UNIVERSIDAD DE MURCIA

Interfaces de Usuario

L. Daniel Hernández < *ldaniel@um.es* >

Dpto. Ingeniería de la Información y las Comunicaciones
Universidad de Murcia
4 de diciembre de 2023

[1]

[1] Imagen: https://www.ntu.edu.sg/home/ehchua/programming/java/J4a_GUI.html

Índice de Contenidos

Desarrollo

GUIs en Python

- No hay que confundir las herramientas de GUI con herramientas de gráficos o herramientas de visualización de datos.
- **Tkinter** es un *binding* de Python al kit de herramientas de la GUI de Tk.
- **Tk** es la biblioteca GUI original para el lenguaje Tcl.
- Tkinter se implementa como un envoltorio (*wrapper*) de Python alrededor de un intérprete de Tcl completo incrustado en el intérprete de Python.
- Otros lenguajes que usan Tk son: Tcl, Ruby, Perl.
- Hay **otras librerías** de GUI para Python: wxPython, PyQt y PyGTK.
- Python ya tiene construidas algunas GUI por nosotros.
- Lo normal es que construyamos también nuestra propia GUI.

Desarrollo

Conceptos Tk

https://docs.python.org/3.9/library/tkinter.html#tkinter.Tk

- **Widget:**
  - Es cualquier objeto que represente a un elemento de la ventana (botones, frames, ...)
  - Crear un widget es crear un objeto de una clase específica.
  - Todos los widget forman parte de un árbol jerárquico de widget.

```
root = Tk()  #  root es la ventana principal
root.title("Título de la ventana")
# root.resizable(True, False)
# root.geometry("720x350")
# Crear árbol jerárquico
root.mainloop()
```

- **Geometry Manager:**
  - Es el encargado de indicar exactamente dónde se colocará cada widget.
  - Trabaja con widgets master (maestro) y slave (esclavo)
  - Un widget maestro suele hacer referencia a una ventana o marco que contiene widgets esclavos.
- **Event Handling:**
  - Tk ejecuta un bucle de eventos que recibe eventos del sistema operativo (pulsaciones de teclas, movimiento del ratón, cambio de tamaño en la ventana, etc).
  - Cuando se recibe un evento, el widget afectado invocará a un comando.

```
tk.Button(mainframe, text="Calculate", command=calculate)
```

Variables de Control

- Las variables de control son objetos que se asocian a los widgets.
  - Estas variables almacenan valores del widget para que estén disponibles en otras partes del programa.
  - Cuando cambian sus valores queda reflejado de forma automática en el widget.
- Hay cuatro tipos:

```
entero   = IntVar()      # Para trabajar con enteros
real     = DoubleVar()   # Para usar números de tipo flotante
cadena   = StringVar()   # Para usar cadenas de caracteres
booleano = BooleanVar()  # Para trabajar con booleanos.
```

- Métodos:
  - `.set(valor)`, asigna un valor a la variable de control.
  - `.get()`, retorna el valor de la variable de control.
  - `.trace(tipo, función)`, se emplea para detectar cuándo la variable es leída, cambia su valor o es borrada.
    - `tipo`, indica el tipo de suceso a comprobar: 'r', lectura de variable; 'w', escritura de variable, 'u', borrado de variable.
    - `función` indica la función que será llamada cuando ocurra el suceso.

Construcción de Variables de Control y Ejemplo de Uso

- Constructor: `var = tk.xxxVar (contenedor, valor, nombre)`
  - Son 3 parámetros opcionales.
  - `contenedor`: es un widget asociado con el objeto xxxVar. Si se omite el contenedor, el valor predeterminado es la ventana raíz.
  - `valor`: es el valor inicial que por defecto es una cadena vacía `''`.
  - `nombre`: es un nombre de Tcl que por defecto es `PY_VARnum`.
- Un ejemplo de uso[2]:
  - `StringVar` administra el valor de un widget, como una etiqueta o una entrada.
  - Después de crear el objeto StringVar, puedes **asignarlo a la variable de texto**, `text`, de un widget que acepta un objeto StringVar.
  - `string_var.get()` retorna el valor actual del `string_var`

  Ejemplo:
  - `Label(parent, text=string_var.get())`
  - `Entry(parent, text=string_var)`

Desarrollo

Tkinter Dialogs

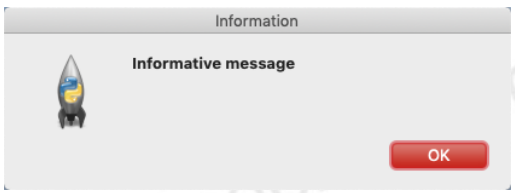
https://docs.python.org/3.9/library/tkinter.messagebox.html

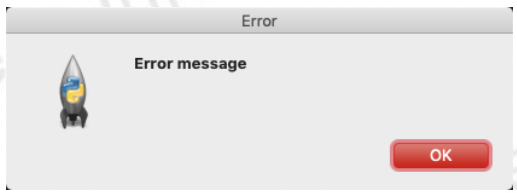
- El módulo `tkinter.messagebox` crea ventanas de mensaje modales.
- messagebox.`Message(master=None, **options)` crea una ventana de mensaje de información por defecto.
- Pero se pueden concretar aún más:
  - Mensaje de información:
    - messagebox.`showinfo()`
  - Mensajes de aviso
    - messagebox.`showwarning()`
    - messagebox.`showerror()`
  - Mensajes de preguntas
    - messagebox.`askquestion()`
    - messagebox.`askokcancel()`
    - messagebox.`askretrycancel()`
    - messagebox.`askyesno()`
    - messagebox.`askyesnocancel()`

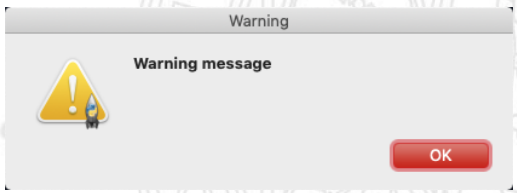
Profundizamos en ...

messagebox para mostrar información

```
from tkinter import messagebox

messagebox.showinfo("Information", "Informative message")
messagebox.showerror("Error", "Error message")
messagebox.showwarning("Warning", "Warning message")
```



