

por

cuadroTexto.grid() indicando filas y columnas que tendrá nuestro frame

0 , 0	0 , 1	0 , 2
1 , 0	1 , 1	1 , 2
2 , 0	2 , 1	2 , 2

Row (Filas) Column (columnas)

```
from tkinter import *
raiz=Tk()
miFrame=Frame(raiz,width=1200, height=600)
miFrame.pack()
cuadroTexto=Entry(miFrame)
cuadroTexto.grid(row=0, column=1)
nombreLabel=Label(miFrame,text="Nombre:")
nombreLabel.grid(row=0, column=0)
raiz.mainloop()
```

pero el problema será que no respeta las dimensiones como pasó con los pack()

redimensionando manualmente notamos cómo la etiqueta está junto al cuadro de texto

Actividad

Generar tres label y con sus respectivos cuadros de texto indicando: Apellido, DNI y dirección.

Por defecto se pone en el centro al texto que creamos.

Para modificar esto usamos la funcion sticky utilizando coordenadas
N NE E SE S SW W NW

En los label agregar los parámetros
nombreLabel.grid(row=0, column=0, sticky="e") # posiciona a la derecha

A esto le podemos agregar el padding, también llamado pady, que es el relleno.

El modelo de caja CSS

Todos los elementos HTML se pueden considerar como cuadros. En CSS, el término "modelo de caja" se usa cuando se habla de diseño y diseño.
El modelo de cuadro CSS es esencialmente un cuadro que envuelve todos los elementos HTML. Consiste en: márgenes, bordes, relleno y el contenido real. La imagen a continuación ilustra el modelo de caja:
Explicación de las diferentes partes:

- **Contenido**: el contenido del cuadro, donde aparecen el texto y las imágenes.
- **Relleno**: borra un área alrededor del contenido. El relleno es transparente
- **Borde**: un borde que rodea el relleno y el contenido.
- **Margen**: borra un área fuera del borde. El margen es transparente

El modelo de caja nos permite agregar un borde alrededor de los elementos y definir el espacio entre los elementos.

El código se reflejaría de esta manera
nombreLabel.grid(row=0, column=0, sticky="e" , padx=50) #coloca el relleno a 50 de distancia en laterales
nombreLabel.grid(row=0, column=0, sticky="e" , padx=50, pady=50)

Actividad

Intente que el cuadro de texto esté alineado en el centro y de color rojo.

Utilice la propiedad config y justify

cuadroTexto.config(fg="red", justify="center")

---------- queda así hasta ahora

```
from tkinter import *

raiz =Tk()

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

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

cuadroApellido=Entry(miFrame)
cuadroApellido.grid(row=1, column=1, padx=10, pady=10)
cuadroApellido.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)

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

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

raiz.mainloop()

----------agregamos campo contraseña
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)
----------AGREGANDO CAMPO TEXT COMENTARIO PARA USUARIO------
```

Agregamos un label, respetamos la columna

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

Agregamos un cuadro text

```
textoComentario=Text(miFrame)
textoComentario.grid( row=4, column=1, padx=10, pady=10)
```

Nos quedará un cuadro de texto grande de tamaño porque, por defecto, es así de enorme
Para solucionarlo agregamos width y height

```
textoComentario=Text(miFrame, width=16, height=5)
```

Si escribimos bastante sigue y no agrega barra lateral.