PROGRAMACION 2

Interfaz Gráfica

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

Técnico Superior en Desarrollo de Software

Interfaz gráfica

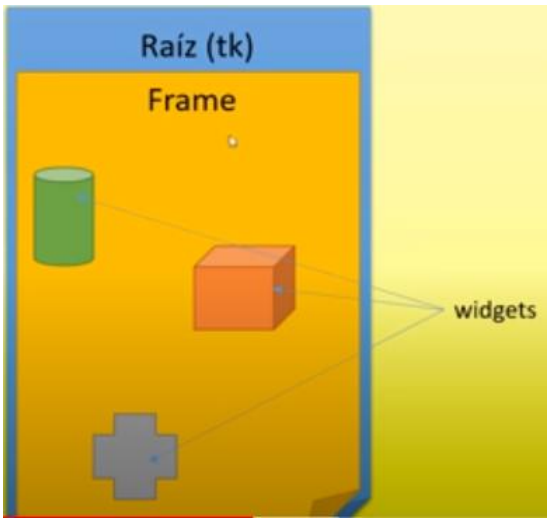
La interfaz gráfica de usuario, conocida también como GUI (graphical user interface) es fundamental cuando queremos implementar programas para usuarios finales que faciliten la entrada y salida de datos.

Librerias con las que trabajar para crear una GUI

- Tkinter
- WxPython
- PyQT
- PyGTK

Módulo Tkinter

Estructura de la Ventana de tkinter



Lo primero que vamos a crear es la estructura base o *raíz(tk)* La ventana marco. Y luego dentro de esa ventana puede haber varios elementos pero, es habitual que se construya un *Frame* como organizador o aglutinador de elementos.

Dentro de ese Frame podría haber diferentes elementos que se los denomina como ***widgets***, que pueden ser botones, casillas de verificación, menúes desplegables, etc. incluso un frame es considerado en Python como widgets

Creamos la ventana raiz

```
from tkinter import *

raiz=Tk()

raiz.mainloop()
```

Lo primero que tenemos que hacer es importar el módulo *tkinter*

La librería tkinter está codificada con la metodología de programación orientada a objetos. El módulo 'tkinter' tiene una clase llamada 'Tk' que representa una ventana. Debemos crear un objeto de dicha clase:

```
raiz=Tk()
```

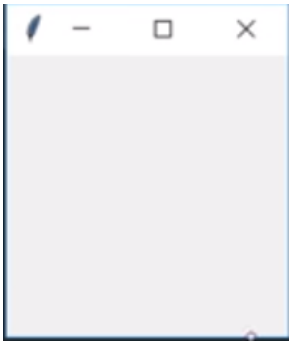
La opción más conveniente es `from tkinter import *` ya que si lo hacemos como `import tkinter as tk` tendremos que utilizar la nomenclatura del **.** y anteponer tk a todos los métodos por ejemplo

```
Raiz=tk.Tk()
```

Para que se muestre la ventana en el monitor debemos llamar por al método 'mainloop' que pertenece a la clase Tk:

```
raiz.mainloop()
```

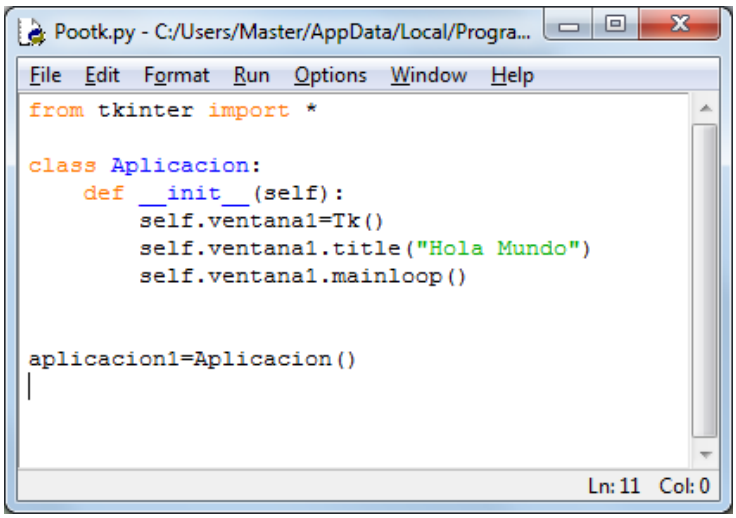
Ejecutamos y ya tenemos la ventana



Es importante que el método mainloop() esté siempre en último lugar

Aplicación orientada a objetos

Todos los ejemplos que haremos independientemente de su complejidad podemos utilizar la metodología de POO (Programación Orientada a Objetos)



