

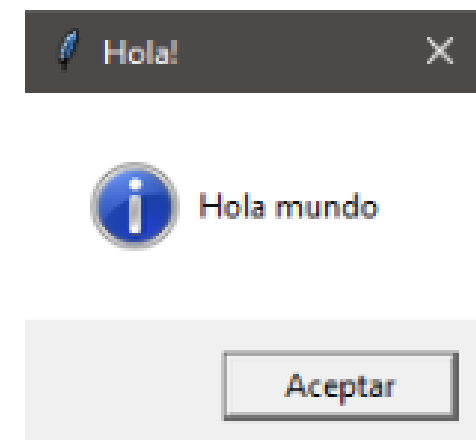
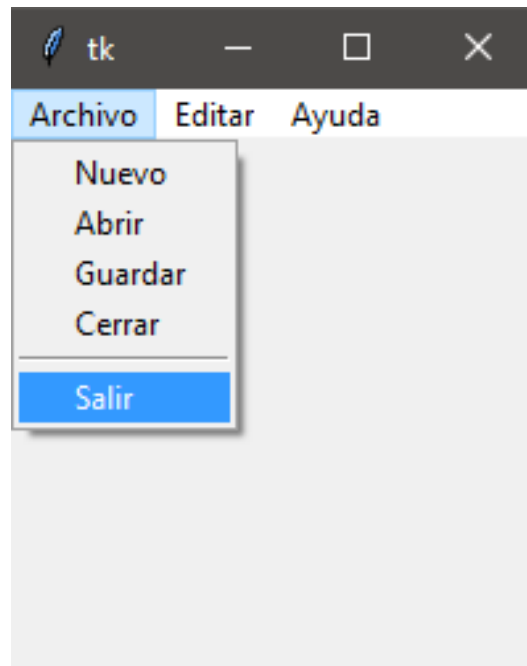
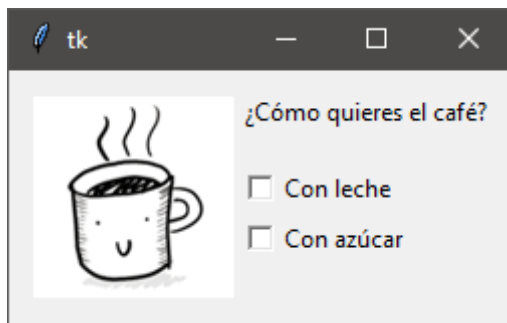
tkinter (... continuación)

Radio button

- ☐ Radio button
- ☒ Radio button 2

Checkbox

- ☒ Styled checkbox (Checked)
- ☐ Styled checkbox
- ☐ Styled disabled checkbox



Más widgets ...

Los widgets que veremos en esta introducción son :

Radiobutton: Botón radial que se usa en conjunto donde es posible marcar una opción.

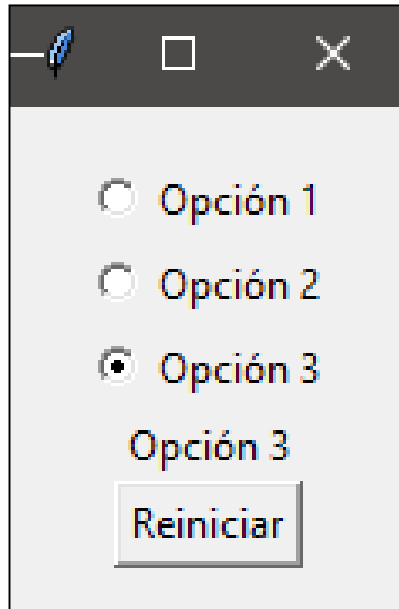
Checkbutton: Botón cuadrado que se puede marcar con un tic.

Menu: Estructura de botones centrados en la composición de menús superiores.

Dialogs: Ventanas emergentes que permiten desde mostrar información al usuario (típico mensaje de alerta o de confirmación) hasta ofrecer una forma gráfica de interactuar con el sistema operativo (seleccionar un fichero de un directorio para abrirlo).

Hay otros widgets, pero estos son los más importantes.

Radiobutton – Botón circular



Un botón circular representa una lista de opciones de las cuales solo puedes seleccionar una.

Si marcas una opción y luego haces click en otra, la primera casilla que has marcado no se recuerda.

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

- Internamente son varias casillas que apuntan a una misma variable. Cada una de las opciones se guarda como un valor numérico (1,2,3...etc.).
- Al marcar cualquiera de ellas se llama a la misma función pero entra con un valor diferente.

Probando como se traduce un botón en un valor numérico

```
from tkinter import *
```

```
def selec():  
    quetengo.config(text = "Opción {}".format(opcion.get() ) )
```

```
#----- Programa principal
```

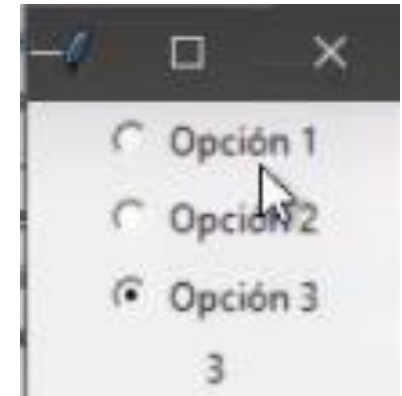
```
root = Tk()  
root.config(bd=15)
```

```
opcion = IntVar()          # Como StringVar pero en entero
```

```
Radiobutton(root, text="Opción 1", variable=opcion, value=1, command=selec).pack()  
Radiobutton(root, text="Opción 2", variable=opcion, value=2, command=selec).pack()  
Radiobutton(root, text="Opción 3", variable=opcion, value=3, command=selec).pack()
```

```
quetengo = Label(root)  
quetengo.pack()
```

```
root.mainloop()
```



