

Programación Técnica y Científica

Prof.Miguel García Silvente

Tema 3

Introducción a la programación de interfaces de usuario en Python

Introducción

- Existen distintas opciones:
 - Tkinter (Tk)
 - wxPython (wxWindows)
 - PyQt (Qt)
 - PyGtk (Gtk)
 - PySide (Qt)
 - PythonWin (MFC)

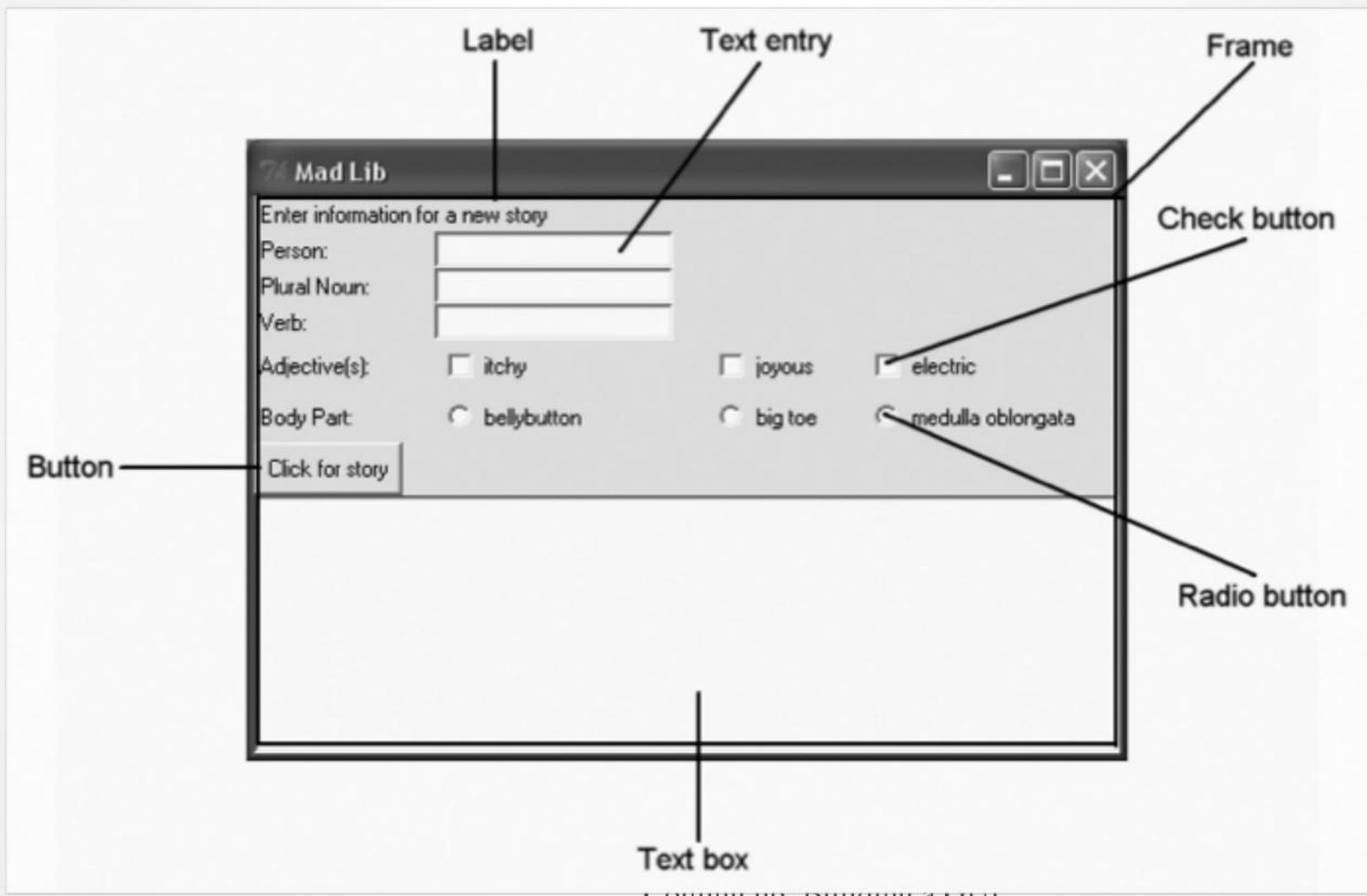
tkinter

- tkinter es un interfaz Python a la biblioteca Tk.
 - Tk es una biblioteca gráfica estándar
- Es la utilizada por defecto
- tkinter forma parte de Python. Para usarlo:
 - `from tkinter import *`

Uso de tkinter

- tkinter da la oportunidad de poder crear ventanas con distintos elementos dentro de él (widgets).
- Definición: **widget** es un componente gráfico dentro en un ventana (botón, etiqueta de texto, menú desplegable, barra de scroll, fotografía, etc...)
- Los GUIs se construyen combinando varios de estos widgets.

