

Interfaces Gráficas de Usuario

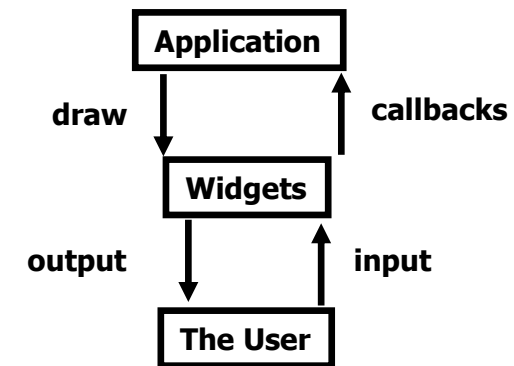
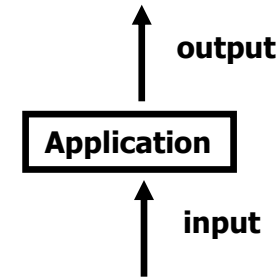
Unidad 13

Python para principiantes
dgm@uma.es



Conceptos Generales

- Una Interfaz Gráfica de Usuario (GUI) permite interactuar con una aplicación de manera "amigable"
- Programación convencional:
 - La secuencia de operaciones es determinada por el programa
 - Lo que quieres que ocurra, ocurre cuando tú quieres
- Programación dirigida por eventos:
 - La secuencia de operaciones es determinada por la interacción del usuario con la interfaz de la aplicación
 - Cualquier cosa que ocurra, ocurre en cualquier momento



La "experiencia de usuario"

- Los usuarios aprenden "por intentos"
 - Rara vez leen los manuales
 - Pensar cuidadosamente cuál tendría que ser el comportamiento por defecto de cualquier función de nuestra aplicación
- Deberíamos ayudar al usuario...
 - Usando teclas y opciones familiares (p.ej. Ctrl + C para copiar)
 - Incluyendo ayuda y tutoriales

Diseñando para los usuarios

- En cada momento, pensar: ¿es obvio lo que hace?
- Hacer todas las pantallas tan simples como sea posible
- Desconectar funcionalidades hasta que se necesiten. Por ejemplo:

```
model_menu.entryconfig("Run model", state="disabled")
```

```
# Hasta que el usuario elija los ficheros, entonces:
```

```
model_menu.entryconfig("Run model", state="normal")
```

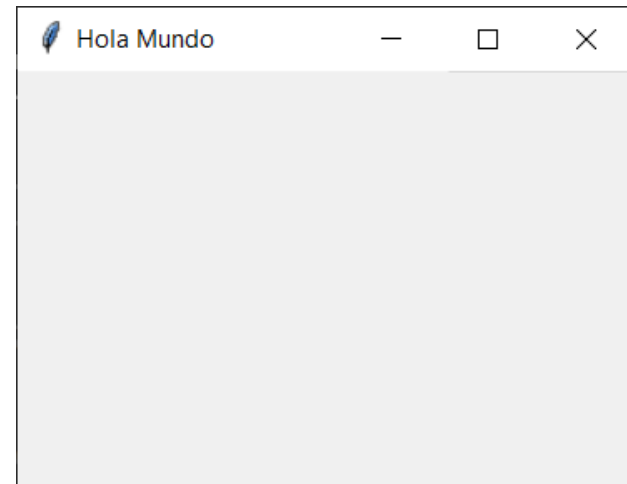
- Ocultar la funcionalidad compleja y las opciones para cambiar el comportamiento por defecto en los menús de "Opciones"

Tkinter

- Tkinter (“**ToolKit interface**”) es una librería Python que permite desarrollar GUIs
 - <https://docs.python.org/3/library/tk.html>
- Es una capa sobre Tcl/Tk para Python y está incluida por defecto en Python 3 (no es la única opción)

```
import tkinter as tk

ventana1=tk.Tk()
ventana1.title("Hola Mundo")
ventana1.mainloop()
```