Ahora, ¿ cómo podemos modificar las características que tiene esta ventana por defecto?

- Si queremos darle título utilizamos en método ***title*** y le pasamos un string con el mensaje que queremos que aparezca en la barra del título de la ventana:

  `raiz.title("mi ventana")`

- Permitir redimensionar o no la ventana

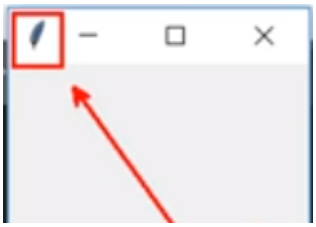
  `raiz.resizable(0,0)` `raiz.resizable(False,False)`

El método tiene dos valores booleanos, el primero el ancho(`width`) y el segundo el alto (`height`) 0 es igual a False 1 es igual a True. En este caso no nos permitirá redimensionar ni ancho, ni alto. Y estará desactivado el botón de maximizar. Por defecto están activados ambos.

- Si queremos cambiar el ícono que por defecto pone Python

Utilizamos el método ***iconbitmap*** y entre paréntesis y comillas escribimos la ruta donde está el archivo y el nombre del archivo .ico

`raiz.iconbitmap("../images/gato.ico")`



- Definir el tamaño de la ventana. Utilizamos el método ***geometry*** e indicamos el tamaño que queremos

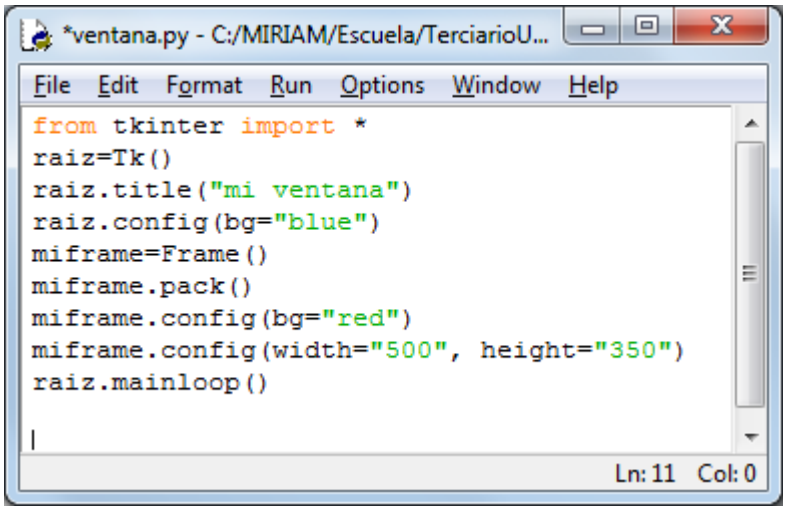
  `raiz.geometry("500x350")`

- Configurar apariencia de la ventana. Vamos a utilizar el método ***config*** con el cual podemos cambiar por ejemplo:

  `raiz.config(bg="blue")` **bg** indica **background** es decir el color de fondo de la ventana

- Si ejecutamos estas líneas va a abrir la ventana y también la consola de Python detrás. Para que nos abra directamente la ventana sin la consola atrás debemos cambiar al archivo la extensión de **py** a **pyw** y a partir de ese momento abrirá directamente la ventana.