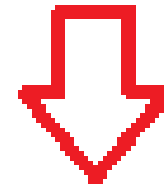
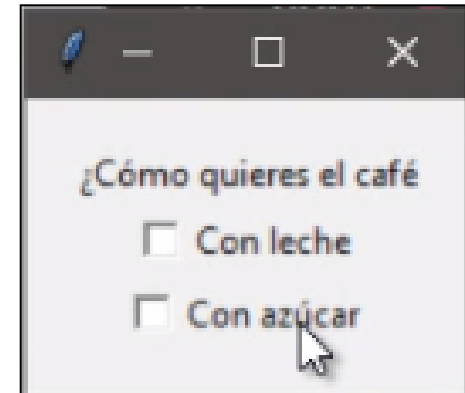
Checkbox – Botón de selección

```
from tkinter import *
```

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

```
leche = IntVar()    # 1 si, 0 no  
azucar = IntVar()  # 1 si, 0 no
```

```
Label(root, text="¿Cómo quieres el café?").pack()  
Checkbox(root, text="Con leche", variable=leche, onvalue=1, offvalue=0).pack()  
Checkbox(root, text="Con azúcar", variable=azucar, onvalue=1, offvalue=0).pack()  
  
root.mainloop()
```



En el siguiente diapositiva se detalla el texto para añadir una foto, y también un texto que indique lo que se ha señalado.

```
Checkbox(root, text="Con leche", variable=leche, onvalue=1, offvalue=0).pack()  
Checkbox(root, text="Con azúcar", variable=azucar, onvalue=1, offvalue=0).pack()
```



También añadimos un dibujo, justo después de las IntVar, y también creamos un frame para poder centrar el resto de los componentes a la derecha

```
imagen = PhotoImage(file="./img/imagen.gif")  
Label(root, image=imagen).pack(side=LEFT)  
frame = Frame(root).pack(side=RIGHT)
```

Reasignamos los controles al Frame y completamos los Check buttons el command que apunta a la función seleccionar.

Ahora el frame de los checks se sitúan a la derecha porque la imagen está a la izquierda.

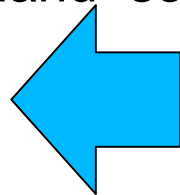
```
frame = Frame(root)
frame.pack(side="left")
```

```
Label(frame, text="¿Cómo quieres el café?").pack(anchor="w")
```

```
Checkbutton(frame, text="Con leche", variable=leche, onvalue=1,
             offvalue=0, command=seleccionar).pack(anchor="w")
```

```
Checkbutton(frame, text="Con azúcar", variable=azucar, onvalue=1,
             offvalue=0, command=seleccionar).pack(anchor="w")
```

```
monitor = Label(frame)
monitor.pack()
```

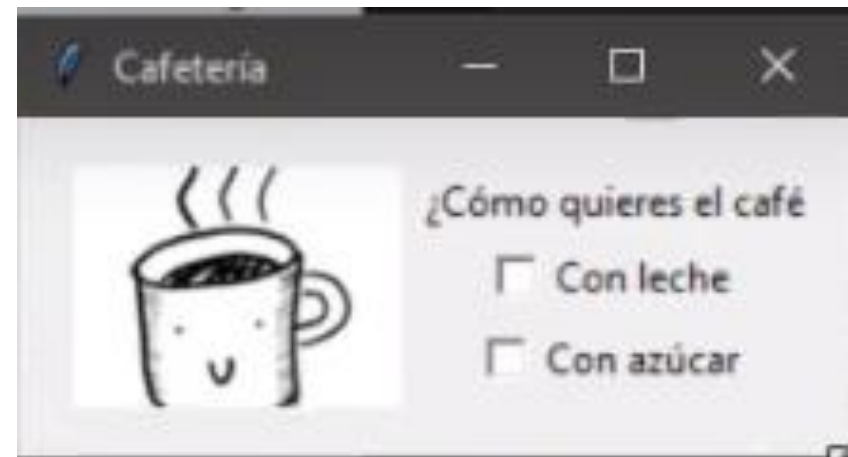


Creamos una variable para
Mostrar en letras la selección
realizada

```
# Finalmente bucle de la aplicación
root.mainloop()
```

```
def seleccionar():  
    cadena = ""  
    if (leche.get()):  
        cadena += "Con leche"  
    else:  
        cadena += "Sin leche"  
  
    if (azucar.get()):  
        cadena += " y con azúcar"  
    else:  
        cadena += " y sin azúcar"  
  
    monitor.config(text=cadena)
```

Añadimos una función que muestra una cadena (monitor) con el detalle de las opciones elegidas.



<https://docs.hektorprofe.net/python/interfaces-graficas-con-tkinter/widget-checkbutton-seleccionable/>

Widget de Menú

En esta lección vamos a aprender a crear un menú superior de toda la vida con varias secciones.

El primer widget menú que creamos hace referencia a la barra de menú, de ahí que se le suele llamar menubar:

menu.py

```
from tkinter import *  
  
root = Tk()  
  
menubar = Menu(root)  
root.config(menu=menubar) # Lo asignamos a la base  
  
root.mainloop()
```

Una vez creada la barra podemos comenzar a añadir submenús y comandos. Empecemos con los submenús:

```
from tkinter import *
```

```
# Configuración de la raíz
```

```
root = Tk()
```

```
menubar = Menu(root)
```

```
root.config(menu=menubar)
```

```
#----- Crea un submenú
```

```
filemenu = Menu(menubar, tearoff=0)
```

```
filemenu.add_command(label="Nuevo")
```

```
filemenu.add_command(label="Abrir")
```

```
filemenu.add_command(label="Guardar")
```

```
filemenu.add_command(label="Cerrar")
```

```
filemenu.add_separator()
```

```
filemenu.add_command(label="Salir", command=root.quit)
```