Orientación a Objetos y GUIs

- Podemos aplicar Orientación a Objetos para hacer nuestro código más reutilizable

```
import tkinter as tk

class Aplicacion:
    def __init__(self):
        self.ventana1=tk.Tk()
        self.ventana1.title("Hola Mundo")
        self.ventana1.mainloop()

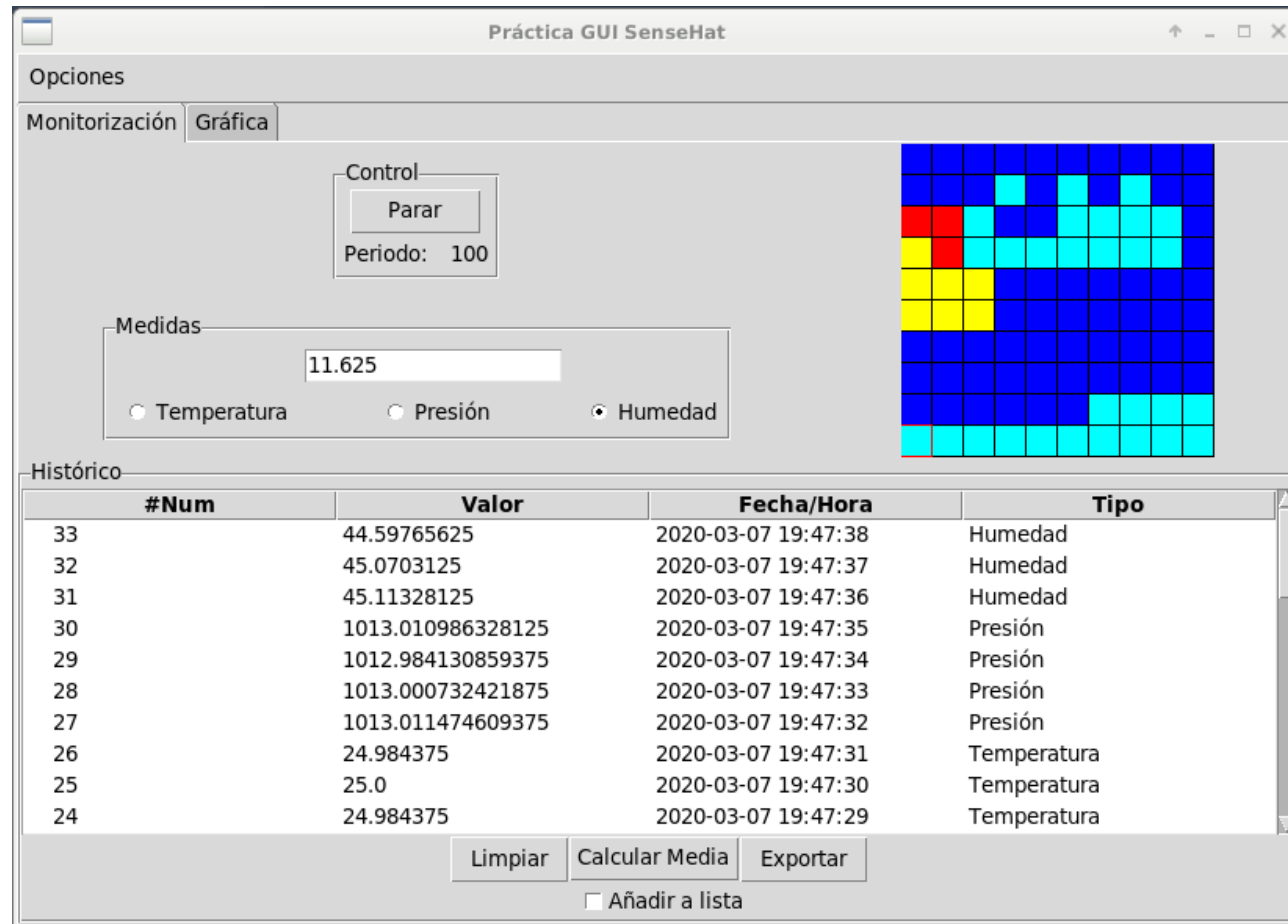
aplicacion1=Aplicacion()
```

Repositorio de Ejemplos

- Todos los ejemplos (y más) de la presentación pueden encontrarse en el repositorio <https://github.com/dgarridouma/ejemplos-tkinter>

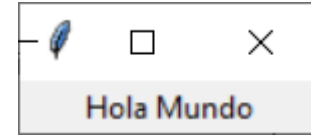
Widgets

- Los widgets son elementos que se pueden añadir a nuestra Ventana y con los que interactuará el usuario. Ejemplos:
 - Buttons, Checkbuttons, Radiobuttons, Menus, Entry, Labels, Frames, Scale, Scrollbar, Canvas...