Widgets (Componentes GUI)

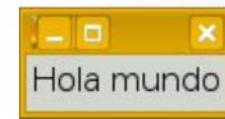


Construcción de un GUI

- Crear widgets
 - ▶ Se construyen los widgets que van a aparecer.
- Disposición de widgets
 - ▶ Determinar donde se colocan los widgets (y cómo se mueven cuando se cambia el tamaño de la ventana o se hace scroll, etc)
- Procesar Eventos
 - ▶ Escribir funciones que procesen eventos como pulsar un botón, pulsar botón del ratón, etc...

Primera ventana

```
from tkinter import *
def main():
    root = Tk() # ventana raíz
    root.grid()
    # widget con texto asociado. Se indica la ventana
    etiqueta = Label(root, text="Hola mundo")
    etiqueta.grid() # Para que aparezca la etiqueta
    # Lo siguiente, obligatorio para que aparezcan las ventanas
    root.mainloop() # Entra en un “bucle de eventos”
main()
```



Los widgets son objetos

<http://www.effbot.org/tkinterbook/tkinter-index.htm#class-reference>

- Clases:
 - ▶ Button, Canvas, Checkbutton, Entry, Frame, Label, Listbox, Menu, Menubutton, Message, Radiobutton, Scale, ScrollBar, Text, TopLevel, and many more...

Más objetos

```
from tkinter import *
```

```
def main() :
```

```
    root = Tk()
```

```
    root.geometry("200x100")
```

```
    etiqueta = Label(root, text="Hola mundo")
```

```
    etiqueta.grid(row=0, column=0)
```

```
    boton = Button(root, text="Salir")
```

```
    boton.grid(row=0, column=1)
```

```
    root.mainloop()
```

```
main()
```

Hay que asociarle un evento al botón



Asociar una tarea I

```
from tkinter import *

def botonPulsado() :
    print("botón pulsado")

def main() :
    root = Tk()
    root.geometry("200x100")
    etiqueta = Label(root, text="Hola mundo")
    etiqueta.grid(row=0, column=0)
    boton = Button(root, text="Salir", command=botonPulsado)
    boton.grid(row=0, column=1)
    root.mainloop()

main()
```

Asociar una tarea II

```
from tkinter import *

def botonPulsado() :
    global raiz
    raiz.destroy()

def main() :
    global raiz

    root = Tk()
    root.geometry("200x100")
    etiqueta = Label(root, text="Hola mundo")
    etiqueta.grid(row=0, column=0)
    boton = Button(root, text="Salir", command=botonPulsado)
    boton.grid(row=0, column=1)

    raiz = root

    root.mainloop()

main()
```

Crear entrada de texto

Forma general para todos los widgets:

1. # Crear el widget

```
widget = <nombrewidget>(padre, atributos...)
```

2. widget.grid(row=<numero>, column=<numero>)

muestra el widget

```
tBox = Entry(parent)
```

```
tBox.grid(row=3, column=0)
```



Uso de una entrada de texto

Para usarla se debe poder obtener información cuando la necesites. (Generalmente como respuesta a un evento)

Hay que crearlo como un atributo para que se pueda obtener información al pulsar un botón

Uso de una entrada de texto

- Crearlo como un atributo
- Usar el método “get” cuando sea necesario para obtener el texto introducido



Uso de una entrada de texto (II)

```
from tkinter import *
entryBox = None

def botonPulsado():
    global entryBox
    txt = entryBox.get()
    print ("El texto es:",txt)

def crearCajaTexto(parent):
    global entryBox
    entryBox = Entry(parent)
    entryBox.grid()

def main():
    root = Tk()
    miBoton = Button(root, text="Mostrar texto",command=botonPulsado)
    miBoton.grid()

    crearCajaTexto(root)
    root.mainloop()

main()
```

Crear etiquetas modificables

- Las etiquetas habitualmente no se pueden cambiar una vez creadas
- Una alternativa: usar el objeto StringVar
- Asignar un StringVar a la etiqueta
- Al cambiar el StringVar, la etiqueta cambia

```
miTexto = StringVar()  
miTexto.set("Algo")
```

```
# Asociar el StringVar con la etiqueta  
etiqueta = Label(parent,textvariable=miTexto)
```

```
# Modifica el texto  
miTexto.set("Otro valor")
```

Text widgets

- Usa Text widgets para entradas de datos con varias líneas de texto
- `aWidget = Text(root, width=40, height=5, borderwidth=3)`
Crea una entrada de texto de 40 caracteres de ancho, 5 líneas de alto y con un borde de 3 píxeles de ancho
- `aWidget.config(relief = RAISED)`
Cambia el aspecto del borde
SUNKEN, RAISED, GROOVE, RIDGE o FLAT
- `aWidget.config(borderwidth = 2)`. Para que aparezca recuadrado
- `aWidget.delete(1.0, END)` Vacía la entrada
- `aWidget.insert(END, cadena)` Inserta un string
- `aWidget.get(0.0, END)`. Devuelve todo el texto

Añadir una barra de scroll

La barra de Scroll aparece si el texto no cabe.