Creando el Frame



`miframe=Frame()` creamos una instancia de la clase Frame

Con esto creamos el frame pero debemos empaquetarlo es decir meterlo en la ventana raiz

Para ello usamos `miframe.pack()`

Opciones del empaquetador

- `Miframe.pack(side="right")` le estamos dando la posición en el marco. Aparece anclado a la derecha. Los otros valores que tenemos son left, botton, top.
- Otra opción para ubicar el frame es **`anchor`** utiliza como parámetros los puntos cardinales **n**(north) **s**(south) **e**(east) **w**(west). Podemos combinar la opción side con la anchor

  `Miframe.pack(side="right", anchor="n")`
- Si queremos que al redimensionar la raiz se adapte el marco podemos utilizar la opción **`fill`** que tiene los siguientes parámetro: `x, y, both, none` combinado con la opción **`expand`**

  `Miframe.pack(fill="y", expand="True")`

Anchor Tipo de anclaje. Indica dónde debe colocar el empaquetador a cada esclavo en su espacio.

Expand Un valor booleano, `0` o `1`.

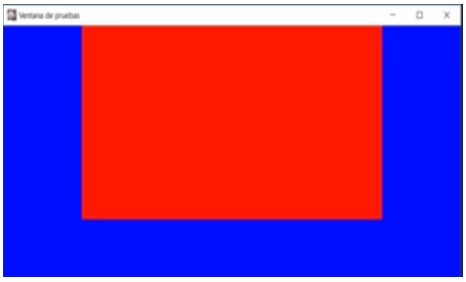
Fill Los valores legales son: `'x'`, `'y'`, `'both'`, `'none'`.

ipadx y ipady Una distancia que designa el relleno interno a cada lado del widget esclavo.

padx y pady Una distancia que designa el relleno externo a cada lado del widget esclavo.

Side Los valores legales son: `'left'`, `'right'`, `'top'`, `'bottom'`.

Configuración de Opciones



Usando el método ***config()*** para actualizar múltiples atributos después de la creación del objeto

```
miframe.config(bg=”red”)
```

```
miframe.config(fg="red", bg="blue")
```

Cambiamos el color de fondo (bg backgroung fg foregroung) del frame

Para colores también podemos utilizar rgb

Para que el frame se vea debemos darle tamaño y se lo quitaremos a la ventana raiz la cuál siempre va a adaptar su tamaño al del frame

```
miframe.config(width=”500”, height=”350”)
```

Por defecto el Frame tiene tamaño fijo no se adapta al tamaño de la raiz y aparece anclado a la parte superior.

Todo esto lo podemos cambiar, para ver las diferentes opciones podemos ir al siguiente link donde está la documentación