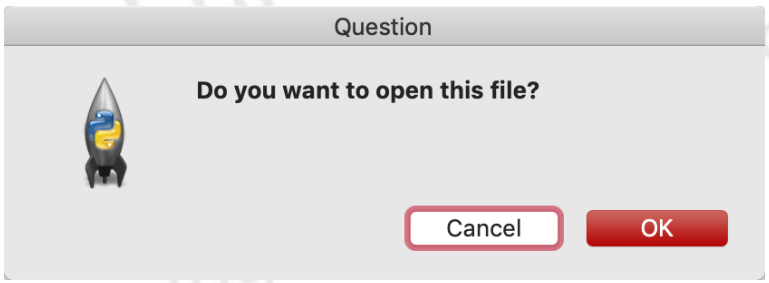


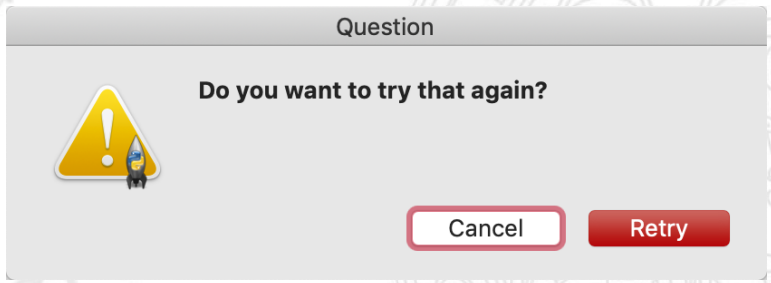


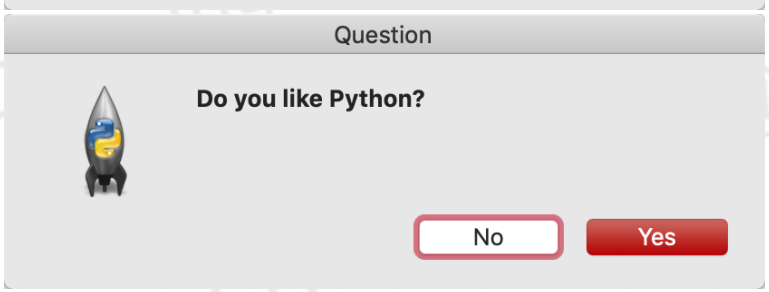
Profundizamos en ...

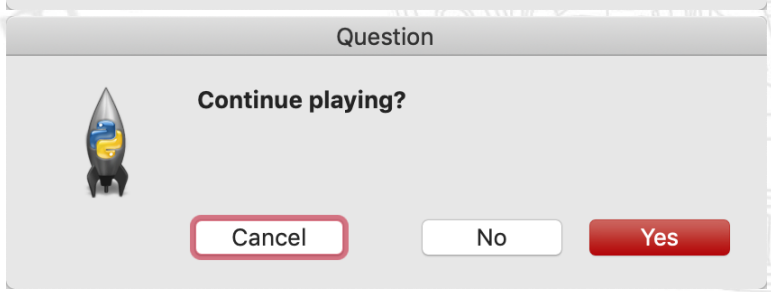
messagebox para pedir información

```
from tkinter import messagebox
answer = messagebox.askokcancel("Question", "Do you want to open this file?")
# False, True
answer = messagebox.askretrycancel("Question", "Do you want to try that again?")
# False, True
answer = messagebox.askyesno("Question", "Do you like Python?")
# False, True
answer = messagebox.askyesnocancel("Question", "Continue playing?")
# None, False, True
```









Desarrollo

Tkinter Dialogs

https://docs.python.org/3.9/library/dialog.html

tkinter.filedialog File selection dialogs

- Proporcionan **ventanas de diálogo para archivos** que combinan una apariencia nativa con opciones de configuración para personalizar el comportamiento.
- Los siguientes argumentos son aplicables a las clases y funciones de este módulo:
  - parent: la ventana para colocar el diálogo encima de
  - title: el título de la ventana
  - initialdir: el directorio en el que comienza el cuadro de diálogo
  - initialfile: el archivo seleccionado al abrir el cuadro de diálogo
  - filetypes: una secuencia de tuplas (etiqueta, patrón), se permite el comodín "*"
  - defaultextension: extensión predeterminada para agregar al archivo (guardar cuadros de diálogo)
  - multiple: cuando es verdadero, se permite la selección de varios elementos

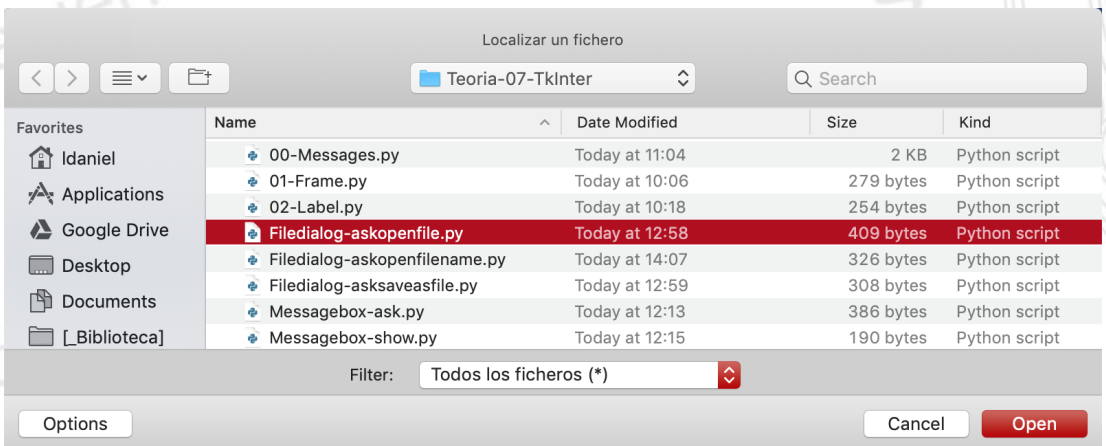
tkinter.simpledialog Standard Tkinter input dialogs

- Contiene clases y funciones adecuadas para crear **diálogos simples** para obtener un valor del usuario.
- No olvides indicar a la ventana padre.

Profundizamos en ...