1. Crear la entrada de texto

`tBox = Text(root, width=40, height=5)`

2. Ponerlo en el grid

`tBox.grid(row=0, column=0, columnspan=5)`

3. Crea la barra de scroll

`scrollBarY = Scrollbar (root, orient=VERTICAL,command=tBox.yview)`

4. Mostrarlo con grid

`scrollBarY.grid (row=0, column=6,sticky=N+S)`

5. Asociar la orden de scroll con la barra de scroll

`tBox.config(yscrollcommand = scrollBarY.set)`

<http://effbot.org/zone/tkinter-scrollbar-patterns.htm>

Métodos generales

- `w.config(bg="red")` #cambia el color
- `w.config(state=DISABLED)` # difumina el widget
- `w.config(state=NORMAL)` # Restablece el aspecto

<http://effbot.org/tkinterbook/widget.htm>

Más sobre widgets

<http://effbot.org/tkinterbook/>

<http://infohost.nmt.edu/tcc/help/pubs/tkinter/>

<http://www.pythonguis.com/library/tkinter/introduction/index.htm>

Gestión del diseño

- Siempre hay que usar grid. Si no, los distintos widgets no aparecen.
- grid es el responsable de la disposición de los demás elementos. Determina donde se colocan los distintos widgets en la ventana y qué ocurre cuando la ventana cambia de tamaño o se añaden o eliminan widgets,etc
- Las distintas bibliotecas gráficas tienen este sistema de gestión del diseño para ayudar a organizar los widgets.

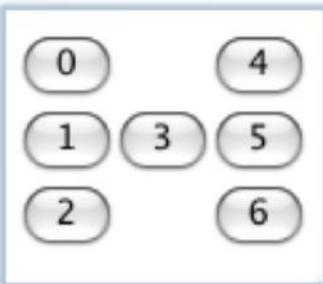
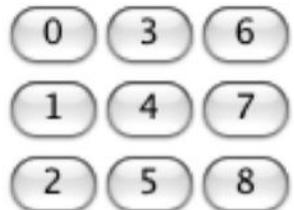
Parámetros de grid (I)

- **row, column** – especifica la localización de cada widget.
 - ▶ 0,0 es la esquina superior izquierda.
 - ▶ Las filas vacías se descartan (no se admiten espacios vacíos)
- **rowspan, rowspan** – especifica cuantas filas y columnas puede ocupar un determinado widget.
- **padx, pady** – especifica cuando espacio vacío se debe colocar alrededor del widget

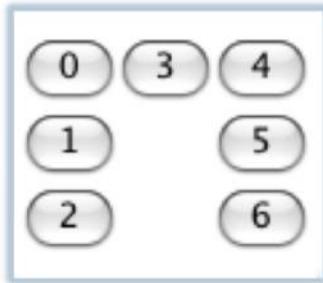
Parámetros de grid (II)

- **sticky** – Define cómo expandir el widget si el espacio que tiene es mayor que su tamaño. Puede ser una combinación de S, N, E y W, o NW, NE, SW y SE.
- Por ejemplo, W (oeste) significa que el widget se debe alinear al borde izquierdo. W+E significa que se debe alargar horizontalmente para llenar el espacio. W+E+N+S significa que se debe expandir en ambas direcciones.
- Por defecto se centra.

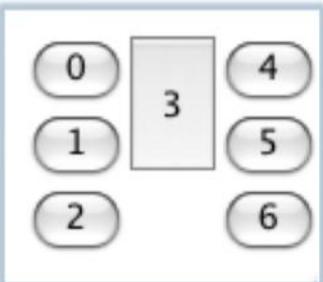
Ejemplos



`createButton(root).
grid(row=0,column=1, rowspan=3)`

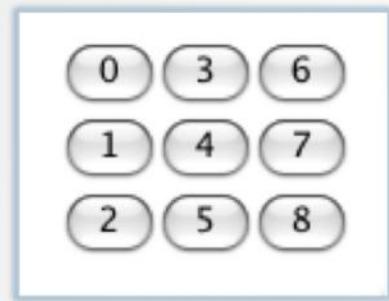


`createButton(root).
grid(row=0,column=1,
rowspan=3, sticky=N)`



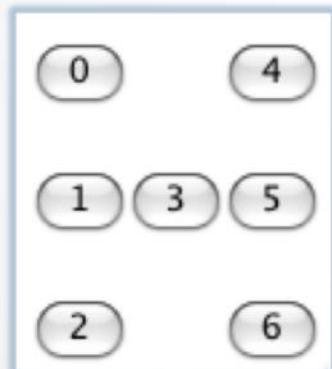
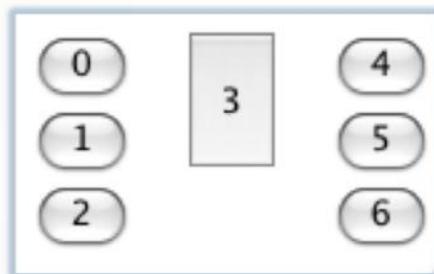
`createButton(root).
grid(row=0,column=1,
rowspan=2, sticky=N+S)`