Widgets

- Ejemplo: Añadir una Etiqueta (Label)
 - Crear la etiqueta
 - Pasar la Ventana donde se colocará y el texto
 - Definir la posición de la etiqueta (`self.label.pack()`)
 - Indica que se coloque en la Ventana y se muestre
 - Comenzar el bucle de eventos (`root.mainloop()`)



```
import tkinter as tk

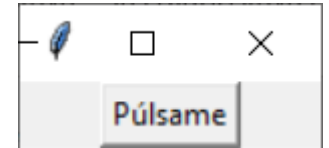
class Aplicacion:
    def __init__(self):
        self.ventana=tk.Tk()
        self.label=tk.Label(self.ventana, text="Hola Mundo")
        self.label.pack()
        self.ventana.mainloop()

aplicacion1=Aplicacion()
```

GUIs Manejo de Eventos

- Button:

- Para hacer que el usuario pueda interactuar con un botón, tenemos que usar ***command*** indicando qué método se ejecutará cuando se haga click sobre el botón



```
import tkinter as tk
from tkinter.messagebox import showinfo

class Aplicacion:
    def __init__(self):
        self.ventana=tk.Tk()
        self.boton=tk.Button(self.ventana, text="Púlsame",
                             command=self.saludar)

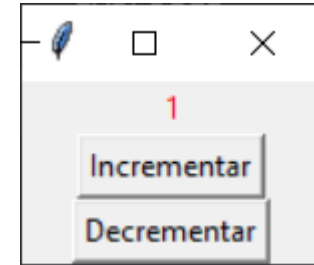
        self.boton.pack()
        self.ventana.mainloop()

    def saludar(self):
        showinfo(message="Me has pulsado")

aplicacion1=Aplicacion()
```

Ejercicio

- Implementar una interfaz de usuario como la que se muestra, de manera que cada vez que se pulse uno de los botones, el valor aumentará o disminuirá en una unidad
- Ayudas:
 - Utilizar un atributo "valor" (self.valor) en la clase, que contenga el valor actual mostrado (inicializado a 1)
 - Cuando se pulse un botón, podemos cambiar el texto de la etiqueta haciendo:
 - `self.label.config(text=self.valor)`
 - Podemos hacer que la etiqueta salga en rojo utilizando:
 - `self.label1.configure(foreground="red")`
- Bonus: hacer que la etiqueta salga en azul si el valor es ≥ 0 , rojo en caso contrario



Layouts

- Tkinter proporciona 3 formas diferentes de colocar los widgets dentro de su contenedor
 - No pueden mezclarse en el mismo contenedor
- Pack
 - Se indican posiciones relativas. Posiblemente el más fácil
- Grid
 - Se indica fila o columna. El tamaño se determina automáticamente
- Place
 - Permite indicar posiciones y medidas exactas en términos absolutos o relativos a otra ventana

Pack layout

El método `pack()` especifica la posición de un widget dentro de su contenedor



Opción	Descripción
side	LEFT, RIGHT, TOP, BOTTOM,
fill	'both', 'x', 'y', o 'none'
expand	True o False

```
from tkinter import Tk, Label, PhotoImage, BOT
TOM, LEFT, RIGHT, RIDGE
root = Tk()

text = Label(root, font=('Helvetica', 16,
    'bold italic'), foreground='white',
    background='black', pady=10,
    text='Universidad de Málaga')
text.pack(side=BOTTOM)

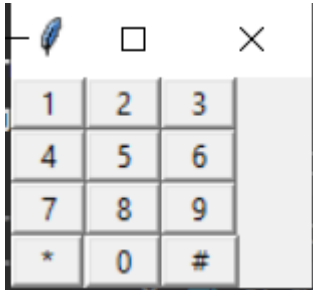
uma1 = PhotoImage(file='img/uma1.png')
umaLabel = Label(root,
    borderwidth=3, relief=RIDGE, image=uma1)
umaLabel.pack(side=LEFT)

uma2 = PhotoImage(file='img/uma2.png')
uma2Label = Label(root, image=uma2)
uma2Label.pack(side=RIGHT)

root.mainloop()
```