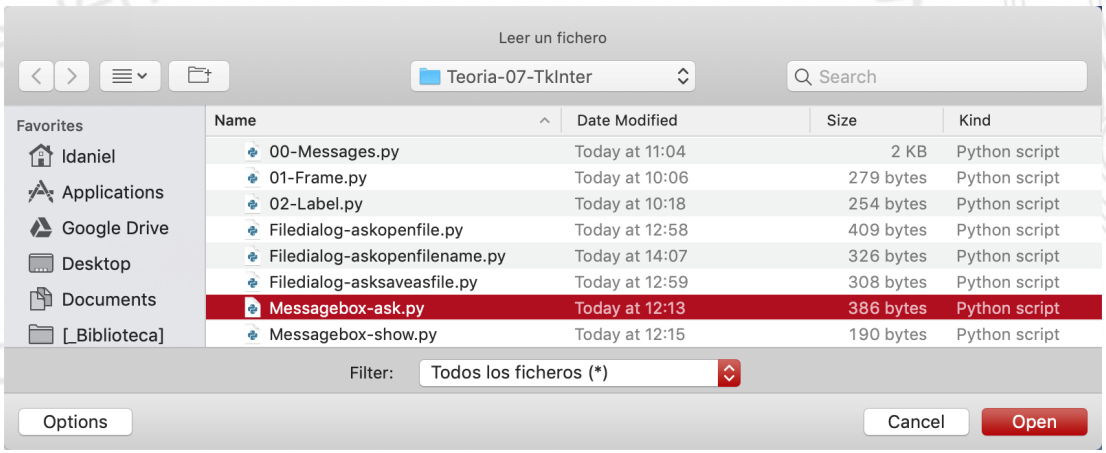
filedialog para Localizar un Fichero

```
from tkinter import filedialog

fichero = filedialog.askopenfilename(
    initialdir=".",
    filetypes=(("Ficheros de texto", "*.txt"),("Todos los ficheros", "*.*")),
    title="Localizar un fichero"
    )

lineas = "Nada que imprimir"
if fichero:
    with open(fichero, "r") as f:
        lineas = f.readlines()

print(lineas)
```



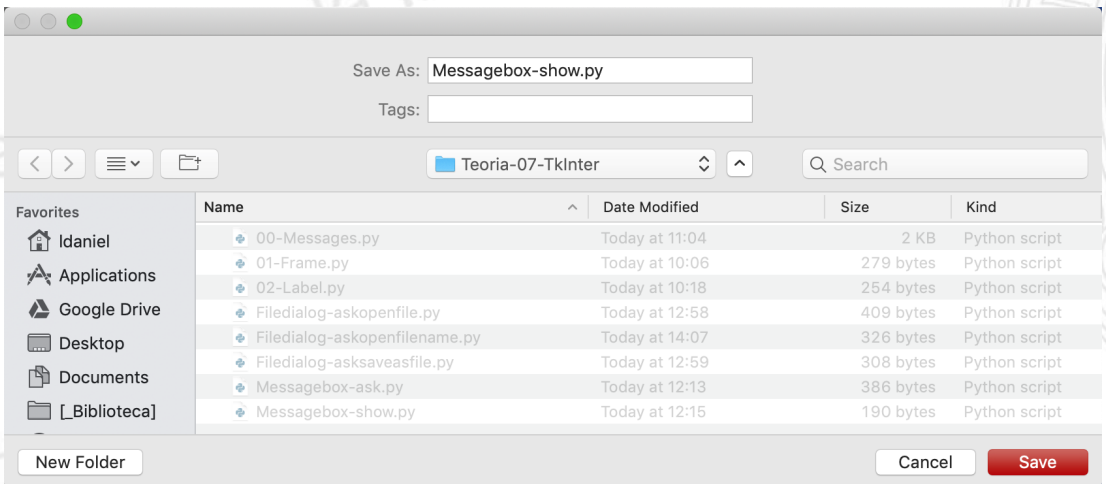
filedialog para Abrir un Fichero

```
from tkinter import filedialog

fichero = filedialog.askopenfile(mode="r",
    initialdir=".",
    filetypes=(("Ficheros de texto", "*.txt"), ("Todos los ficheros", "*.*")),
    title="Leer un fichero"
)

lineas = "Nada que leer"
if fichero is not None:
    lineas = fichero.readlines()
    fichero.close()

print(lineas)
```



filedialog para Escribir un Fichero

```
from tkinter import filedialog

fichero = filedialog.askopenfile(mode="w",
    initialdir=".",
    title="Guardar en un fichero"
)

lineas = ["Un par\n", "de lineas\n"]
if fichero is not None:
    fichero.writelines(lineas)
    fichero.close()
```



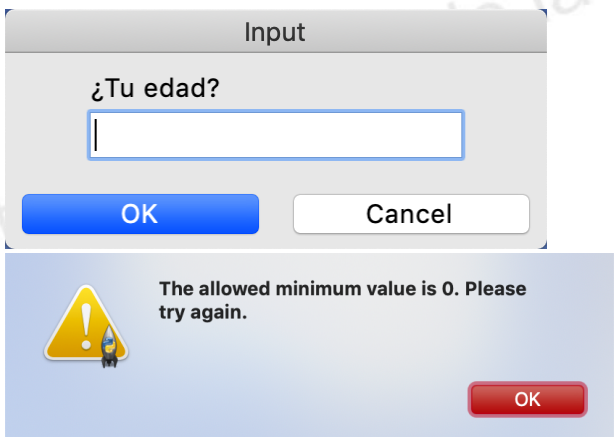
Profundizamos en ...

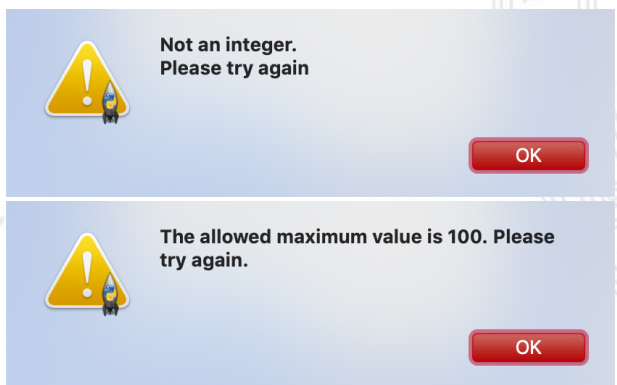
Tres formas de pedir datos con simpledialog

```
import tkinter as tk
from tkinter import simpledialog
window = tk.Tk()  # NO olvides la ventana padre
answer = simpledialog.askstring("Input", "Tu nombre?", parent=window)
answer = simpledialog.askinteger("Input", "Tu edad?",  parent=window,
                                 minvalue=0, maxvalue=100)
answer = simpledialog.askfloat("Input", "Tu sueldo?",  parent=window,
                               minvalue=0.0, maxvalue=100000.0)
```

- Si al pedir un dato no cumple las restricciones se mostrará un ventana de aviso.

  Ejemplo: Ventanas de aviso al pedir un entero





Ejercicio

Ejercicio.