Ejemplos (II)



`createButton(root).`

`grid(row=0,column=1, rowspan=2,
sticky=N+S, padx=20)`



`createButton(root).`

`grid(row=1,column=1,
sticky=N+S, pady=20)`

Otros gestores de diseño

Python tiene varias opciones:

- pack – permite colocar elementos unos juntos a otros de distintas formas y permite expandirlos
- grid – permite especificar una localización por fila, columna y cuantas filas y columnas se puede expandir cada uno
-
- place – especifica una localización por pixeles

IMPORTANTE: Nunca se deben usar varios al mismo tiempo porque no son compatibles entre sí y podrían ocasionar bucles infinitos.

Mostrar imágenes

Una imagen es un tipo de widget.

```
photo = PhotoImage(file='somefile.gif')
```

tkinter sólo admite GIF, PGM, PBM. Para leer JPGs hay que usar
Python Imaging Library

```
im = PhotoImage(file='cake.gif') # Crea el widget PhotoImage
```

```
# Añade la foto a una etiqueta:
```

```
w = Label(root, image=im) # Crea una etiqueta con una imagen
```

```
w.image = im # Siempre se debe guardar una referencia para evitar la  
recolección de basura (limpieza de memoria) y que desaparezca
```

```
w.grid() # Pone la etiqueta en la ventana
```

Mostrar imágenes (II)

Un Canvas es un contenedor que permite mostrar imágenes y dibujar. Se pueden representar gráficas, mostrar gráficos, manejar eventos con el ratón,etc.

```
myCanvas = Canvas(root, width=400, height=200)
myCanvas.create_line(0, 0, 200, 100)
myCanvas.create_line(0, 100, 200, 0, fill="red", dash=(4, 4))
myCanvas.create_image(0, 0, anchor=NW, image=myPhotoImage)
```

<http://effbot.org/tkinterbook/canvas.htm>

Ejemplo de uso de Canvas

```
def Ejemplo(root):
    root.title("Colors")

    canvas = Canvas(root)
    canvas.create_rectangle(30, 10, 120, 80, outline="#fb0", fill="#fb0")
    canvas.create_rectangle(150, 10, 240, 80, outline="#f50", fill="#f50")
    canvas.create_rectangle(270, 10, 370, 80, outline="#05f", fill="#05f")
    canvas.grid()
```

```
root = Tk()
Ejemplo(root)
root.geometry("400x100+300+300")
root.mainloop()
```



Título de una ventana

- Tan sólo hay que llamar al método title aplicado a la ventana inicial:

```
root.title("Título de la ventana")
```

Mensaje de diálogo

- Es una pequeña ventana modal (detiene temporalmente los eventos) que aparece por encima de la principal
 - ▶ Para hacer una pregunta, mostrar un mensaje, etc
 - ▶ Por ejemplo: Fichero -> Abrir

```
from tkinter.messagebox import *

showinfo(title="Fin del juego",
         message="Has resuelto correctamente el puzzle")
```

- Se puede usar tambien **showwarning**, **showerror** , sólo cambia el icono que aparece en la ventana

Ventanas con preguntas

```
from tkinter.messagebox import *

res = askyesno("Continuar", "¿Debemos continuar?")
res será True (para Si) o False (para No)
```

Otras opciones: askokcancel, askretrycancel, askquestion

AVISO: askquestion devuelve “yes” o “no” como strings, NO
True y False!

Ventanas de selección de archivos

- Es posible
 - ▶ Abrir un fichero
 - ▶ Seleccionar un directorio
 - ▶ Seleccionar un fichero para guardar
- `from tkinter.filedialog import *`
 - ▶ métodos: askdirectory, askopenfile, askopenfilename, askopenfilenames, askopenfiles, asksaveasfile, asksaveasfilename

<http://www.pythonguis.com/library/tkinter/introduction/x1164-data-entry.htm>

Entrada de datos

- Se puede usar `tkinter.simpledialog` para introducir un número o un string usando una ventana:

`askstring(title, prompt),
askinteger..., askfloat...`

```
from tkinter.simpledialog import *
res = askstring("Título", "Introduce tu nombre")
print (res)
res = askinteger("Título", "Introduce un entero")
print (res)
res = askinteger("Num", "Introduce un entero entre 0 y 100",
    minvalue=0, maxvalue=100)
print (res)
```

