



# PYTHON

GUI Programming

Lexicon of Widgets and more Tkinter Essentials

1

## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Todos los constructores de los widgets reciben como primer argumento el *master widget* que puede ser una ventana (Tk), un **Frame** o un **LabelFrame**.
- El resto de los argumentos dependerá del widget.
- Los widgets se dividen en clicables y no clicables.
  - Clicables:
    - Button
    - Checkbutton
    - Radiobutton
  - No clicables:
    - Label
    - Message (similar a Label, pero con ajustes automático, múltiples líneas, ...)
    - Frame
    - LabelFrame
    - Entry
    - Menu

## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

### ■ Button:

Button property	Property meaning
<code>command</code>	the <b>callback</b> being invoked when the button is clicked
<code>justify</code>	the way in which the inner text is justified: possible (self-describing) values are: <code>LEFT</code> , <code>CENTER</code> , and <code>RIGHT</code>
<code>state</code>	if you set the property to <code>DISABLED</code> , the button becomes deaf and doesn't react to clicks, while its title is shown in gray; setting it to <code>NORMAL</code> restores normal button functioning; when the mouse is located above the button, the property changes its value to <code>ACTIVE</code>

Button method	Method role
<code>flash()</code>	the button <b>flashes</b> a few times but doesn't change its state
<code>invoke()</code>	activates the <b>callback</b> assigned to the widget and returns the same value the callback returned; note: this is the <b>only way</b> to invoke your own callback explicitly, as the event manager must be aware of the fact

## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- CheckButton:

Checkbutton property	Property meaning
<code>bd</code>	the <code>checkbutton</code> frame <b>width</b> (default is two pixels)
<code>command</code>	the <b>callback</b> being invoked when the <code>checkbutton</code> changes its state
<code>justify</code>	the same as for <code>Button</code>
<code>state</code>	the same as for <code>Button</code>
<code>variable</code>	an observable <code>IntVar</code> variable reflecting the widget's state; defaultly it's set to <code>1</code> when the <code>checkbutton</code> is checked, and to <code>0</code> otherwise
<code>offvalue</code>	the non-default value being assigned to a <code>variable</code> when the <code>checkbutton</code> is not checked
<code>onvalue</code>	the non-default value being assigned to a <code>variable</code> when the <code>checkbutton</code> is checked

## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- CheckButton:

Checkbutton method	Method role
<code>deselect()</code>	<b>unchecks</b> the widget
<code>flash()</code>	the same as for <code>Button</code>
<code>invoke()</code>	the same as for <code>Button</code>
<code>select()</code>	<b>checks</b> the widget
<code>toggle()</code>	<b>toggles</b> the widget (changes its state to the opposite one)

## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Radiobutton:

Radiobutton property	Property meaning
<code>command</code>	the <b>callback</b> being invoked when the <code>Radiobutton</code> (not the group it belongs to!) changes its state
<code>justify</code>	the same as for <code>Button</code>
<code>state</code>	the same as for <code>Button</code>
<code>variable</code>	an observable <code>IntVar</code> or <code>StringVar</code> variable reflecting the <b>current</b> selection within the <code>Radiobutton</code> 's group; changing the variable's value automatically <code>changes</code> the selection
<code>value</code>	a unique (inside the group) value <b>identifying</b> the <code>Radiobutton</code> ; can be an integer value or a string, and should be compatible with the variable's type

## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Radiobutton:

Radiobutton method	Method role
<code>deselect()</code>	<b>unchecks</b> the widget
<code>flash()</code>	the same as for <code>Button</code>
<code>invoke()</code>	the same as for <code>Button</code>
<code>select()</code>	<b>checks</b> the widget

## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Label y Message:

Label property	Property meaning
<code>text</code>	a string which will be <b>shown</b> within the <code>Label</code> ; note: newline characters ( <code>\n</code> ) are interpreted in the <code>usual way</code>
<code>textvariable</code>	the same as for <code>text</code> , but makes use of an observable <code>StringVar</code> variable, so if you change the variable's alteration, it will be immediately visible on the screen.



## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

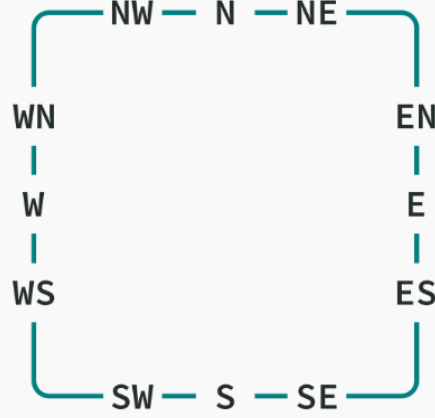
- Frame:
  - Es un contenedor de widgets (pero no puede ser el principal).
  - Tiene su propio sistema de coordenadas.
  - Tiene su propio esquema de distribución de componentes (gestores de geometrías).
  - Puede contener cualquier otro widget, incluido otros Frames.
  - `frm = Frame(master, option, ...)`

Frame property	Property meaning
<code>takefocus</code>	normally, the <code>Frame</code> <b>doesn't take the focus</b> (which would seem to be obvious) but if you really want it to behave in this way, you can set the property to <code>1</code> .

## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- **LabelFrame:**
  - Es un Frame enriquecido con un borde y un título.
  - El título se puede ubicar en cualquiera de las 12 ubicaciones que proporciona.
  - `lfrm = LabelFrame(master, option, ...)`

LabelFrame property	Property meaning
<code>takefocus</code>	the same as for the <code>Frame</code>
<code>text</code>	the <code>LabelFrame</code> 's title
<code>labelanchor</code>	the <b>title's location</b> , defined as a string containing a quasi-compass coordinate (as shown by the image)



## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Entry:
  - Permite introducir texto.

Entry property	Property meaning
<code>command</code>	although <code>Entry</code> is obviously a clickable widget, it doesn't allow you to bind a callback through the <code>command</code> property. You can observe and control all occurring changes instead by setting the <b>tracer function</b> for the observable variable which cooperates with <code>Entry</code> (we'll show you this – be patient!)
<code>show</code>	a string assigned to this property will be displayed instead of the <b>actual</b> characters entered into the input field; e.g., if you set <code>show='*'</code> , this will enable the widget to safely edit the user's password
<code>state</code>	the same as for <code>Button</code>
<code>textvariable</code>	an observable <code>StringVar</code> reflecting the current state of the input field
<code>width</code>	the input field's width (in characters)

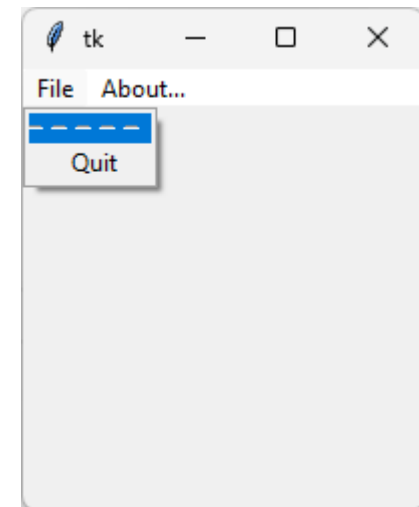
## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Entry:
  - Permite introducir texto.

