PYTHON

GUI Programming

Lexicon of Widgets and more Tkinter Essentials

02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

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

Button method	Method role
`flash()`	the button **flashes** a few times but doesn't change its state
`invoke()`	activates the **callback** assigned to the widget and returns the same value the callback returned; note: this is the **only way** 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
`bd`	the `checkbutton` frame **width** (default is two pixels)
`command`	the **callback** being invoked when the `checkbutton` changes its state
`justify`	the same as for `Button`
`state`	the same as for `Button`
`variable`	an observable `IntVar` variable reflecting the widget's state; defaultly it's set to `1` when the `checkbutton` is checked, and to `0` otherwise
`offvalue`	the non-default value being assigned to a `variable` when the `checkbutton` is not checked
`onvalue`	the non-default value being assigned to a `variable` when the `checkbutton` is checked

02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- CheckButton:

Checkbutton method	Method role
`deselect()`	**unchecks** the widget
`flash()`	the same as for `Button`
`invoke()`	the same as for `Button`
`select()`	**checks** the widget
`toggle()`	**toggles** the widget (changes its state to the opposite one)

02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Radiobutton:

Radiobutton property	Property meaning
`command`	the **callback** being invoked when the `Radiobutton` (not the group it belongs to!) changes its state
`justify`	the same as for `Button`
`state`	the same as for `Button`
`variable`	an observable `IntVar` or `StringVar` variable reflecting the **current** selection within the `Radiobutton`'s group; changing the variable's value automatically `changes` the selection
`value`	a unique (inside the group) value **identifying** the `Radiobutton`; 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
`deselect()`	**unchecks** the widget
`flash()`	the same as for `Button`
`invoke()`	the same as for `Button`
`select()`	**checks** the widget

02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Label y Message:

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

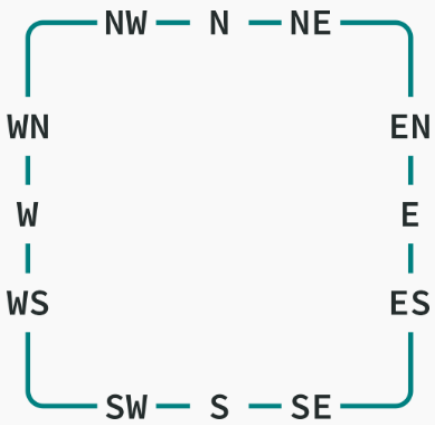
02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- 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
`takefocus`	normally, the `Frame` **doesn't take the focus** (which would seem to be obvious) but if you really want it to behave in this way, you can set the property to `1`.

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

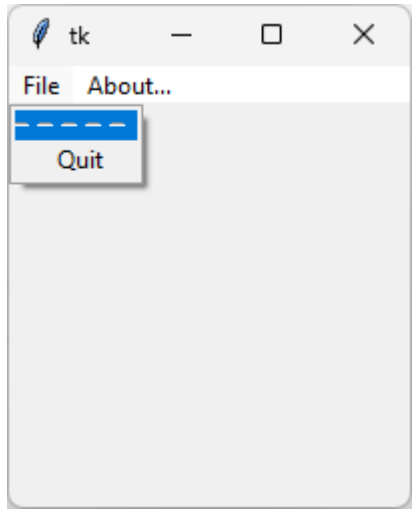
02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Entry:
  - Permite introducir texto.

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

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

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

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 mínimo 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 widgtes.
  - Instanciación:
    - `c = Canvas(master, options...)`
  - Propiedades:

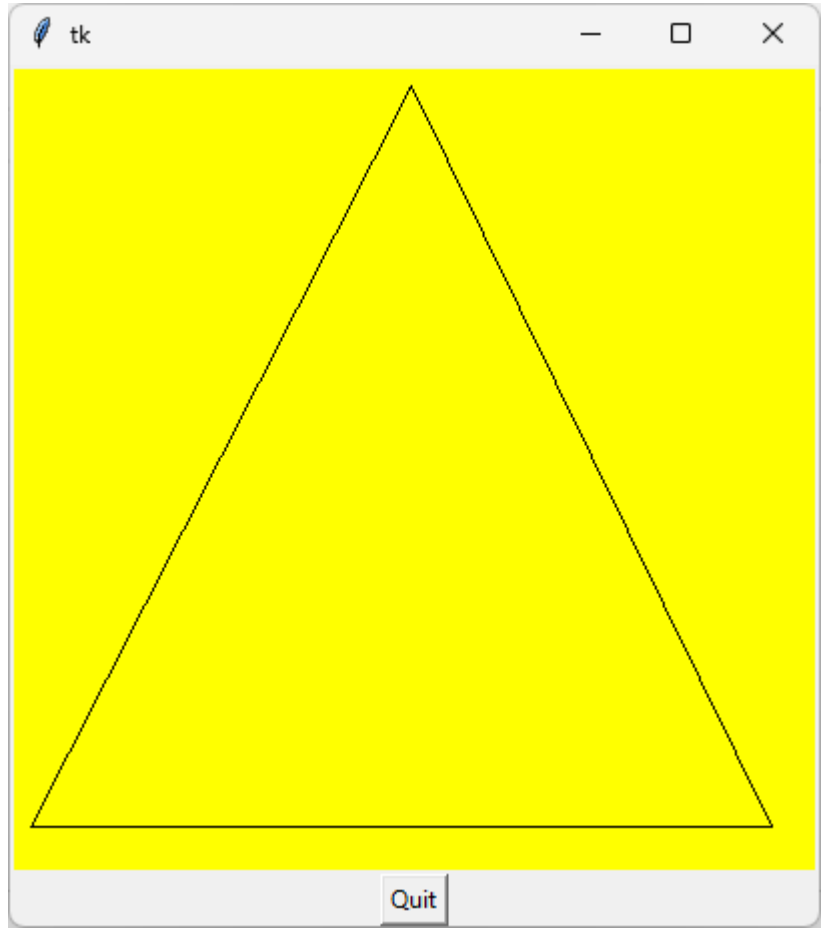
Property name	Property role
`borderwidth`	canvas border's **width** in pixels (default: 2)
`background` (`bg`)	canvas border's **color** (default: the same as the underlying window's color)
`height`	canvas **height** (in pixels)
`width`	canvas **width** (in pixels)

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...)

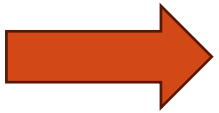
Option name	Option meaning
`arrow`	normally, the chain ends aren't marked in any special way, but you may want them to be finished with **arrowheads**; setting the arrow option to `FIRST` results in drawing an arrowhead at the chain's beginning, `LAST` at the chain's end, `BOTH` at both sides of the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

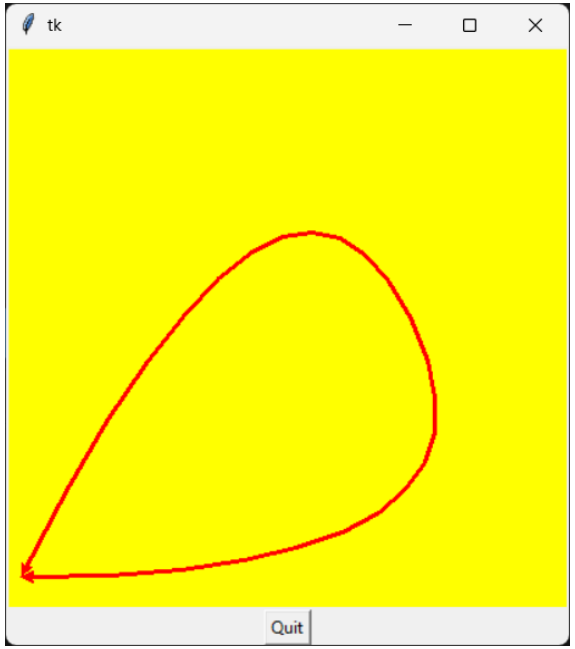
02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- 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...)

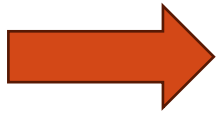
Option name	Option meaning
`outline`	rectangle **edge color** (if specified as an empty string, the edge is transparent)
`fill`	rectangle **interior color**
`width`	rectangle **edge width** in pixels (default: 1)

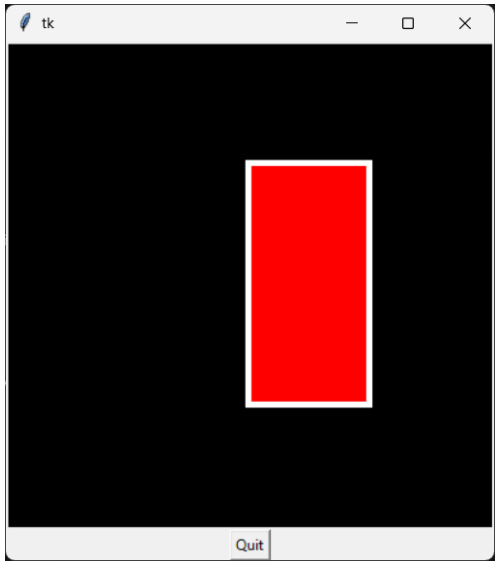
02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- 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()
```



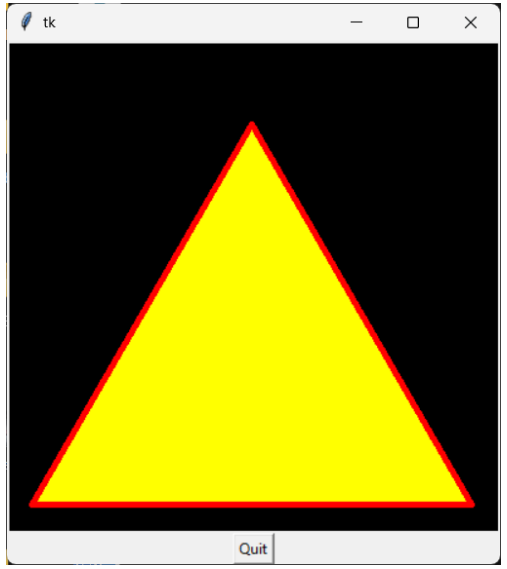


02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- Clase Canvas:
  - Método **create_poligon**: 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()
```