Ejemplo de entrada de datos

```
from tkinter import *
from tkinter.simpledialog import *

def main():
    root = Tk()
    root.withdraw() # esconde la ventana
    res = askstring("Título", "Una cadena")

    print(res)

main()
```

Más información

Sobre diálogos de todo tipo:

- <http://infohost.nmt.edu/tcc/help/pubs/tkinter/dialogs.html>
- <http://www.pythontutorial.net/tkinter/tkinter-standard-dialogs/>

Captura de eventos de ratón

- Hay que asociar los eventos a un widget.
 - ▶ `widget.bind(event, handler)`
 - ▶ event es lo que hace el usuario
 - ▶ handler es la función a la que se llama cuando el usuario realiza el evento.
 - ▶ Los eventos pueden ser:
 - ▶ <Button-1>
 - (1 es el botón izquierdo, 2=derecho, 3=central)
 - ▶ <Double-Button-1> - doble click del botón 1
 - ▶ <Enter> - ratón entra en el widget
 - ▶ <Leave> - ratón deja el widget
 - ▶ <Return> - usuario pulsa el botón enter
 - ▶ <key> (<a> for example) – usuario pulsa “a”

Capturar eventos del ratón

Por ejemplo, un botón que pide que lo pulsen

```
def mouseEntered(event):
    button = event.widget
    button.config(text = "Por favor, púlsame")

def mouseExited(event):
    button = event.widget
    button.config(text = "Entra de nuevo")

def main() :
    root = Tk()
    b = Button(root, text="Logon")
    b.bind("<Enter>",mouseEntered) # Asociar gestor de eventos
    b.bind("<Leave>",mouseExited)
    b.grid()
    root.mainloop()

main()
```

Manejo del ratón

```
def mouseEntered(event):
    button = event.widget
    button.config(text = "Por favor, púlsame")
```

Se usa “event.widget”... porque todos los eventos se guardan como una serie de datos generados por el evento. En este caso es un botón.

Se pueden obtener distintos datos con

```
myVariable = event.<atributo>
Por ejemplo,
print ("pulsado en", event.x, event.y)
```

<http://effbot.org/tkinterbook/tkinter-events-and-bindings.htm>

Atributos de Eventos

- **widget**: el widget que generó el evento
- **x, y** : posición actual, en píxeles
- **x_root, y_root**: la posición relativa a la esquina superior izquierda
- **char**: el carácter pulsado
- **keysym/keycode** : símbolo/ código de tecla
- **num** : el número de botón de ratón
- **width, height** : nuevo tamaño del widget
- **type** : el tipo de evento

Problema habitual

```
def main():
    global root
    root = Tk()
    b = Button(root, text="Logon")
    b.bind("<Enter>",mouseEntered)
    b.bind("<Leave>",mouseExited)
    b.pack()
    root.mainloop() # Start the event loop

main()

b.bind("<Enter>", mouseEntered) # BIEN

b.bind("<Enter>", mouseEntered()) # MAL!
```

Control del ratón: gestor eventos

- Se usa "root.mainloop()" cuyo funcionamiento es:
 - ▶ while (True): # Bucle infinito
esperar a un evento
maneja el evento (llama a un gestor de eventos
con la información del objeto del evento)
- Muchos eventos no se ven: cambio de tamaño de ventana, iconizar ventana, ventana que reaparece después de estar oculta, etc
- Se pueden capturar si se quiere pero tkinter los maneja generalmente de la forma más adecuada.

Diseño

- Diseña el GUI – Los widgets y su disposición
- Codifica el GUI
- Indica los eventos que quieras manejar
 - ▶ Asocia los eventos mediante un manejador
- Indica al sistema que empiece a aceptar eventos
`root.mainloop()`

Programación por eventos

Según la wikipedia:

- Programación por eventos – un paradigma de programación en el que el flujo está dirigida por sensores o acciones de usuario (eventos). Ejemplo: juegos
- Programación por lotes – un paradigma de programación donde el flujo de eventos está definido por el programador.
- Ejemplo:

Lotes

Respuesta a pregunta 1

Respuesta a pregunta 2

Etc...

Por eventos

Usuario pulsa botón q1

Usuario pulsa botón q3

Usuario pulsa botón q2

Etc...

Dirigida por eventos vs Procedural

Sistema Procedural

Solicita entradas a través de menús, sub-menus, etc.