Construye un programa que haga lo siguiente, en este orden:

1. Solicita el nombre y edad de dos personas.

- La longitud del nombre no puede ser vacío y tienen que ser al menos de 3 caracteres.
- La edad mínima será 16.
- La información de cada persona es una tupla.

2. Guarda la información en un fichero.

- Pide al usuario que te indique el nombre del fichero.
- Los datos de cada persona ocupan una línea.

3. Muestra un mensaje informativo de que toda la tarea se ha realizado.

Desarrollo

Widgets

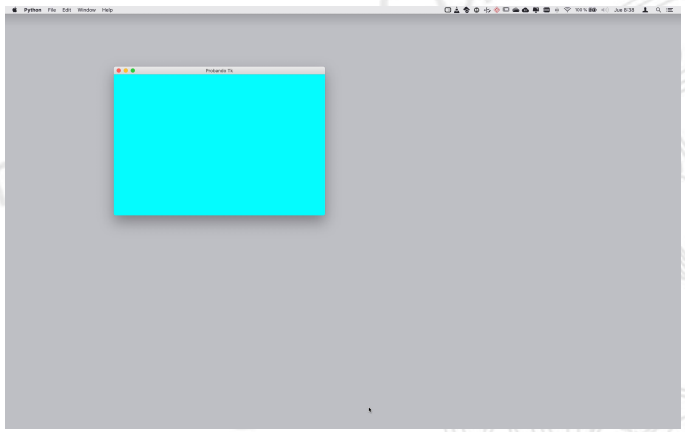
- Widgets Básicos:
  - Frame: Contenedor de widgets
  - Label: Etiquetas (de texto o gráfica)
  - Buttom: Botón
  - Checkbutton: Seleccionar o no una opción. Pueden ser varias.
  - Radiobutton: Seleccionar una opción.
  - Entry: Introducir un texto
- Otros widgets (hay más):
  - Listbox: Seleccionar una o varias entradas de una lista de opciones.
  - Combobox: Seleccionar una entrada de una lista de opciones.
  - Text: Introducir varias líneas de texto.
  - Scrollbar: Barra de desplazamiento cuando el contenido es demasiado grande.
  - Scale: Barra de desplazamiento para seleccionar un valor numéricos.
  - Spinbox: Seleccionar un número (o de una lista arbitraria) con flecha arriba/abajo.
  - Progressbar: Da al usuario feedback del progreso de la operación.
  - Canvas: Dibuja líneas, círculos, texto, imágenes, ...
  - Treeview: Muestra una jerarquía de items y permite que el usuario los visualice.
  - Separator: Muestra una línea de separación.
  - Notebook: Pestañas
- Objetos que no son widgets: menús, ventanas (de diálogo, de directorios, de ficheros, de colores).

Profundizamos en ...

Creación de Ventanas: Tk()

- El sistema de coordenadas considera el origen (0, 0) en la esquina superior izquierda.
- Sobre una ventana se debe de crear uno o varios Frames (widget).
- Sobre cada Frame se colocarán otros widgets (botones, etiquetas, ...)

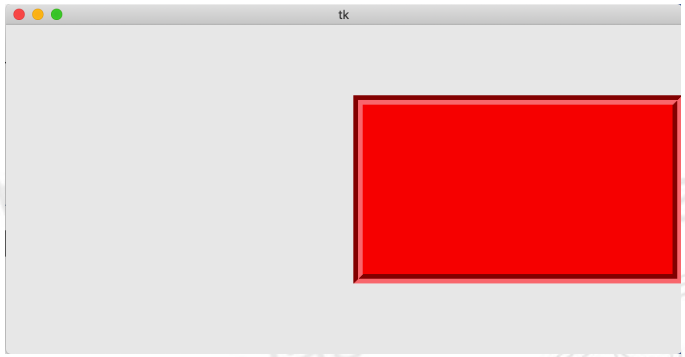
```
import tkinter as tk

# Crea Ventana Principal
root = tk.Tk()

# Asigna un título
root.title("Probando Tk")

# Se pueden retocar parámetros en su configuración
root.config(bg="Aqua")

# Ejecución de un bucle sin fin con eventos
root.mainloop()
```



Widgets Básicos: Frame

`frame = tk.Frame(root)`

```
import tkinter as tk

root = tk.Tk()
root.geometry("720x350")

frame = tk.Frame(root)
frame.config(width="350", height="200")
frame.config(borderwidth='10', relief='groove')
frame.config(bg="red")

frame.pack(side="right")  # 'left', 'right', 'top', 'bottom'

root.mainloop()
```



Relief: FLAT RAISED SUNKEN GROOVE RIDGE

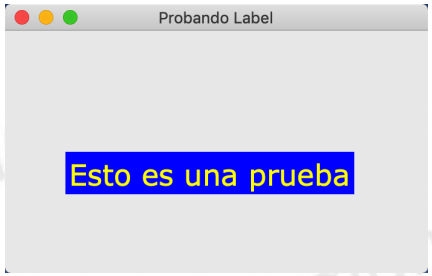
- Para `pack()` ver pág. 51.

Profundizamos en ...

Widgets Básicos: Label

`label = tk.Label(frame, text="txt")` con `place()`

```
root = tk.Tk()
root.title("Probando Label")


frame = tk.Frame(root, width=350,
                                    height=200)
frame.pack()


label = tk.Label(frame, text="Esto es una prueba")
label.place(x=50, y=100)
label.config(fg="yellow", bg="blue")
label.config(font=("Verdana", 24))


root.mainloop()
```

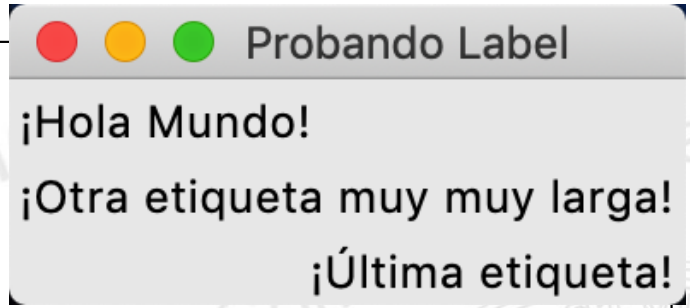


- Para `place()` ver pág. 50.