Entry method	Method role
<code>get()</code>	returns the current input field's contents as a string
<code>set(s)</code>	sets the whole input field's contents with the <code>s</code> string
<code>delete(first, last=None)</code>	deletes a part of the input field's contents; <code>first</code> and <code>last</code> can be integers with values indexing the string; if the <code>last</code> argument is omitted, a single character is deleted; if <code>last</code> is specified as <code>END</code> , it points to the place after the last field's character
<code>insert(index, s)</code>	inserts the <code>s</code> string at the field position pointed to by <code>index</code>

## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Menu:
  - Widget Menu → Válido tanto para el menú principal como para los submenús.
  - Se agrega a la ventana principal a través del método **config** y el parametro **menu**:
    - `window.config(menu=main_menu)`
  - Métodos para agregar elementos:
    - `add_cascade` → Agrega submenús.
    - `add_command` → Agregar opciones con acción asociada.
    - El atributo `underline=numero` subraya la letra que está en la posición número y la convierte en atajo Alt+letra. La línea aparece al pulsar la tecla ALT.
    - El atributo **tearoff=0** elimina las líneas punteadas de los menús.
    - El método **add\_separator** permite añadir una línea de separación.



## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Menu:
  - Se puede añadir un acceso directo a una opción de acción (command):
    - Agregar el atributo **accelerator**:
      - `sub_menu_file.add_command(label="Quit", accelerator="Ctrl-Q", underline=0, command=are_you_sure)`
    - Haciendo un bind a la Ventana que relacione la combinación de teclas con la función callback.
      - `window.bind_all("<Control-q>", are_you_sure)`
  - Los items del menu **no se pueden modificar con config** → Utilizar el método **entryconfigure** propio de los menús. Se utiliza de la siguiente manera:
    - `nombre_menu.entryconfigure(índice_opción_menú, atributo=valor)`

## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Menu:

Property	Property role
<code>postcommand</code>	a <b>callback</b> invoked every time a menu's item is activated
<code>tearoff</code>	set to zero <b>removes</b> the tear-off decoration from the top of the cascade
<code>state</code>	when set to <code>DISABLED</code> , the menu item is grayed and inaccessible; setting it to <code>ACTIVE</code> restores its normal functionality
<code>accelerator</code>	a <b>string</b> describing a hot-key bound to the menu's item

Method	Method role
<code>add_cascade(prop=val, ...)</code>	adds a <b>cascade</b> to the menu's item
<code>add_command(prop=val, ...)</code>	assigns an <b>action</b> to the menu's item
<code>add_separator()</code>	adds an <b>separator</b> line to the menu
<code>entryconfigure(i, prop=val, ...)</code>	modifies the <i>i</i> -th menu item's property named <code>prop</code>

## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Ventana (clase Tk):
  - Método **title()** → Permite cambiar el título.
  - **Cambio de icono con tk.call (solución EDUBE):**

```
import tkinter as tk

window = tk.Tk()
window.title('Icon?')
icono = tk.PhotoImage(file="logo.png")
window.tk.call('wm', 'iconphoto', window._w, icono)
window.mainloop()
```

- **Cambio de icono con iconphoto o iconbitmap:**

```
import tkinter as tk

window = tk.Tk()
window.title("Mi Aplicación")
icono = tk.PhotoImage(file="logo.png")
window.iconphoto(True, icono)
window.mainloop()
```

```
import tkinter as tk

window = tk.Tk()
window.title("Mi Aplicación")
window.iconbitmap("icono.ico")
window.mainloop()
```



## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Ventana (clase Tk):
  - Método **geometry()** → Permite cambiar el tamaño de una ventana. → `window.geometry("500x500")`
  - Método **minsize()** → Permite indicar el tamaño mínimo de una ventana → `window.minsize(width=500, height=300)`
  - Método **maxsize()** → Permite indicar el tamaño máximo de una ventana → `window.maxsize(width=500, height=300)`
  - Método **resizable()** → Bloque el cambio de tamaño → `window.resizable(width=False, height=False)`
  - Método **protocol()** → Permite asociar una operación de la Ventana con una función callback → `window.protocol("WM_DELETE_WINDOW", function_callback)`
    - "WM\_DELETE\_WINDOW" → Detecta el cierre de la ventana. (Más usado y compatible)
    - "WM\_TAKE\_FOCUS" → Detecta cuando la ventana recibe el foco. Puede no funcionar en Windows.
    - "WM\_SAVE\_YOURSELF" → En entornos X11, se usa para guardar el estado antes de cerrar.

## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- **messagebox:**
  - **Parámetros:**
    - **title:** texto de la barra de título del diálogo
    - **message:** texto del diálogo. El símbolo \n permite dividir las líneas.
    - **options:** opciones de configuración del diálogo.
      - **default:** establece la respuesta predeterminada (predefinida); normalmente, se centra en el botón situado primero desde la izquierda; Esto se puede cambiar configurando el argumento de palabra clave con identificadores como CANDEL, IGNORE, OK, NO, RETRY y YES
      - **icon:** establece el icono no predeterminado para el diálogo: los valores posibles son: ERROR, INFO, QUESTION y WARNING

## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- **messagebox:**

- **Funciones:**

- **askyesno():** respuestas de usuario sí/no (no permite cerrar). → Icono de interrogación → Devuelve True o False
    - **askokcancel():** respuestas de usuario sí/cancelar (permite cerrar) → Icono de interrogación → Devuelve True o False.
    - **askretrycancel():** respuestas de usuario reintentar/cancelar (permite cerrar) → Icono de aviso → Devuelve True o False
    - **askquestion():** respuestas de usuario sí/no (no permite cerrar) → Icono de interrogación → Devuelve las cadenas “yes” o “no”.
    - **showerror():** muestra un mensaje de error (permite cerrar) y un botón para aceptar → Icono de error
    - **showwarning():** muestra un mensaje de advertencia (permite cerrar) y un botón para aceptar → Icono de aviso

## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Clase Canvas:
  - Representa una superficie rectangular sobre la que se puede dibujar, escribir textos o representar widgets.
  - Instanciación:
    - `c = Canvas(master, options...)`
  - Propiedades:

Property name	Property role
<code>borderwidth</code>	canvas border's <b>width</b> in pixels (default: 2)
<code>background</code> (bg)	canvas border's <b>color</b> (default: the same as the underlying window's color)
<code>height</code>	canvas <b>height</b> (in pixels)
<code>width</code>	canvas <b>width</b> (in pixels)

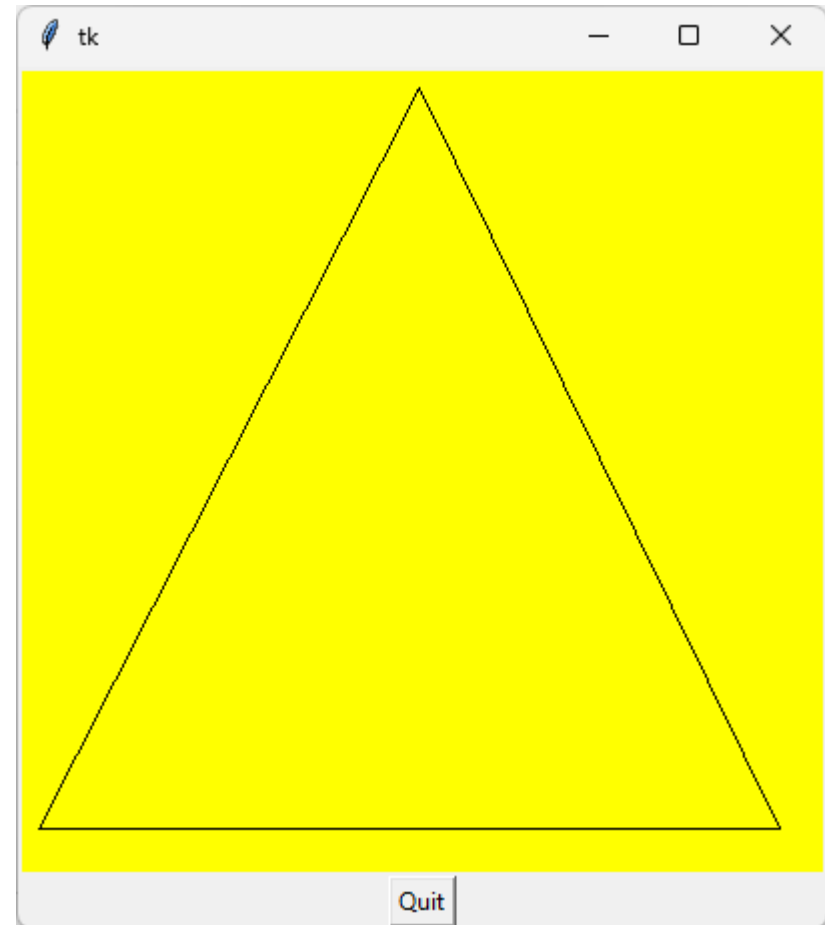
## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Clase Canvas:

- Ejemplo:

```
import tkinter as tk
```

```
window = tk.Tk()
canvas = tk.Canvas(window, width=400, height=400, bg='yellow')
canvas.create_line(10, 380, 200, 10, 380, 380, 10, 380)
button = tk.Button(window, text="Quit", command=window.destroy)
canvas.grid(row=0)
button.grid(row=1)
window.mainloop()
```



## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Clase Canvas:
  - Método **create\_line**: `canvas.create_line(x0, y0, x1, y1, ..., xn, yn, option...)`

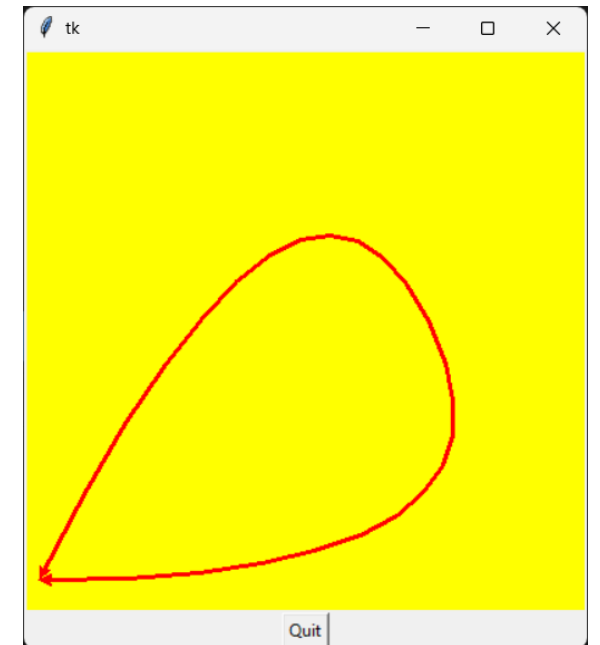
Option name	Option meaning
<code>arrow</code>	normally, the chain ends aren't marked in any special way, but you may want them to be finished with <b>arrowheads</b> ; setting the arrow option to <code>FIRST</code> results in drawing an arrowhead at the chain's beginning, <code>LAST</code> at the chain's end, <code>BOTH</code> at both sides of the chain.
<code>fill</code>	chain <b>color</b> (setting the option to an empty string causes the line to be transparent)
<code>smooth</code>	setting it to <code>True</code> rounds the chain's corners using a set of connected parabolas
<code>width</code>	line <b>width</b> (default: 1 pixel)

## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Clase Canvas:
  - Método **create\_line**: ejemplo

```
import tkinter as tk

window = tk.Tk()
canvas = tk.Canvas(window, width=400, height=400, bg='yellow')
canvas.create_line(10, 380, 200, 10, 380, 380, 10, 380,
                  arrow=tk.BOTH, fill='red', smooth=True, width=3)
button = tk.Button(window, text="Quit", command=window.destroy)
canvas.grid(row=0)
button.grid(row=1)
window.mainloop()
```



## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Clase Canvas:
  - Método **create\_rectangle**: `canvas.create_rectangle(x0, y0, x1, y1, option...)`