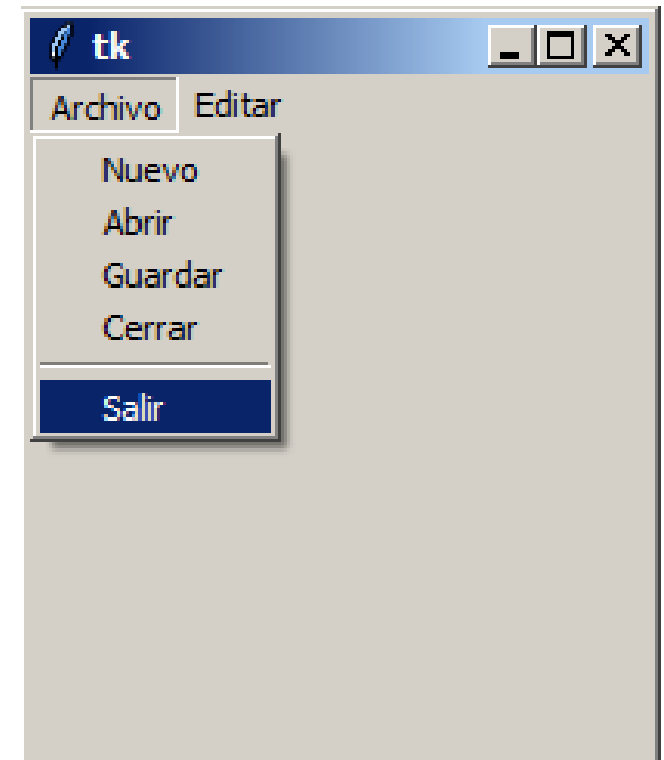
```
#----- Añade los submenús al
```

```
# menú principal
```

```
menubar.add_cascade(label="Archivo", menu=filemenu)
```

```
# Finalmente bucle de la aplicación
```

```
root.mainloop()
```



Puedes continuar
añadiendo opciones

Dialogs (Diálogos)

Las ventanas emergentes, cuadros de diálogo o simplemente Pop Ups, sirven para mostrar o pedir información rápida al usuario. Reciben ese nombre porque no forma parte de la ventana principal, sino que aparecen de golpe encima.

La ventana emergente por excelencia es la MessageBox, que sirve para mostrar un icono y un mensaje, pero tiene algunas variantes. Desde la clásica ventana con la opción de aceptar, la de alerta para informar de excepciones o errores, y las de aceptar o rechazar algo.

Vamos a echar un vistazo a todas ellas.

ShowInfo

Sirve para mostrar un diálogo de más información:

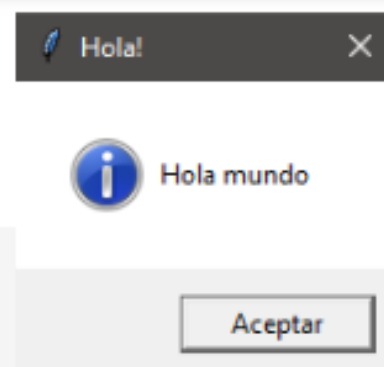
```
from tkinter import *
from tkinter import messagebox as MessageBox

def test():
    MessageBox.showinfo("Hola!", "Hola mundo") # titulo, mensaje

root = Tk()

Button(root, text = "Clicame", command=test).pack()

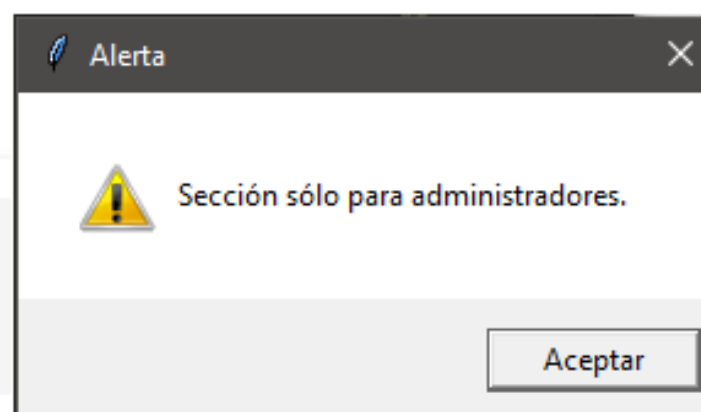
root.mainloop()
```



ShowWarning

Sirve para mostrar un diálogo con un mensaje de alerta:

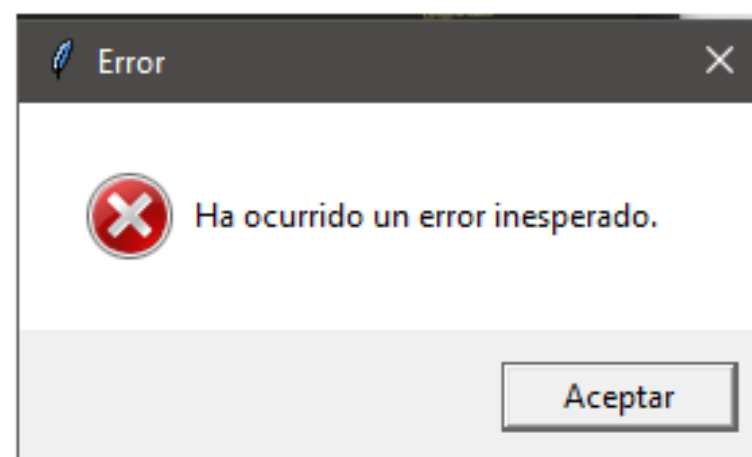
```
MessageBox.showwarning("Alerta",  
    "Sección sólo para administradores.")
```



ShowError

Sirve para mostrar un diálogo con un mensaje de error:

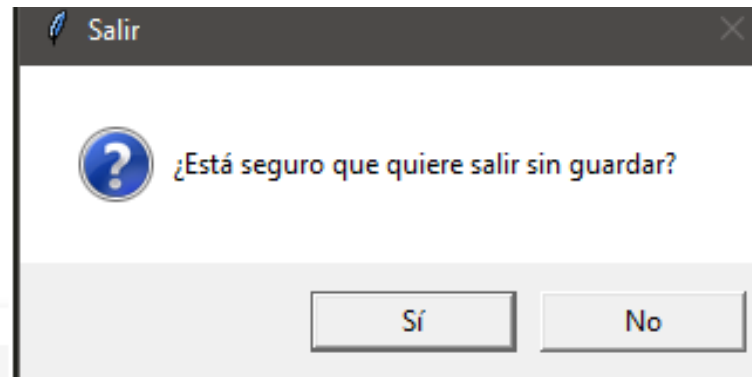
```
MessageBox.showError("Error",  
    "Ha ocurrido un error inesperado.")
```