Grid layout

El método `grid()` permite colocar widgets en una cuadrícula



Opciones

`column`

`columnspan`

`row`

`rowspan`

```
from tkinter import Tk, Label, RAISED

root = Tk()
labels = [['1', '2', '3'],
          ['4', '5', '6'],
          ['7', '8', '9'],
          ['*', '0', '#']]

for r in range(4):
    for c in range(3):
        # crear label para fila r y columna c
        label = Label(root,
                      relief=RAISED,
                      padx=10,
                      text=labels[r][c])
        # colocar label en fila r y columna c
        label.grid(row=r, column=c)

root.mainloop()
```

Place layout

El método `place()` permite colocar widgets en posiciones específicas y con el tamaño indicado



Opciones

`x`

`y`

`width`

`height`

```
import tkinter as tk
import random

root = tk.Tk()
root.geometry("170x200+30+30")

temas = ['Python', 'POO', 'Git', 'GUI', 'BBDD']
labels = range(5)
for i in range(5):
    ct = [random.randrange(256) for x in range(3)]
    ct_hex = "%02x%02x%02x" % tuple(ct)
    bg_colour = '#' + "".join(ct_hex)
    l = tk.Label(root,
                  text=temas[i],
                  bg=bg_colour)
    l.place(x = 20, y = 30 + i*30, width=120,
            height=25)

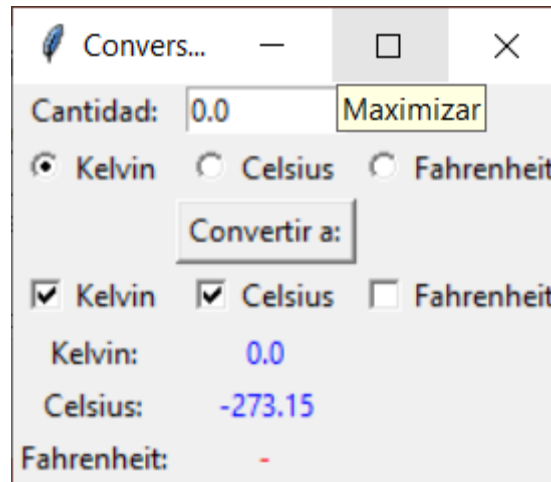
root.mainloop()
```

Más Widgets...

- Vamos a ver más ejemplos de widgets del repositorio <https://github.com/dgarridouma/ejemplos-tkinter>
- En concreto vamos a ver ejemplos de los siguientes widgets:
 - Entry
 - Radiobutton
 - Checkbutton
 - Listbox y scroll

Ejercicio

- Conversor de temperaturas: implementar un conversor de temperaturas con una interfaz similar a la siguiente



Convers...

Cantidad: 0.0 Maximizar

☒ Kelvin ☐ Celsius ☐ Fahrenheit

Convertir a:

☒ Kelvin ☒ Celsius ☐ Fahrenheit

Kelvin: 0.0

Celsius: -273.15

Fahrenheit: -

- Bonus: Añadir un control de lista a la interfaz que almacene todas las conversiones realizadas

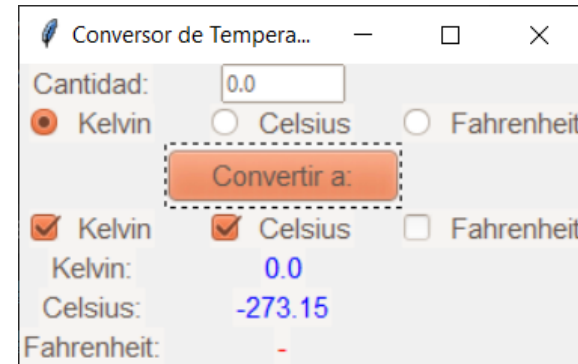
ttk

- El modulo **ttk** proporciona acceso a "temas" en los widget
- La idea básica es separar el comportamiento de la apariencia (y un aspecto más "nativo")
- Para comenzar a usar Ttk
 - **from tkinter import ttk**
- Automáticamente varios widget de Tk son reemplazados por las nuevas versiones
- Ttk viene con 6 nuevos widgets no incluidos en Tk
 - Combobox, Notebook, Progressbar, Separator, Sizegrip y Treeview