El usuario debe proporcionar toda la información en un orden determinado

Procesa los datos y responde al usuario

Sistemas dirigidos por eventos

El usuario puede proporcionar información y usar los controles en cualquier orden, en función del diseño del GUI

Modelo general

- Evento
 - Un objeto generado como resultado de una interacción específica con el sistema
 - Ejemplos: pulsar un botón, click de ratón, escribir texto en un campo y pulsar <ENTER>, etc.
- Oyente de Eventos (Event Listener)
 - Un mecanismo que espera la notificación de que un determinado evento se ha producido --- mainloop es una función de este tipo.
- Manejador de Eventos
 - Un proceso que se ejecuta como respuesta a un evento que se ha generado.

Cuadros con Listas

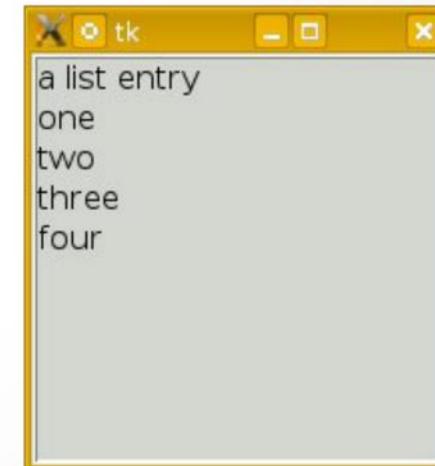
- Permiten seleccionar uno o más elementos de una lista
- <http://effbot.org/tkinterbook/listbox.htm>

```
from tkinter import *
```

```
master = Tk()  
listbox = Listbox(master)  
listbox.grid()  
  
listbox.insert(END, "a list entry")
```

```
for item in ["one", "two", "three", "four"]:  
    listbox.insert(END, item)
```

```
master.mainloop()
```



Cuadros con Listas II

```
from tkinter import *

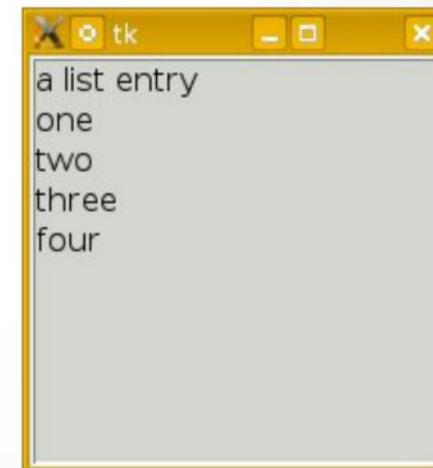
def pulsarBoton(event) :
    global listbox
    item = listbox.curselection()[0]
    print (item, event.widget.get(int(item))) # posición y valor del elemento seleccionado

master = Tk()
listbox = Listbox(master)
listbox.bind('<<ListboxSelect>>', pulsarBoton)
listbox.grid()

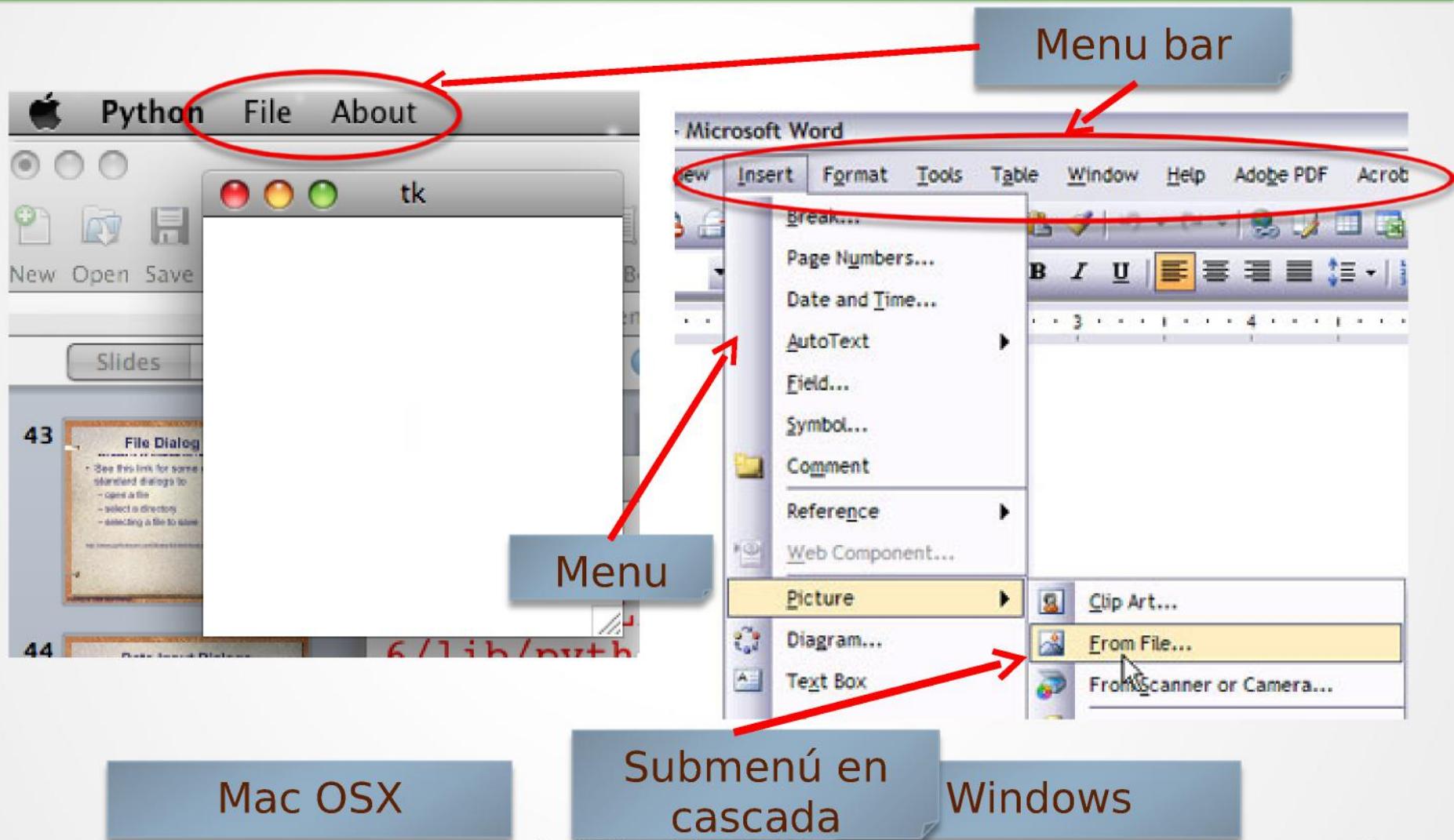
listbox.insert(END, "a list entry")

for item in ["one", "two", "three", "four"]:
    listbox.insert(END, item)

master.mainloop()
```