AskQuestion

Sirve para mostrar un diálogo con una pregunta de Sí/No al usuario:

```
resultado = MessageBox.askquestion("Salir",  
    "¿Está seguro que desea salir sin guardar?")  
  
if resultado == "yes":  
    root.destroy() # Destruir, alternativa a quit
```

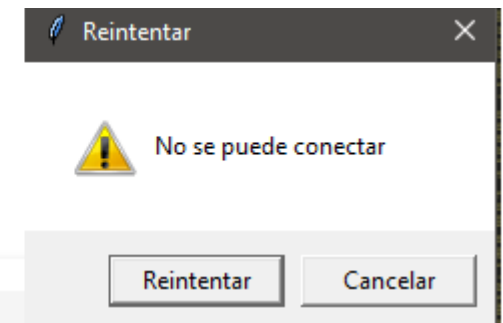


<https://docs.hektorprofe.net/python/interfaces-graficas-con-tkinter/dialogs-dialogos/>

AskRetryCancel

Sirve para mostrar un diálogo con una pregunta de Reintentar/Cancelar al usuario:

```
resultado = MessageBox.askretrycancel("Reintentar",  
    "No se puede conectar")  
  
if resultado == True:  
    # Hacer algo
```

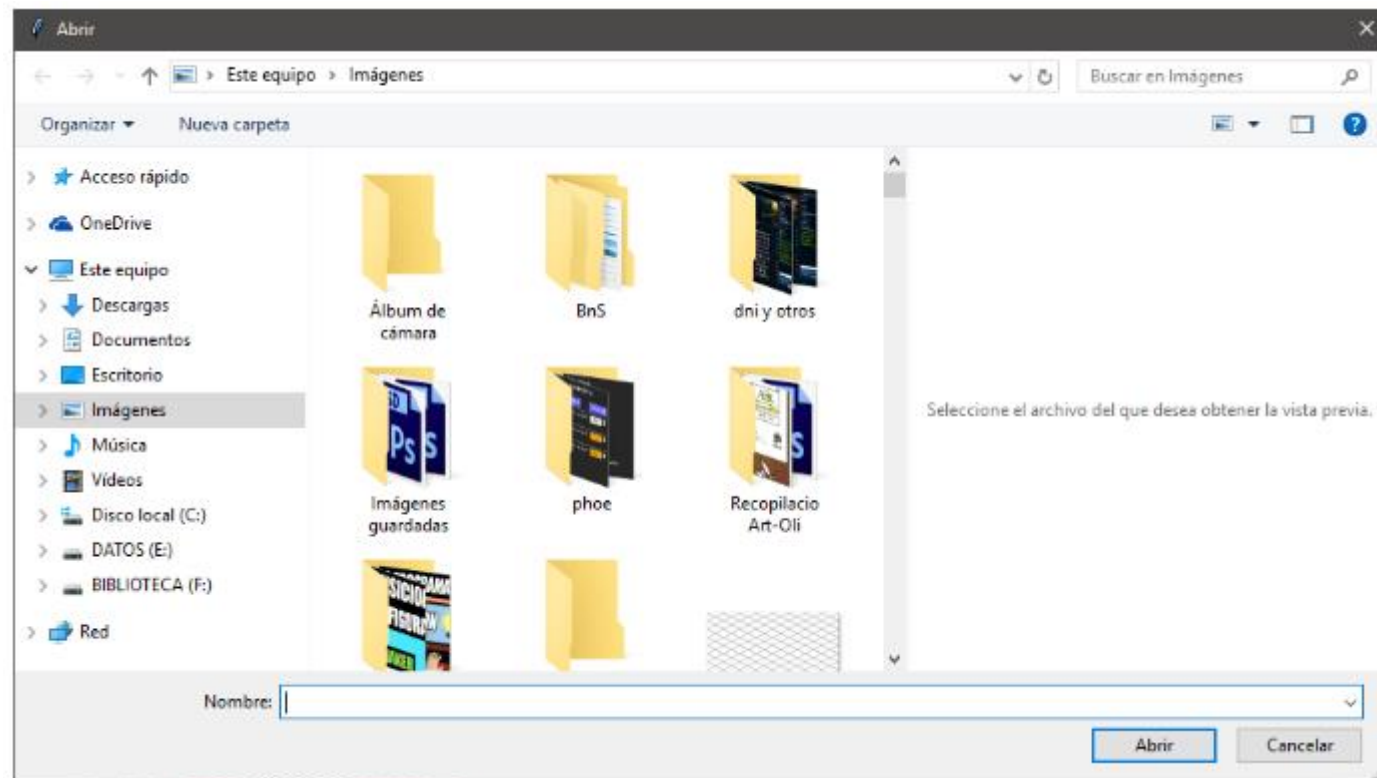


AskOpenFile

Y por último, un vistazo a la clase `FileDialog`, que nos permite realizar varias tareas como conseguir la ruta de un fichero para poder abrirlo, o para guardarlo:

```
from tkinter import filedialog as FileDialog

def test():
    fichero = FileDialog.askopenfilename(title="Abrir un fichero")
    print(fichero)
```