Option name	Option meaning
<code>outline</code>	rectangle <b>edge color</b> (if specified as an empty string, the edge is transparent)
<code>fill</code>	rectangle <b>interior color</b>
<code>width</code>	rectangle <b>edge width</b> in pixels (default: 1)

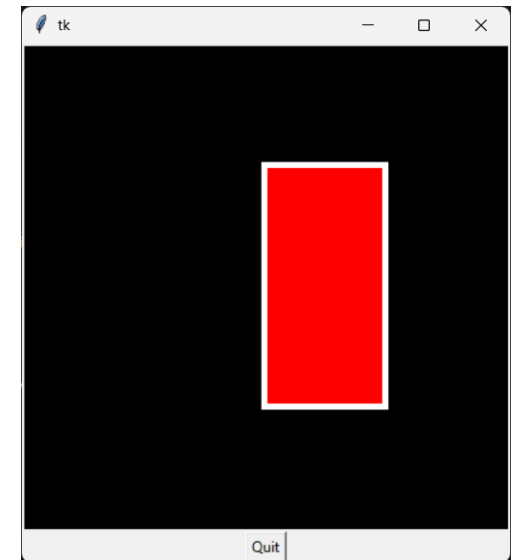


## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Clase Canvas:
  - Método **create\_rectangle**: ejemplo

```
import tkinter as tk

window = tk.Tk()
canvas = tk.Canvas(window, width=400, height=400,
bg='black')
canvas.create_rectangle(200, 100, 300, 300,
outline='white', width=5, fill='red')
button = tk.Button(window, text="Quit",
command=window.destroy)
canvas.grid(row=0)
button.grid(row=1)
window.mainloop()
```

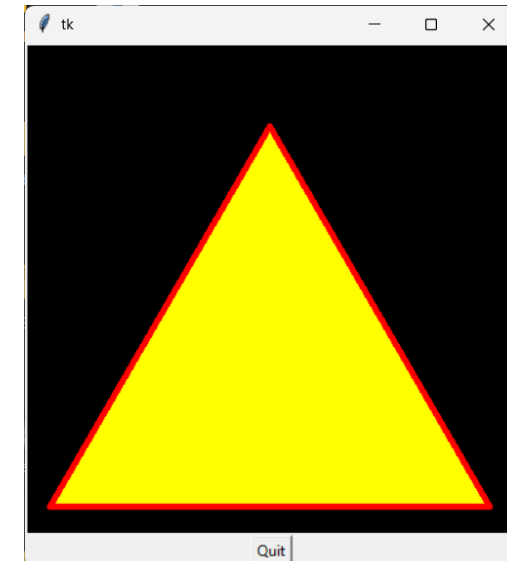


## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Clase Canvas:
  - Método **create\_polygon**: `canvas.create_polygon(x0, y0, x1, y1, xn, yn, option...)`

```
import tkinter as tk

window = tk.Tk()
canvas = tk.Canvas(window, width=400, height=400, bg='black')
canvas.create_polygon(20, 380, 200, 68, 380, 380, outline='red',
width=5, fill='yellow')
button = tk.Button(window, text="Quit", command=window.destroy)
canvas.grid(row=0)
button.grid(row=1)
window.mainloop()
```