Aplicando temas

- Vamos a cambiar el tema a nuestro conversor de temperaturas
 - <https://ttkthemes.readthedocs.io/en/latest/themes.html>
- Instalamos el paquete ttkthemes
 - `pip3 install ttkthemes`
- Importamos los paquetes necesarios
 - `import tkinter as tk`
 - `import tkinter.ttk as ttk`
 - `from ttkthemes import ThemedStyle`
- Aplicamos el tema a nuestra Ventana

```
self.ventana1=tk.Tk()
style = ThemedStyle(self.ventana1)
style.set_theme("radiance")
```
- Puede que tengamos que hacer algunos ajustes a nuestro código



Widgets Ttk

- Vamos a ver más ejemplos de widgets, incluyendo widgets Ttk y otros elementos como Menús: <https://github.com/dgarriouma/ejemplos-tkinter>
- En concreto vamos a ver ejemplos de los siguientes widgets:
 - Combobox
 - Menus
 - Notebook
 - Frame
 - Labelframe

Ejercicio

- Vamos a mejorar la apariencia de nuestro conversor de unidades añadiendo widget LabelFrames. El resultado debe ser similar al siguiente:
 - Pista: en Resultados usar sticky="WE" en el LabelFrame

Conversor...

Datos de Entrada:

Cantidad: 0.0

☒ Kelvin ☐ Celsius ☐ Fahrenheit

Datos Conversión:

Convertir a:

☒ Kelvin ☒ Celsius ☐ Fahrenheit

Resultados:

Kelvin: 0.0

Celsius: -273.15

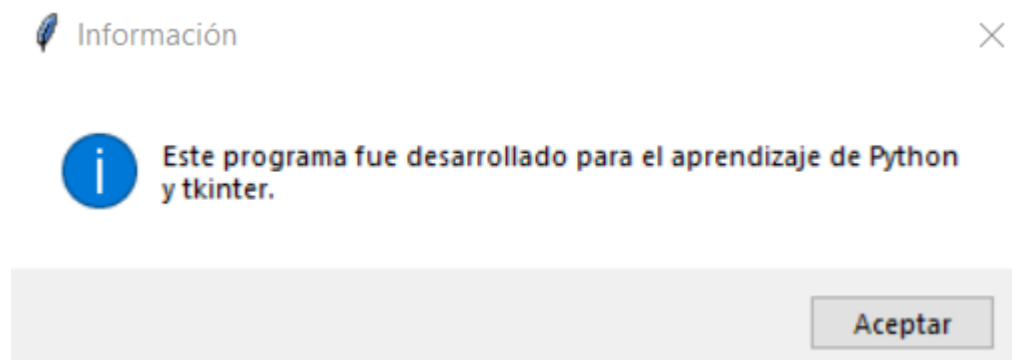
Fahrenheit: -

Diálogos

- Los cuadros de información son un mecanismo muy útil para mostrar información al usuario
 - Hay de varios tipos (información, advertencia y error, sí/no)
- Ejemplo:

```
from tkinter import messagebox as mb
```

```
mb.showinfo("Información", "Este programa fue desarrollado para el  
aprendizaje de Python y tkinter.")
```



Diálogos

- Los **diálogos** se usan para mostrar ventanas adicionales
 - Pueden ser usados para muy diversas tareas: recuperar información del usuario, modificar opciones, ...
- Se suelen desarrollar en una clase aparte
- Hay que dar la orden de mostrarlas y habitualmente tienen un comportamiento "modal". Hasta que no se cierran no se puede volver a la ventana inicial
- Una vez cerrada, podemos recoger el resultado