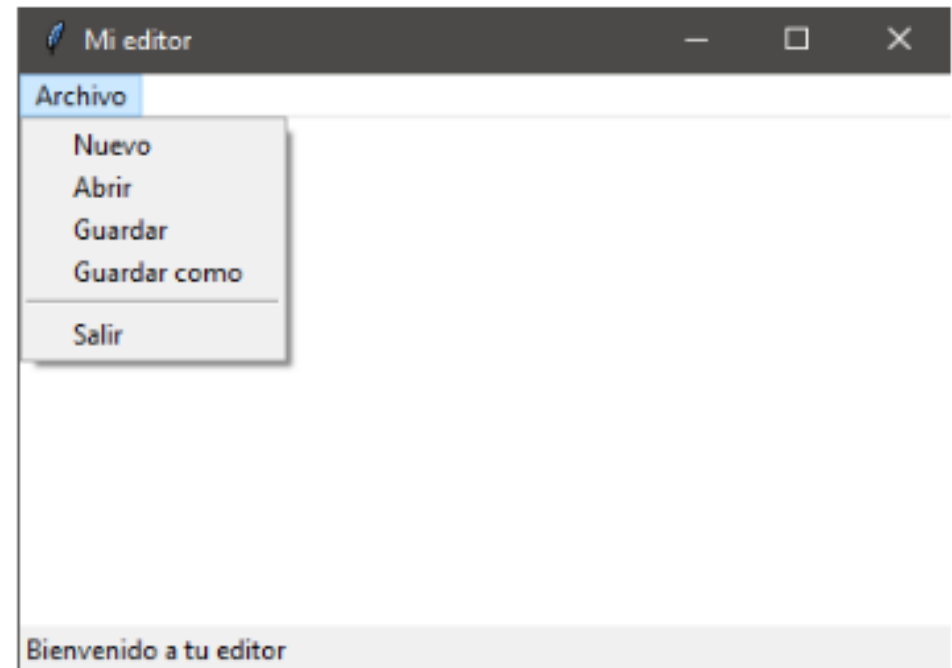


Practica P04- Editor de Texto

CREAR UN EDITOR DE TEXTO

En la web de **HectorProfe** hay un ejercicio para crear un editor de texto. Vale la pena leerlo y estudiarlo porque da una idea de la potencia de tkinter.

<https://docs.hektorprofe.net/python/interfaces-graficas-con-tkinter/editor-de-texto/>



Listbox

Listboxes are created using the **Listbox** function:

```
l = Listbox(parent, height=10)
```



```
from tkinter import *  
from tkinter.ttk import *
```

```
root = Tk()
```

```
countrysnames = ('Argentina', 'Australia', 'Belgium',  
                 'Brazil')
```

```
cnames = StringVar(value=countrysnames)
```

```
scrollbar = Scrollbar(root, orient=VERTICAL)
```

```
lbox = Listbox(root, listvariable=cnames, height=12)
```

```
lbox.pack(side=LEFT, fill=BOTH, expand=1)
```

```
lbox.insert(END, 'Colombia')
```

```
root.mainloop()
```

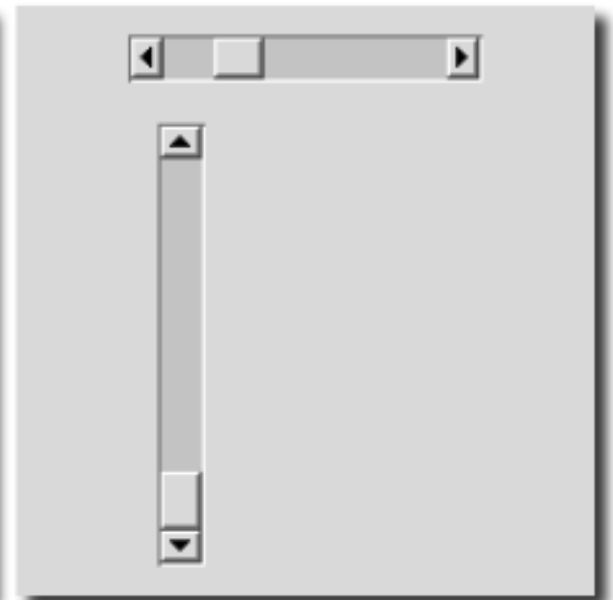

Scrollbar



Mac OS X



Windows



Linux

Scrollbar Widgets

Scrollbars are created using the **ttk.Scrollbar** command:

```
s = ttk.Scrollbar( parent, orient=VERTICAL, command=listbox.yview)
listbox.configure(yscrollcommand=s.set)
```

```
from tkinter import *
from tkinter.ttk import *
```

```
root = Tk()
```

Practica P05

selector de Paises y provincias

```
countrysnames = ('Argentina', 'Australia', 'Belgium', 'Brazil', 'Canada', 'China', 'Denmark', \
'Finland', 'France', 'Greece', 'India', 'Italy', 'Japan', 'Mexico', 'Netherlands', 'Norway', 'Spain', \
'Sweden', 'Switzerland')
```

```
cnames = StringVar(value=countrysnames)
scrollbar = Scrollbar(root, orient=VERTICAL)
lbox = Listbox(root, listvariable=cnames, height=12,
selectmode=EXTENDED, yscrollcommand=scrollbar.set)
```

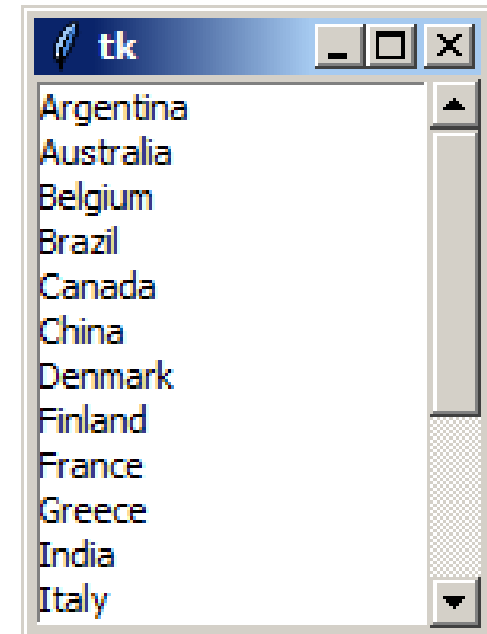
```
scrollbar.config(command=lbox.yview)
scrollbar.pack(side=RIGHT, fill=Y)
```