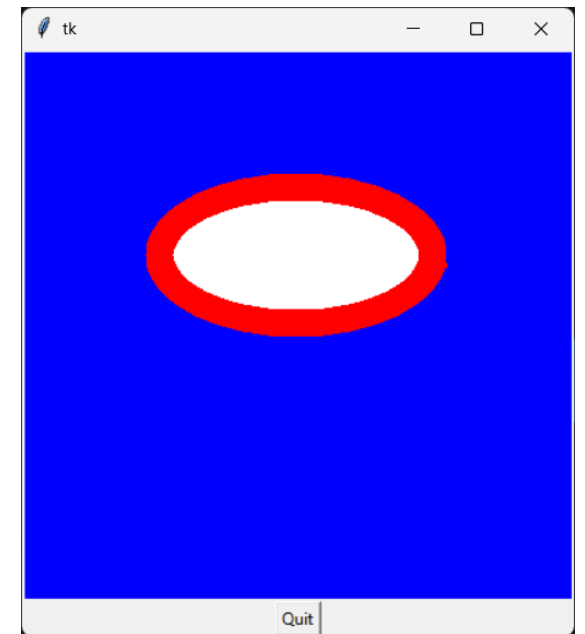


## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Clase Canvas:
  - Método **create\_oval**: `canvas.create_oval(x0,y0,x1,y1,xn,yn,option...)`




```
import tkinter as tk

window = tk.Tk()
canvas = tk.Canvas(window, width=400, height=400, bg='blue')
canvas.create_oval(100, 100, 300, 200, outline='red',
width=20, fill='white')
button = tk.Button(window, text="Quit",
command=window.destroy)
canvas.grid(row=0)
button.grid(row=1)
window.mainloop()
```



## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Clase Canvas:
  - Método **create\_arc**: `canvas.create_arc(x0,y0,x1,y1,option...)`

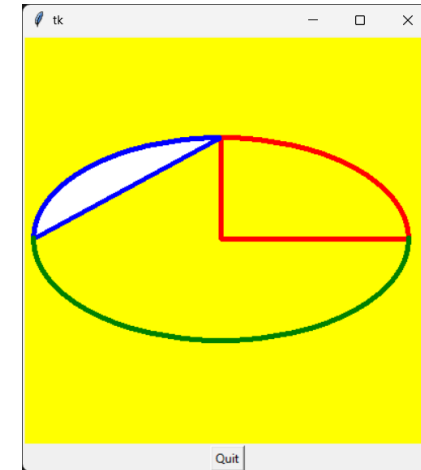
Option name	Option meaning
<code>style</code>	<p>can be set to one of the following: <code>PIESLICE</code> (default), <code>CHORD</code> and <code>ARC</code>; the <b>shape</b> of the resulting drawing is presented here:</p> <div></div> <p>Pieslice                      Chord                      Arc</p>
<code>start</code>	<p>the <b>angle</b> (in degrees) of the arc's start relative to the X-axis (e.g., 90 means the highest point of the ellipse, while 0 is the right-most point. The default is 0)</p>
<code>extent</code>	<p>the arc's <b>span</b> (in degrees) relative to the start point; note: the span is calculated counter-clockwise. The default is 90 (a quarter of an ellipse)</p>

## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Clase Canvas:
  - Método **create\_arc**: `canvas.create_arc(x0,y0,x1,y1,option...)`

```
import tkinter as tk

window = tk.Tk()
canvas = tk.Canvas(window, width=400, height=400, bg='yellow')
canvas.create_arc(10, 100, 380, 300, outline='red', width=5)
canvas.create_arc(10, 100, 380, 300, outline='blue', width=5,
                  style=tk.CHORD, start=90, fill='white')
canvas.create_arc(10, 100, 380, 300, outline='green', width=5,
                  style=tk.ARC, start=180, extent=180)
button = tk.Button(window, text="Quit", command=window.destroy)
canvas.grid(row=0)
button.grid(row=1)
window.mainloop()
```



## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Clase Canvas:
  - Método **create\_text**: `c.create_text(x, y, option...)`