Diálogos

- Definición

```
class DialogoTamano:

    def __init__(self, ventanaprincipal):
        self.dialogo=tk.Toplevel(ventanaprincipal)
        # Widgets del diálogo aquí
        self.dialogo.protocol("WM_DELETE_WINDOW", self.confirmar)
        self.dialogo.resizable(0,0)
        self.dialogo.grab_set()

    def mostrar(self):
        self.dialogo.wait_window()
        return (self.dato1.get(), self.dato2.get())

    def confirmar(self):
        self.dialogo.destroy()
```


Diálogos

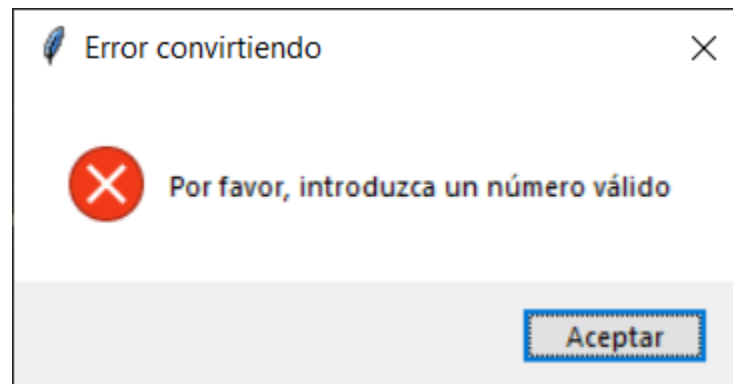
- Creación y recuperación de resultados

```
def configurar(self):  
    dialogo1 = DialogoTamano(self.ventana1)  
    s=dialogo1.mostrar()  
    self.ventana1.geometry(s[0]+"x"+s[1])
```

- Ejemplos completos en
 - <https://github.com/dgarridouma/ejemplos-tkinter>

Ejercicio

- Vamos a comprobar que el número introducido en nuestro conversor es válido, y si no lo es, mostraremos un mensaje de error usando el método `messagebox.showerror`



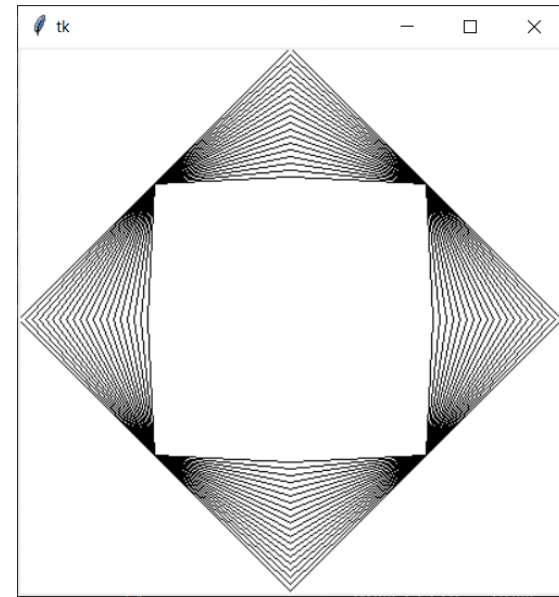
- Bonus: añadir un menu con una opción que muestre el típico diálogo "Acerca de"

Y aún más widgets!

- <https://github.com/dgarridouma/ejemplos-tkinter>
- En concreto vamos a ver ejemplos de los siguientes widgets:
 - Spinbox
 - Progressbar
 - Scrolledtext
 - Treeview

Canvas

- El widget Canvas permite mostrar elementos gráficos como líneas, rectángulos o círculos, así como controlar clicks del ratón o pulsaciones de teclas
- Una muestra...