Menus



Mac OS X

Submenú en
cascada

Windows

Prof.Miguel García Silvente

50

Añadir menús

Es un tipo de widget

```
# crea un menú principal  
menubar = Menu(root)
```

```
# crea un menú desplegable y lo añade a la barra de menú  
filemenu = Menu(menubar)  
filemenu.add_command(label="Open", command=hello)  
  
filemenu.add_separator()  
filemenu.add_command(label="Exit", command=root.destroy)  
  
menubar.add_cascade(label="File", menu=filemenu)  
  
root.config(menu=menubar)
```



Añadir submenús

Se añade el submenú a otro menú en lugar de a un menubar.

Crear otro menú

```
helloMenu = Menu(menuBar)
helloMenu.add_command(label="Say hello", command=hello)
menuBar.add_cascade(label="Hello", menu=helloMenu)
```

Crear un submenú

```
subHello = Menu(helloMenu) # el padre helloMenu
subHello.add_command(label="English", command=hello) # Menu Item 1
subHello.add_command(label="Spanish", command=hello) # Menu Item 2
subHello.add_command(label="Chinese", command=hello) # Menu Item 3
subHello.add_command(label="French", command=hello) # Menu Item 4
```

Añadir submenú en el menu padre sub menu into parent con la etiqueta
International Hello

```
helloMenu.add_cascade(label="International Hello", menu=subHello)
```

Referencias

<http://epydoc.sourceforge.net/stdlib/tkinter.Pack-class.html#pack>

<http://effbot.org/tkinterbook>

<http://www.pythonguis.com/library/tkinter/introduction/>

Gestión de diseño con pack

- pack puede tener asociado un lado por el que agrupar:
 - myWidget.pack(side=LEFT)
 - Esto indica que se coloque myWidget a la izquierda del siguiente widget
 - Otras opciones
<http://docs.python.org/3.3/library/tkinter.html#the-packer>

Ejemplos

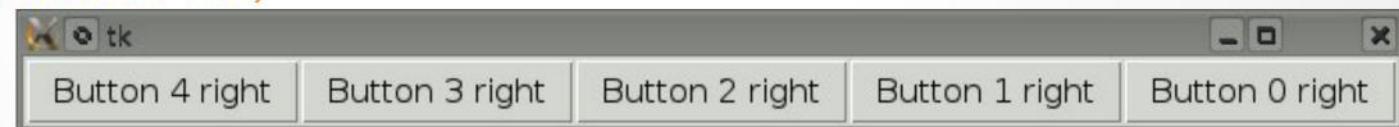
```
from tkinter import *

root = None
count = 0 # Contador de Clicks

def addButton(root, sideToPack):
    global count
    name = "Button " + str(count) + "+" + sideToPack
    button = Button(root, text=name)
    button.pack(side=sideToPack)
    count +=1

def main():
    global root
    root = Tk()
    for i in range(5):
        addButton(root, BOTTOM)
    root.mainloop()

main()
```