Widgets Básicos: Label

`label = tk.Label(frame, text="txt")` con `pack()`

```
root = tk.Tk()
root.title("Probando Label")


frame = tk.Frame(root)
frame.pack()

tk.Label(frame, text="!Hola Mundo!").pack(anchor='nw')
tk.Label(frame, text="!Otra etiqueta muy muy larga!").
                                        pack(anchor='center')
tk.Label(frame, text="!Última etiqueta!").pack(anchor='se')

root.mainloop()
```

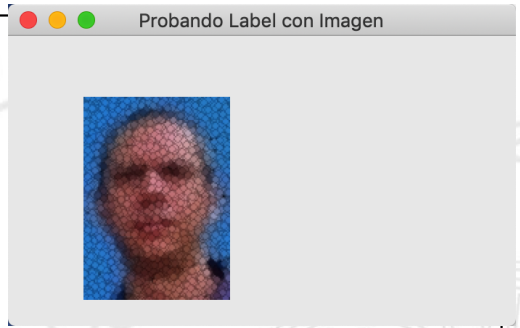


- Para `pack()` ver pág. 51.

Widgets Básicos: Label

`label = tk.Label(frame, img=un_imagen)`

```
root = tk.Tk()
root.title("Probando Label con Imagen")
root.geometry("350x200+100+100")


frame = tk.Frame(root, width=350, height=200)
frame.pack()

mi_imagen=tk.PhotoImage(file="./ldaniel.png")

label = tk.Label(frame, image=mi_imagen)
label.place(x=50, y=40)

root.mainloop()
```



- Para `place()` ver pág. 50.

Label con un StringVar cualquiera

```
root = tk.Tk()
contador = 0
root.title("Label con StringVar")
frame = tk.Frame(root, width=150, height=100).pack()
label = tk.Label(frame).place(x=50, y=25)
label.config(font=("Verdana", 24))

# El StringVar es un objeto libre
string_var = StringVar(value="Mensaje Inicial")

while True:
    contador += 1
    root.update_idletasks()
    root.update()
    time.sleep(0.1)
    # ESTO ES LO IMPORTANTE
    string_var.set(str(contador))
    # ^^^^^ En cada iteración se asigna nuevo valor a string_var
    label.config(text=string_var.get())  # <<<<
    # ^^^^^ En cada iteración reconfiguro Label
```

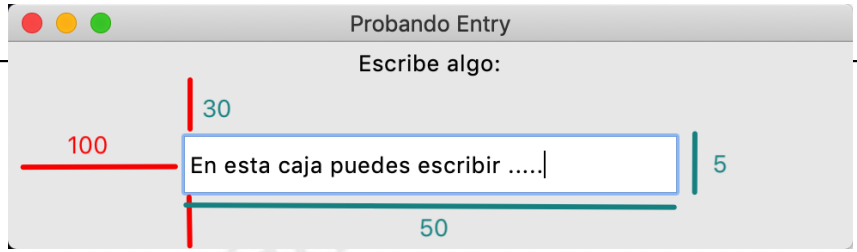
Vinculando un Label a un StringVar: textvariable

```
contador=0
root = tk.Tk()
root.title("Label con StringVar")
frame = tk.Frame(root, width=150, height=100).pack()

# Vinculamos el StringVar al Label
string_var=StringVar(value="Mensaje Inicial")
label = tk.Label(frame, textvariable=string_var) # <<<< Vinculación
label.place(x=50, y=25)
label.config(font=("Verdana", 24))

while True:
    contador += 1
    root.update_idletasks()
    root.update()
    time.sleep(0.1)
    # Basta cambiar string_var para que cambie Label
    string_var.set(str(contador))
    # ^^^^^ En cada iteración se asigna nuevo valor a string_var
    #       No se necesita reconfigurar Label
```

Profundizamos en ...

Widgets Básicos: Entry (texto corto)

`entry = tk.Entry(frame)`



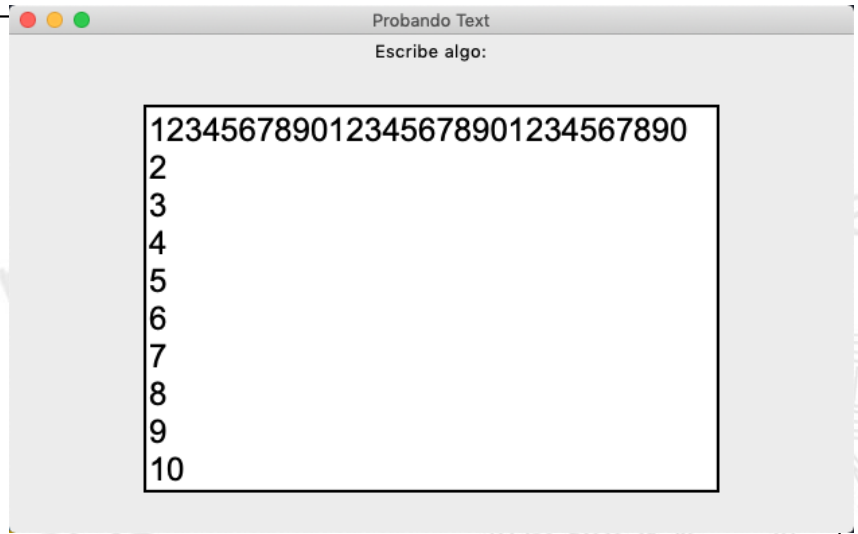

```
import tkinter as tk

root = tk.Tk()
root.title("Probando Entry")

frame = tk.Frame(root, width=350, height=200)
frame.pack()

tk.Label(frame, text="Escribe algo:").pack(padx=10)


tk.Entry(frame).pack(padx=100, pady=30, ipadx=50, ipady=5)


root.mainloop()
```

- Para `pack()` ver pág. 51.

Widgets Básicos: Text (texto largo)

`text = tk.Text(frame)`

```
import tkinter as tk

root = tk.Tk()
root.title("Probando Entry")
frame = tk.Frame(root,
                 width=350,
                 height=200)
frame.pack()


tk.Label(frame, text="Escribe algo:").pack()

text = tk.Text(frame)
text.config(width=30, height=10, font=("Arial", 24))
text.pack(padx=100, pady=30)

root.mainloop()
```



- Para `pack()` ver pág. 51.

¿Cómo deshabilitar la edición?

- Si alguna vez quisiera deshabilitar que se modifique una entrada puede usar alguna de estas opciones.
- Cambiando el estado de habilitación/deshabilitación.

```
text = Text(app, state='disabled', width=44, height=5)
text.configure(state='normal')
text.insert('end', 'Some Text')
text.configure(state='disabled') # Deshabilita el widget
```

- Indicando que el estado es de solo lectura.