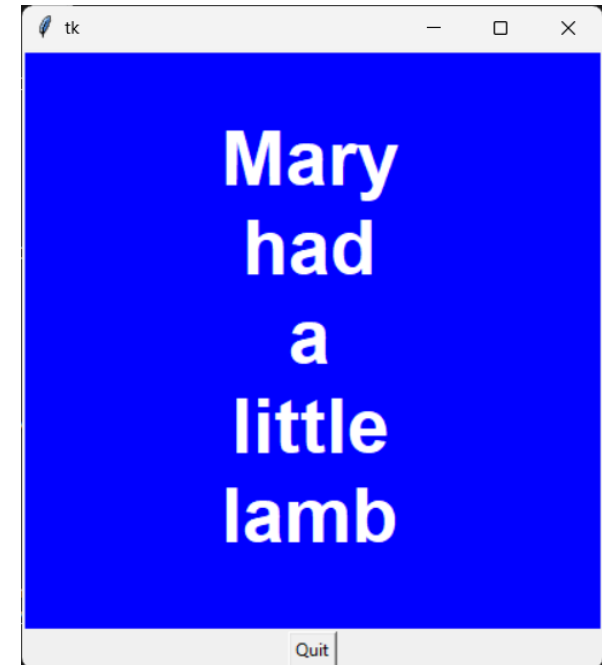
Option name	Option meaning
<code>fill</code>	text <b>color</b>
<code>font</code>	text <b>font</b>
<code>justify</code>	text <b>justification</b> : <code>LEFT</code> (default), <code>CENTER</code> , <code>RIGHT</code>
<code>text</code>	<b>text</b> to display ( <code>\n</code> works as expected)
<code>width</code>	normally, the rectangle is as wide as the <b>longest text line</b> ; using the width option forces the text to be aligned to that size

## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Clase Canvas:
  - Método **create\_text**: ejemplo

```
import tkinter as tk

window = tk.Tk()
canvas = tk.Canvas(window, width=400, height=400, bg='blue')
canvas.create_text(200, 200, text="Mary\nhad\na\nlittle\nlamb",
                  font=("Arial", "40", "bold"),
                  justify=tk.CENTER,
                  fill='white')
button = tk.Button(window, text="Quit", command=window.destroy)
canvas.grid(row=0)
button.grid(row=1)
window.mainloop()
```



## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Clase Canvas:
  - Método **create\_image**: `canvas.create_image(x, y, option...)`

Option name	Option meaning
<code>image</code>	an object of the <code>PhotoImage</code> class containing the image itself; the <code>PhotoImage</code> class constructor needs a keyword argument named <code>file</code> pointing to a <b>bitmap file</b> (note: only GIF and PNG formats are accepted); the argument should specify the file's path

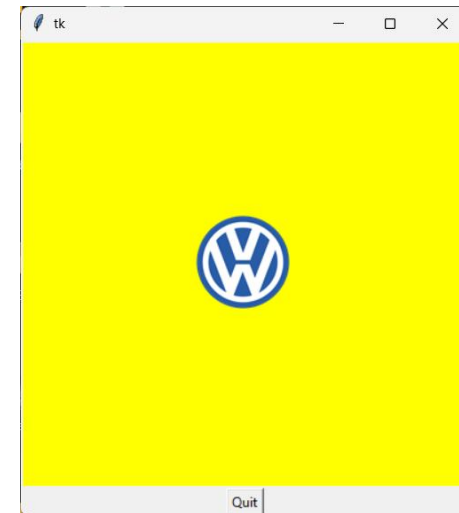


## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Clase Canvas:
  - Método **create\_image**: ejemplo

```
import tkinter as tk

window = tk.Tk()
canvas = tk.Canvas(window, width=400, height=400,
bg='yellow')
image = tk.PhotoImage(file='logo.png')
canvas.create_image(200, 200, image=image)
button = tk.Button(window, text="Quit",
command=window.destroy)
canvas.grid(row=0)
button.grid(row=1)
window.mainloop()
```



## 02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Clase Canvas:
  - Método **create\_image**. Ficheros JPG.
  - Se necesita utilizar el módulo PIL (Pillow)

```
import tkinter as tk
from PIL import Image as PILImage, ImageTk as PILImageTk

window = tk.Tk()
canvas = tk.Canvas(window, width=400, height=400, bg='red')
jpg = PILImage.open('logo.jpg')
image = PILImageTk.PhotoImage(jpg)
canvas.create_image(200, 200, image=image)
button = tk.Button(window, text="Quit",
command=window.destroy)
canvas.grid(row=0)
button.grid(row=1)
window.mainloop()
```