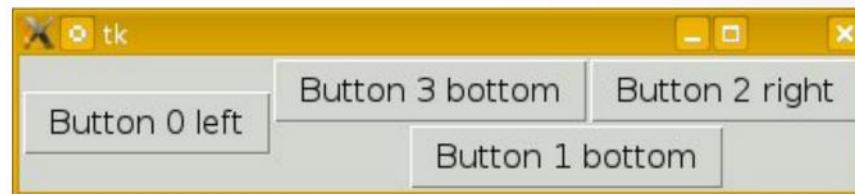
Ejemplos con pack

```
from tkinter import *
root = None
count = 0 # Clicks

def addButton(root, sideToPack):
    global count
    name = "Button " + str(count) + "+" + sideToPack
    button = Button(root, text=name)
    button.pack(side=sideToPack)
    count +=1

def main():
    global root
    root = Tk()
    addButton(root, LEFT)
    addButton(root, BOTTOM)
    addButton(root, RIGHT)
    addButton(root, BOTTOM)
    root.mainloop()

main()
```



Frames

- Facilitan la colocación de elementos
- Frames son widgets que contienen a otros widgets
- Habitualmente root tiene Frames como hijos

Agrupar con Frames (I)

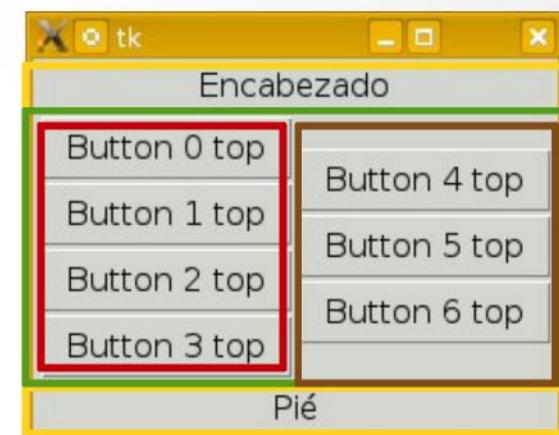
Ejemplo



Agrupar con Frames (II)

Botones verticales

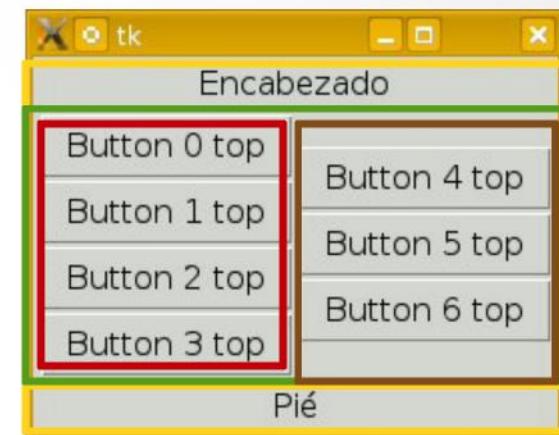
```
 addButton(root, TOP)  
 addButton(root, TOP)  
 addButton(root, TOP)  
 addButton(root, TOP)  
 addButton(root, TOP)
```



Agrupar con Frames (III)

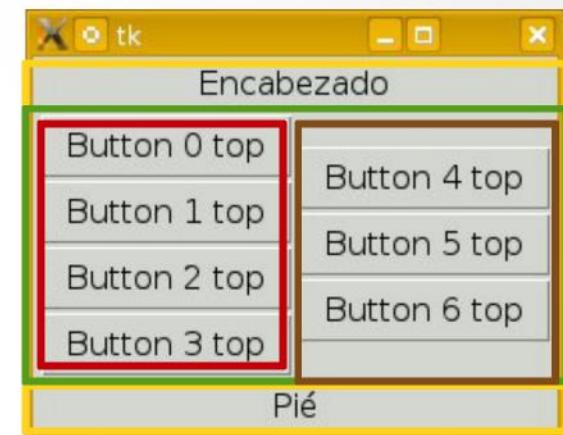
Usando frames

```
frame1 = Frame(root)
addButton(frame1 , TOP)
```



Agrupar con Frames (IV)

```
redFrame.pack(side=LEFT)  
brownFrame.pack(side=LEFT)  
topYellow.pack(side=TOP)  
green.pack(side=TOP)  
bottomYellow.pack(side=TOP)
```



Agrupar con Frames (V)

```
redFrame.pack(side=LEFT)
topYellow = Frame(root)

green = Frame(root)
redFrame = Frame(green)
redFrame.pack(side=LEFT)
brownFrame = Frame(green)

bottomYellow = Frame(root)

brownFrame.pack(side=LEFT)

topYellow.pack(side=TOP)

green.pack(side=TOP)

bottomYellow.pack(side=TOP)

l = Label(topYellow, text="Encabezado")
l.pack()
l = Label(bottomYellow, text="Pié")
l.pack()
addButton(redFrame, TOP)
```