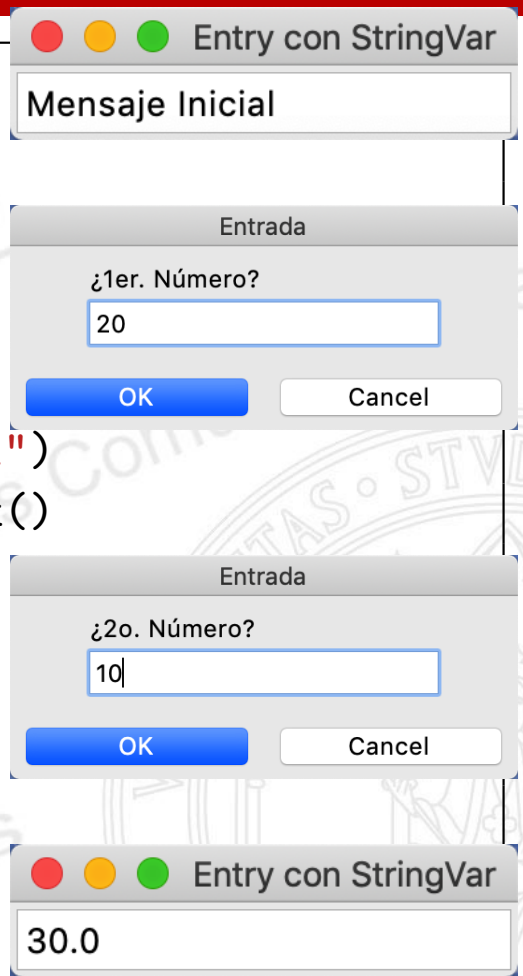
```
entry = Entry(frame)
entry.config(state='readonly')) # Habilitado pero no editable
```

- Vincular cualquier pulsación de tecla a una función que devuelva "break":

```
text = Text(app, state='normal', width=44, height=5)
text = Tkinter.Text(root)
text.bind("<Key>", lambda e: "break")
```

Entry con StringVar

```
root = tk.Tk()
root.title("Entry con StringVar")
frame = tk.Frame(root, width=150, height=100)
frame.pack()

# Vinculamos el StringVar al Entry
string_var = StringVar(value="Mensaje Inicial")
tk.Entry(frame, textvariable=string_var).pack()

n1 = simpledialog.askfloat("Entrada",
                           "1er. Número?",
                            parent=root)
n2 = simpledialog.askfloat("Entrada",
                           "2o. Número?",
                           parent=root)

#Se cambia el valor del string_var
string_var.set(str(n1+n2))       # Muestra el resultado
entry.config(state='readonly')   # y no se puede modificar
root.mainloop()
```



Profundizamos en ...

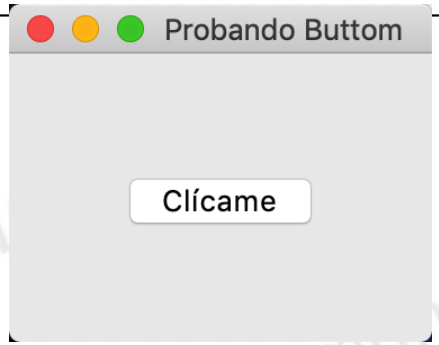
Eventos

- Una aplicación Tkinter pasa la mayor parte de su tiempo dentro de un **bucle de eventos** (ingresado a través del método mainloop).
- Ejemplos de eventos: pulsar una tecla, mover el ratón, dibujar en el canvas, ...
- Tkinter permite que **a cada widget, se le pueda vincular funciones** y métodos de Python a eventos.
- `widget.bind (evento, controlador)`: Si ocurre un evento que coincide con la descripción del evento en el widget, se llama al controlador dado con un objeto que describe el evento.
  Ejemplo: `frame.bind("<Button-1>", callback)` indica que la hacer click con el ratón se invocará a la función `callback()`.
- Estos son algunos nombres de los eventos: `<Button-1>`, `<ButtonPress-1>`, `<B1-Motion>`, `<ButtonRelease-1>`, `<Enter>`, `<Return>`, `<Key>`, etc ...
  Puede consultar los siguientes enlaces para su significado:
  - https://effbot.org/tkinterbook/tkinter-events-and-bindings.htm
  - http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/event-types.html
  - https://www.python-course.eu/tkinter_events_binds.php
- Algunos widget presenta `command` entre sus opciones **para indicar la función o método que será invocado** cuando se produzca ciertos eventos:
  - Para Button, cuando el widget es **pulsado**.
  - Para Checkbutton y Radiobutton, cuando el widget **cambia su estado**.

Widgets Básicos: Button

`bt = tk.Button(parent, text="txt", command=funcion)`

```
import tkinter as tk

def actua():
    global root
    root.destroy()

root = tk.Tk()
root.title("Probando Buttom")

frame = tk.Frame(root, width=350, height=200)
frame.pack()

button = tk.Button(frame, text='Clícame', command=actua)
button.pack(padx=50, pady=50)

root.mainloop()
```

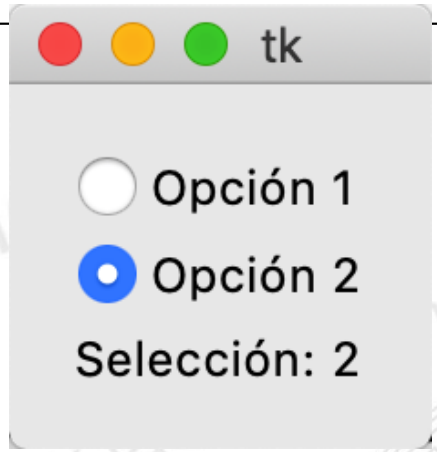


- Al pulsar el botón se invocará al método `root.destroy()`, el cual terminará la aplicación.