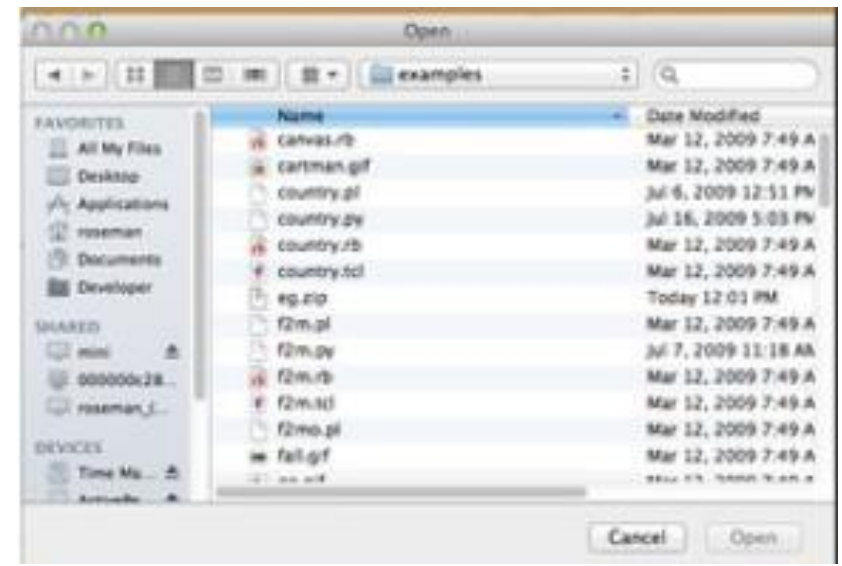
```
lbox.pack(side=LEFT, fill=BOTH, expand=1)
```

```
lbox.insert(END, 'Colombia')
```

```
root.mainloop()
```

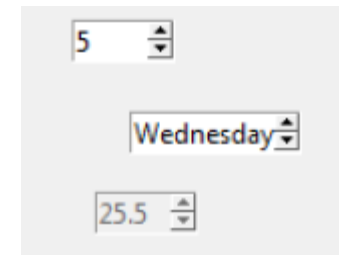
Ejemplo con scroll vertical. Usando este modelo crea un Listbox de Provincias leyendo los datos de un fichero.



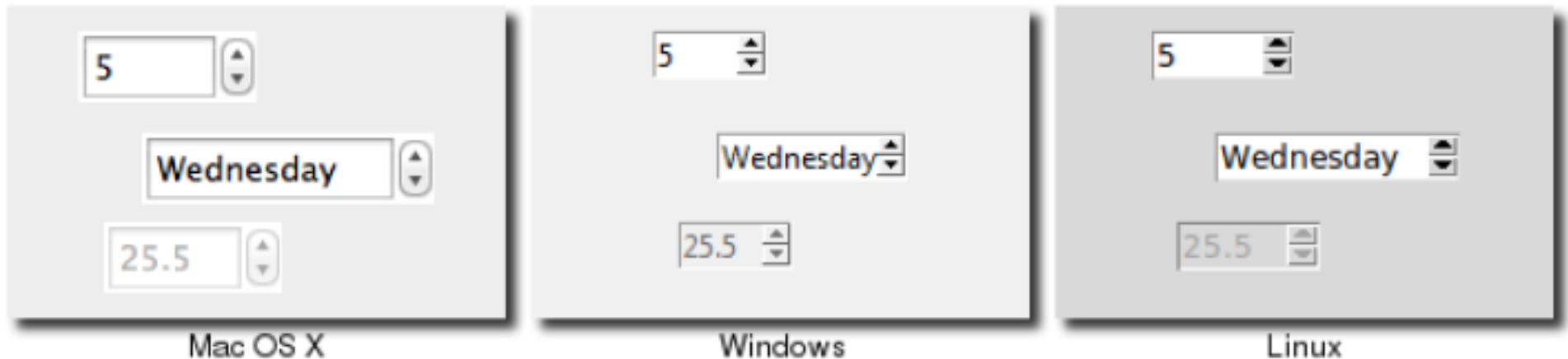


Otros controles Tkinter

Spinbox, selector de fichero, selector de color, paneles de ventana



Spinbox : Caja de incrementos



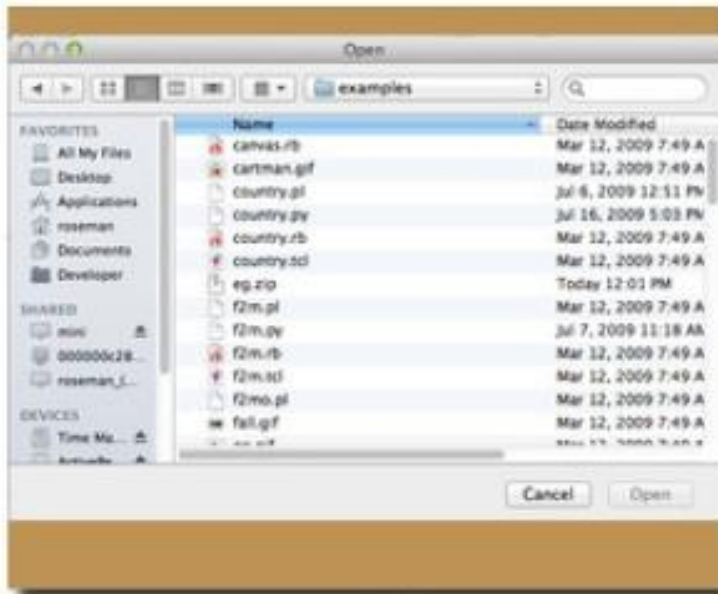
Spinbox widgets are created using the **Spinbox** function:

```
spinval = StringVar()  
s = Spinbox(parent, from_=1.0, to=100.0, textvariable=spinval)
```

Diálogo de selector de fichero

```
from tkinter import filedialog
filename = filedialog.askopenfilename()
filename = filedialog.asksaveasfilename()
dirname = filedialog.askdirectory()
```