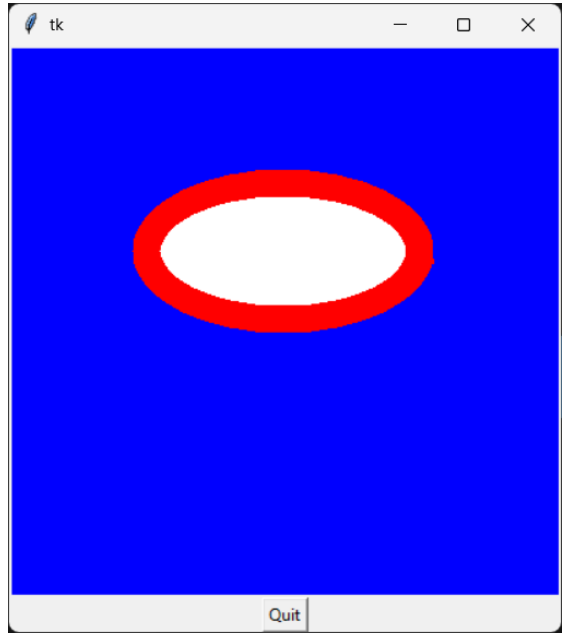


02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- 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...)

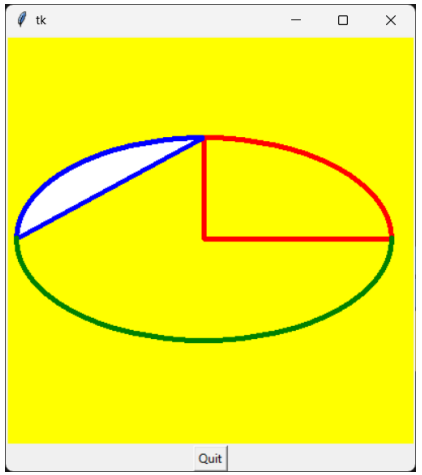
Option name	Option meaning
`style`	can be set to one of the following: `PIESLICE` (default), `CHORD` and `ARC`; the **shape** of the resulting drawing is presented here: 
`start`	the **angle** (in degrees) of the arc's start relative to the X-axis (e.g., 90 means the highest point of the ellipse, while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

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
`fill`	text **color**
`font`	text **font**
`justify`	text **justification**: `LEFT` (default), `CENTER`, `RIGHT`
`text`	**text** to display (`\n` works as expected)
`width`	normally, the rectangle is as wide as the **longest text line**; 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...)

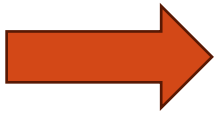
Option name	Option meaning
`image`	an object of the `PhotoImage` class containing the image itself; the `PhotoImage` class constructor needs a keyword argument named `file` pointing to a **bitmap file** (note: only GIF and PNG formats are accepted); the argument should specify the file's path

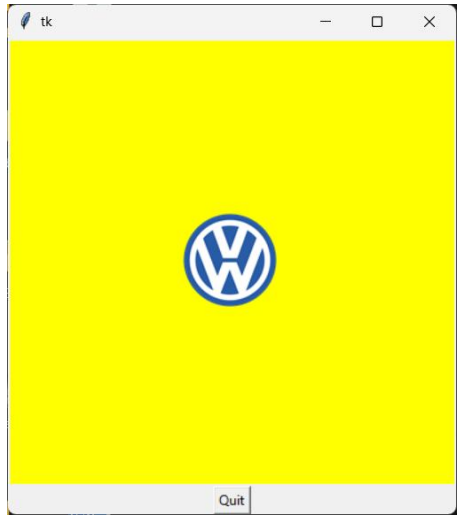
02 - LEXICON OF WIDGETS AND MORE TKINTER ESSENTIALS

- 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()
```