https://docs.python.org/3/library/tkinter.html

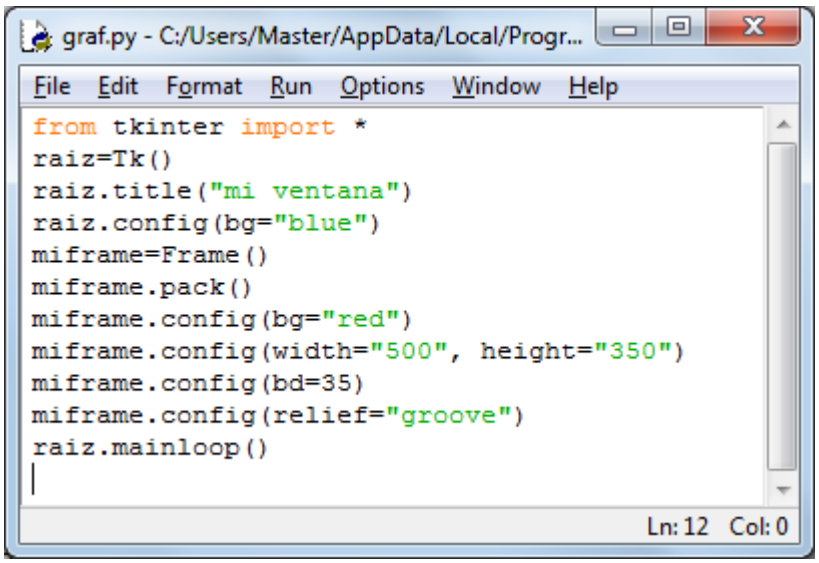
- Para cambiar el ancho del borde

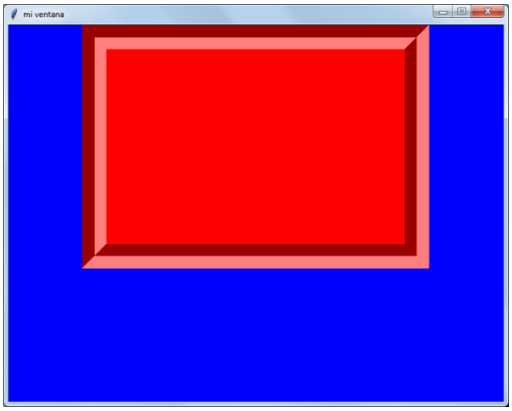
  `Miframe.config(bd=35)` bd borde y ancho px del borde

- Para cambiar el tipo de borde utilizamos la opción `relief.` Se debe combinar con el ancho del borde ya que por defecto es 0 su ancho.Por ejemplo groove, sunken

  ```
  Miframe.config(relief=”groove”)
  ```

Para la documentación sobre otros tipos de bordes pueden consultar la página mencionada anteriormente





- También podemos cambiar el cursor que aparecerá

`Miframe.config(cursor=”hand2”)` aparecerá una mano existen varias opciones para cursores por ejemplo pirate tiene una forma de calavera

Todo lo que hemos aplicado el frame también se lo podemos aplicar a la ventana raiz pero para la ventana raiz existe un método que no puede aplicarse al frame

`raiz.maxsize("700","500")` será el tamaño máximo que puede tomar la ventana raiz

Label

Los label son widgets que nos permiten mostrar textos o imágenes. Tienen como única finalidad mostrar no se puede interactuar con ellos.

```
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

Vamos a diseñar una ventana con un frame que tenga un tamaño de 500x400 que tenga un label que contenga el texto Hola bienvenidos

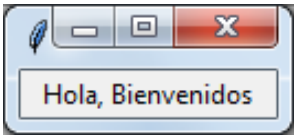
```
from tkinter import *
raiz=Tk()

miframe=Frame(raiz,width=500,height=400)
miframe.pack()

milabel=Label(miframe,text="Hola, Bienvenidos")
milabel.pack()

raiz.mainloop()
```

Al ejecutarlo obtenemos lo siguiente:



Por qué a pesar que le dimos un tamaño de 500x400 no lo mostró así?

Método Place

El pack con el Label adapta el contenedor a las dimensiones del Label, si queremos que respete el tamaño y además ubique el Label en el frame debemos sustituir el pack() por un método diferente que es el **place()**

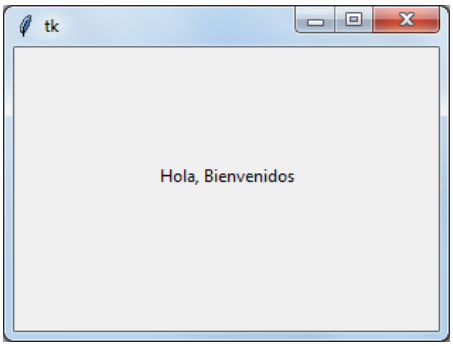
El método **place** nos va a permitir ubicar el texto donde queramos mediante coordenadas x e y

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

miframe=Frame(raiz,width=300,height=200)
miframe.pack()

milabel=Label(miframe,text="Hola, Bienvenidos")
milabel.place(x=100, y=80)

raiz.mainloop()
```



Veamos otro ejemplo

```
File  Edit  Format  Run  Options  Window  Help
from tkinter import *
raiz=Tk()
raiz.title("Aprendiendo Tkinter")
raiz.resizable(True,True)
raiz.iconbitmap("palmera.ico")
raiz.geometry("500x350")
raiz.config(bg="grey")
miframe=Frame(raiz,width=300,height=300)
miframe.pack()
miimagen=PhotoImage(file="pip.png")
milabel=Label(miframe,image=miimagen)
milabel.place(x=60, y=80)
raiz.mainloop()
```

Otra forma de asignar colores