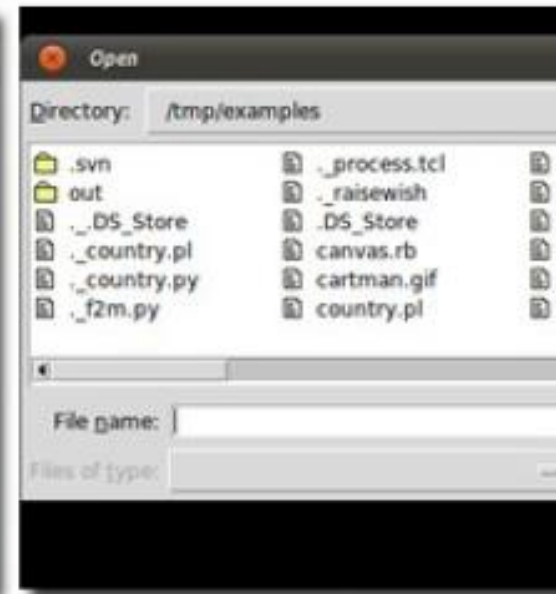
All of these commands produce *modal* dialogs, which means that the commands (and hence your program) will not continue running until the user submits the dialog. The commands return the full pathname of the file or directory the user has chosen, or return an empty string if the user cancels out of the dialog.



Mac OS X

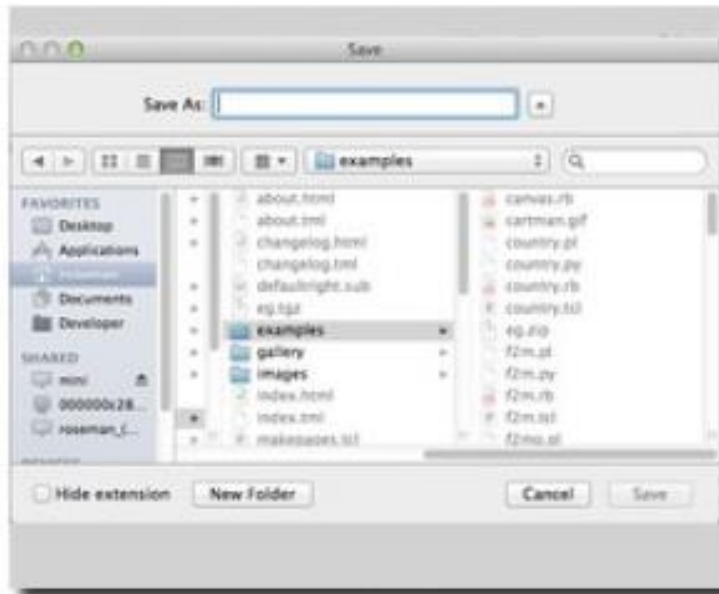


Windows



Linux

Diálogo de grabar fichero



Mac OS X



Windows



Linux

Save File Dialogs

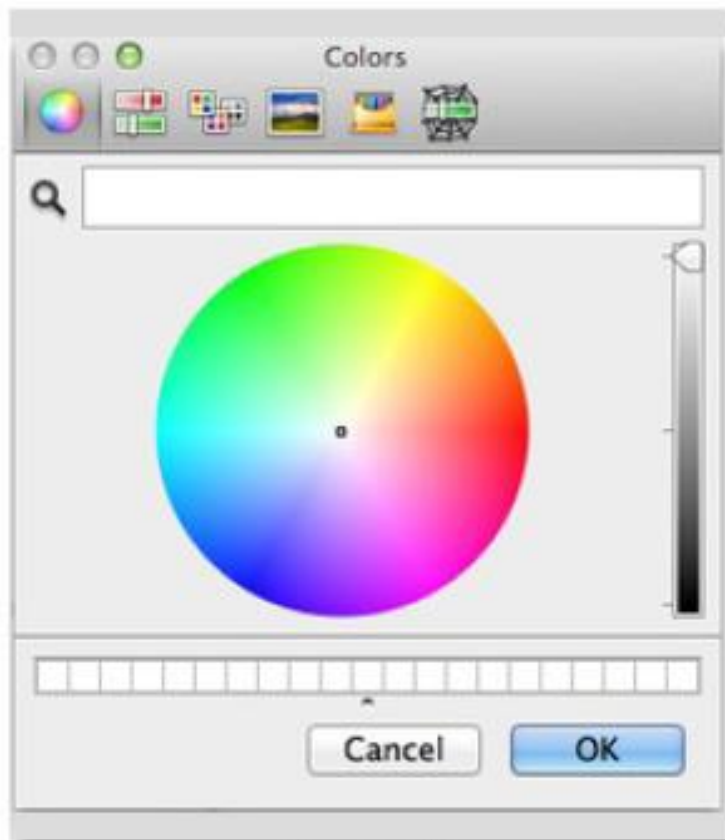
Modern Tkinter for Busy Python Developers

Diálogo de selector del color

Selecting Colors

There is also a modal dialog to let the user select a color. It will return a color value, e.g. "#ff62b8". The dialog takes an optional "initialcolor" option to specify an existing color that the user is presumably replacing.

```
from tkinter import colorchooser
colorchooser.askcolor(initialcolor='#ff0000')
```