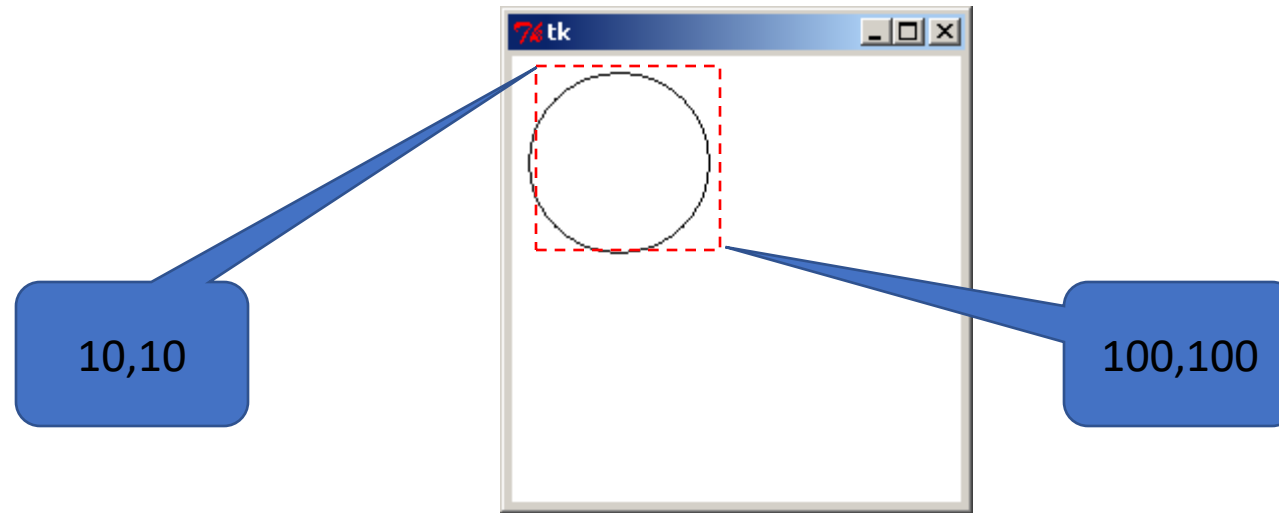


```
self.canvas1=tk.Canvas(self.ventana1, width=400, height=400,
background="white")
self.canvas1.pack()

i=0
while i<100:
    i+=5
    self.canvas1.create_line([(100,100),(200,100-i),
(300,100),(300+i,200),(300,300),(200,300+i),(100,300),
(100-i,200),(100,100)], smooth=0)
```

Importante

Saber cómo funcionan las coordenadas



```
from tkinter import *
tk = Tk()
c = Canvas(tk, width=800, height=600, bg="white")
c.create_oval(10,10,100,100)
c.pack()
tk.mainloop()
```

Más ejemplos con Canvas

- <https://github.com/dgarridouma/ejemplos-tkinter>
- Entre otras cosas veremos:
 - Crear diversas figuras
 - Controlar la posición del ratón y sus pulsaciones
 - Controlar la pulsación de teclas
 - Asignar identificadores a las formas creadas
 - Mover y borrar las formas
 - Mostrar imágenes en el canvas

Integración con matplotlib

- Las figuras de matplotlib pueden ser integradas en tkinter
 - https://matplotlib.org/3.1.0/gallery/user_interfaces/embedding_in_tk_sgskip.html

```
import tkinter as tk
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure

class Aplicacion:
    def __init__(self):
        self.ventana=tk.Tk()
        self.fig = Figure()
        self.ax = self.fig.add_subplot(111)
        self.ax.set_xlabel("X axis")
        self.ax.set_ylabel("Y axis")
        self.ax.cla()    # clear axis
        self.ax.grid()   # configura grid
        self.ax.set_xlim(-10, 10)
        self.ax.set_ylim(0, 100)
```

Integración con matplotlib

- Continuación...

```
x = range(-10,11)
y = [v*v for v in x]
self.line, = self.ax.plot(x, y, marker='o', color='orange')

self.graph = FigureCanvasTkAgg(self.fig,
                                master=self.ventana)
# Agg: Anti-Grain geometry rendering engine
self.graph.get_tk_widget().pack(side="top", fill='both',
                                expand=True)

self.ventana.mainloop()
```

Aplicacion()

- <https://github.com/dgarriDOMarquez/ejemplos-tkinter>

Acknowledgements / Contributions



These slides are Copyright 2020- Daniel Garrido of the University of Málaga and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Continue...

Initial Development: Daniel Garrido, University of Málaga

... Insert new Contributors and Translators here