Widgets Básicos: Radiobutton

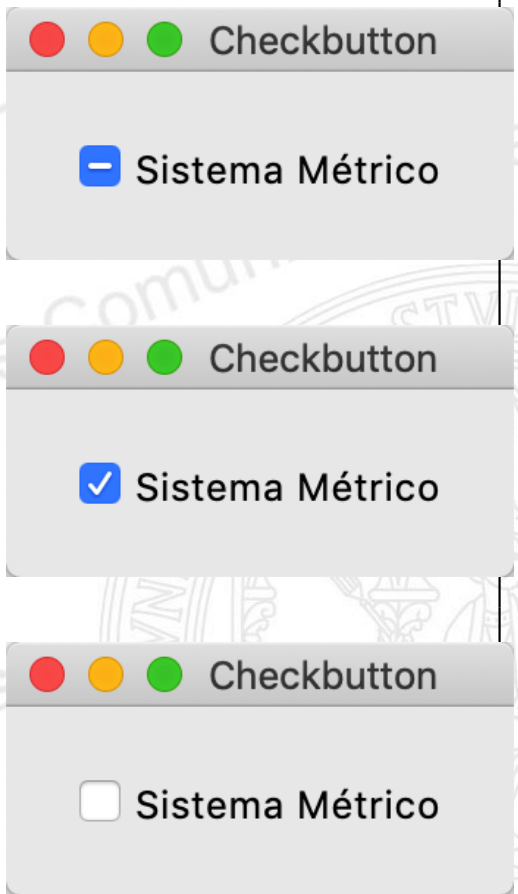
`bt = tk.Radiobutton(parent, text="txt", variable=var, value=v, command=func)`

```
import tkinter as tk
from tkinter import IntVar

def selec():
    t = f"Selección: {opcion.get()}"
    monitor.config(text = t )

root = tk.Tk()
root.config(bd=15)

opcion = IntVar() # Como StrinVar pero en entero

tk.Radiobutton(root, text="Opción 1", variable=opcion,
               value=1, command=selec).pack()
tk.Radiobutton(root, text="Opción 2", variable=opcion,
               value=2, command=selec).pack()

salida = tk.Label(root)
salida.pack()
root.mainloop()
```



Widgets Básicos: Radiobutton

```
import tkinter as tk
from tkinter import StringVar

def muestraMedida():
    print(sistemaMedida.get())

root = tk.Tk()
root.title("Checkbutton")
root.config(bd=20)

sistemaMedida = StringVar()

tk.Checkbutton(root, text='Sistema Métrico',
               command=muestraMedida,
               variable=sistemaMedida,
               onvalue='metrico',
               offvalue='ingles')
               .pack()
tk.mainloop()
```

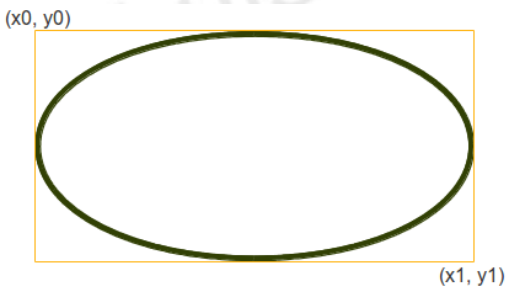


Profundizamos en ...

Canvas

```
canvas = tk.Canvas(parent, width= , height= , bg= )
```

- El widget Canvas proporciona una superficie sobre la que se puede dibujar, mostrar imágenes, texto, etc.
- El origen de coordenadas se sitúa en la parte superior izquierda.
- Colocado el canvas en la ventana se usan los métodos `create_xxx()`.
  - `canvas.create_rectangle(50, 20, 150, 80, fill="#476042")`
  - `canvas.create_line(0, 0, 50, 20, fill="#476042", width=3)`
  - `canvas.create_oval(x0, y0, x1, y1, opciones, ... )`




```
def circle(canvas, x, y, r):
    id = canvas.create_oval(x-r,y-r,x+r,y+r)
    return id
```

  - `canvas.create_text(10, 20, text="Python")`
  - `canvas.create_image(10, 20, image=img)` donde `img=PhotImage(file="file.png")`
  - También se puede añadir polígonos.
    Consulte https://python-course.eu/tkinter_canvas.php si quiere un mini tutorial sobre esto.

Profundizamos en ...

Métodos de distribución

- Tk provee tres métodos para establecer la posición de los widgets dentro de una ventana
- Se corresponden con los métodos `.place()`, `.pack()` y `.grid()`.
- `.place()` ubica el elemento indicando su posición absoluta respecto del padre.
- `.pack()` coloca el elemento de forma relativa a los elementos previos.
- `.grid()` divide la ventana en filas y columnas formando celdas. Se coloca el elemento en la celda que se indique.

Método `.place()`

- `.place()` ubica el elemento indicando su posición absoluta respecto del padre.
- `x=60, y=40` coloca el objeto en la posición (60, 40) respecto del origen.
- `relx=0.1, rely=0.1` Indica la posición relativa al tamaño de la ventana.
  - Toma valores entre 0 y 1.
  - Si el ancho de la ventana son 300px, `relx=0.1` indica que se colocará en la posición `x=30`$=0.1\times 300$.
- `width=100, height=30` Indica el ancho y alto en píxeles del objeto.
- `relwidth=0.5, relheight=0.5` Indica el tamaño del objeto relativo al tamaño de la ventana.
  - Toma valores entre 0 y 1.
  - 0.5 indica que será la mitad del tamaño de la ventana.
- Tiene ejemplos de uso en las páginas anteriores.

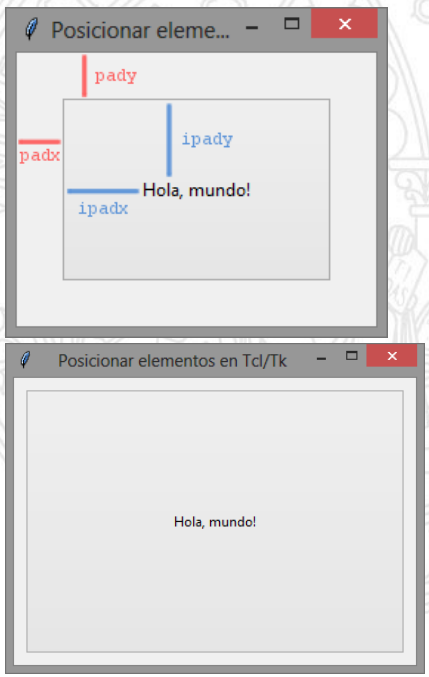
Método .place()

```
import tkinter as tk

root = tk.Tk()
root.geometry("300x100+200+200")
root.title("Label")


button1=tk.Button(root, text="Pos. absoluta")
button1.place(x=10, y = 10, width=290, height=50)

button2=tk.Button(root, text="Pos. realativa")
button2.place(relx=.5, rely=.5,
                       relwidth=.5, relheight=.5)

tk.mainloop()
```

Método .pack()

- `.pack()` coloca el elemento de forma relativa a los elementos previos.
- `.pack()` sin argumentos ubica al elemento debajo del anterior.
- `side=tk.xxx` coloca el widget en el lado que se indique.
  - Sus valores son: `tk.TOP` (por defecto),`tk.BOTTOM`, `tk.LEFT`, `tk.RIGHT`.
- `padx`, `pady` especifican (en píxeles) los márgenes externos de un elemento.
- `ipadx`, `ipady` que especifican (en píxeles) los márgenes internos de un elemento.

```
button = tk.Button(frame, text="Hola, mundo!")
button.pack(padx=30, pady=30, ipadx=50, ipady=50)
```



- Se puede expandir o contraer el elemento a medida que cambie la ventana (`expand=True`), indicando la dirección (`fill=tk.X`, `fill=tk.Y` o `fill=tk.BOTH`)

```
button.pack(expand=True, fill=tk.BOTH,
                         padx=10, pady=10)
```

- Tiene ejemplos de uso en las páginas anteriores.

Fuente: https://recursospython.com/guias-y-manuales/posicionar-elementos-en-tkinter/

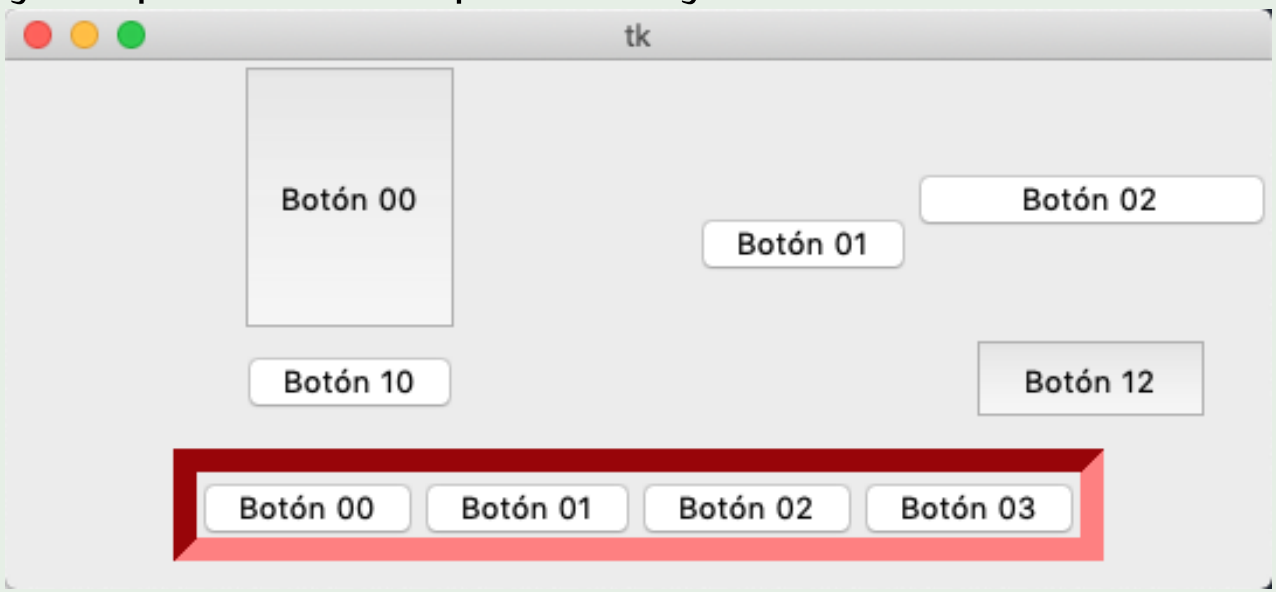
Método `.grid()`

- `.grid()` divide la ventana en filas y columnas formando celdas.
- El elemento se coloca en la celda/coordenadas que se indique.
- Las **coordenadas** se especifican con `row=posY`, `column=posX`.
- Dentro de la celda el elemento **se puede posicionar** en una zona concreta con `sticky="lugar"`. Las opciones son `"n"`(norte), `"s"`(sur), `"e"`(este) o `"w"`(oeste).
- Si se indican varias posiciones, el widget se expandirá.
  - Ejemplo, `sticky="nsew"` expande el elemento en todas las direcciones.
- Por defecto las columnas/filas no se expanden/contraen si la ventana cambia su tamaño.
  - `rowconfigure(fila, weight=1)` indica la fila que puede expandirse.
  - `columnconfigure(col, weight=1)` indica la columna que puede expandirse.
  - Cambiando el valor de `weight` indicará en que proporción debe expandirse una fila/columna frente a las demás.
- `columnspan=numColumnas` El widget ocupará varias **columnas** desde una posición.
- `rowspan=numFilas` El widget ocupará **varias filas** desde una posición.
- Al igual que `.pack()` admite márgenes externos e internos:
  - `padx`, `pady` especifican (en píxeles) los márgenes externos de un elemento.
  - `ipadx`, `ipady` que especifican (en píxeles) los márgenes internos de un elemento.

Ejercicio

Ejercicio.

¿Qué código se podría escribir para conseguir esta ventana?



Ejercicio

Ejercicio.

Cuál sería la secuencia de instrucciones que permite:

1. Solicitar el nombre (string) y edad (entero) de n–personas.
2. Guardar los nombres y edades en un fichero.

Teniendo en cuenta que:

1. Será obligatorio introducir un string y un entero por persona usando una ventana modal.
2. Se guardarán los datos usando una ventana de diálogo, obligando a introducir el nombre de dicho fichero.

No es necesario crear clases, solo escribir la secuencia de instrucciones.

Ejercicio.

Construye una calculadora simple: consta de dos entradas donde se introducen dos valores reales y un botón que al ser pulsado lee los valores de ambos campos para volcar el resultado en otro campo (no editable).
Escribe primero la secuencia de instrucciones y posteriormente escribe la versión en la que la calculadora es una clase hija de Tkinter.

Desarrollo

Referencias

Algunos lugares de referencia **para consultar** sobreTkinter:

- https://python-course.eu/python_tkinter.php
- https://docs.python.org/3.9/library/tkinter.html
- https://zetcode.com/tkinter/
- https://python-para-impacientes.blogspot.com/2016/02/variables-de-control-en-tkinter.html
- https://tkdocs.com/tutorial/widgets.html
- https://docs.hektorprofe.net/python/interfaces-graficas-con-tkinter/
- https://youtube.com/playlist?list=PLU8oAlHdN5BlvPxziopYZRd55pdqFwkeS
- http://www.java2s.com/Code/Python/GUI-Tk/CatalogGUI-Tk.htm