Mac OS X



Windows



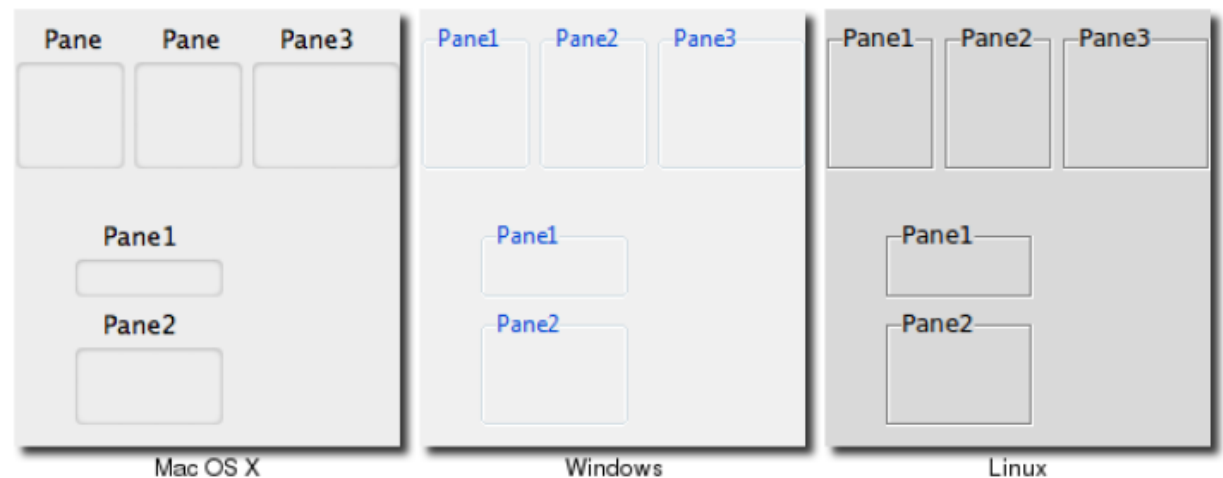
Linux

Paned Windows – Ventanas Panorámicas

Un widget de ventana panorámica le permite apilar dos o más widgets de tamaño variable uno encima del otro y uno debajo del otro (a la izquierda y derecha).

El usuario puede ajustar las alturas relativas (o anchos) de cada panel arrastrando una faja situada entre ellos.

Normalmente, los widgets que está agregando a una ventana panorámica serán frames/ marcos que contienen muchos otros widgets.



```
p = ttk.Panedwindow(parent, orient=VERTICAL)
# first pane, which would get widgets gridded into it:
f1 = ttk.Labelframe(p, text='Pane1', width=100, height=100)
f2 = ttk.Labelframe(p, text='Pane2', width=100, height=100) # second pane
p.add(f1)
p.add(f2)
```


Grid : column span y row span

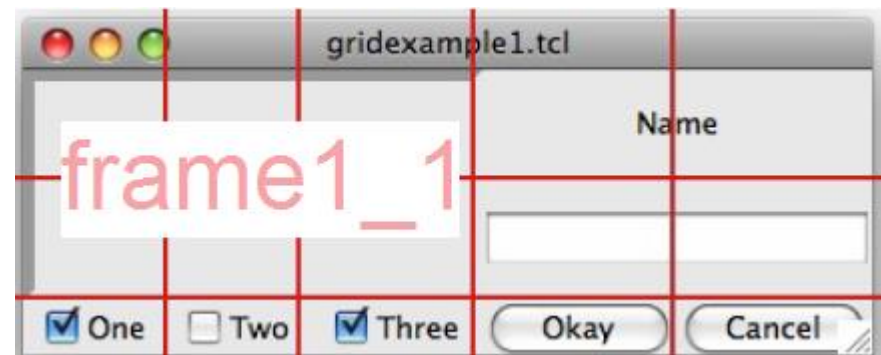
```
from tkinter import *  
from tkinter import ttk
```

```
root = Tk()  
frame1 = ttk.Frame(root)
```

```
frame1_1 = ttk.Frame(frame1, borderwidth=5, relief="sunken", width=200,height=100)  
namebl = ttk.Label(frame1, text="Name")  
name = ttk.Entry(frame1)
```

```
onevar = BooleanVar()  
twovar = BooleanVar()  
threevar = BooleanVar()
```

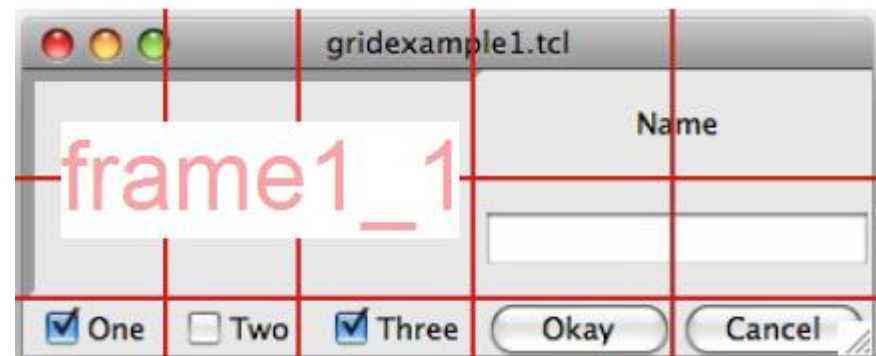
```
onevar.set(True)  
twovar.set(False)  
threevar.set(True)
```



```
one = ttk.Checkbutton(frame1, text="One", variable=onevar, onvalue=True)  
two = ttk.Checkbutton(frame1, text="Two", variable=twovar, onvalue=True)  
three = ttk.Checkbutton(frame1, text="Three", variable=threevar, onvalue=True)
```

```
ok = ttk.Button(frame1, text="Okay")
cancel = ttk.Button(frame1, text="Cancel")
```

```
frame1.grid      (column=0, row=0)
frame1_1.grid    (column=0, row=0, colspan=3, rowspan=2)
name1bl.grid     (column=3, row=0, colspan=2)
name.grid        (column=3, row=1, colspan=2)
one.grid         (column=0, row=3)
two.grid         (column=1, row=3)
three.grid       (column=2, row=3)
ok.grid          (column=3, row=3)
cancel.grid      (column=4, row=3)
root.mainloop()
```



Encapsulado de tkinter

Usar la metodología de POO (Programación Orientada a Objetos) en tkinter, nos facilita gestionar aplicaciones con varias ventanas.

```
import tkinter as tk

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

app=Aplicacion()
```

En el esquema básico, la clase Tk se crea dentro de la aplicación, y crea de uso privado para esa clase.

Ejemplo de encapsulado en clases.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import tkinter as tk
from tkinter import ttk
class Application(ttk.Frame):

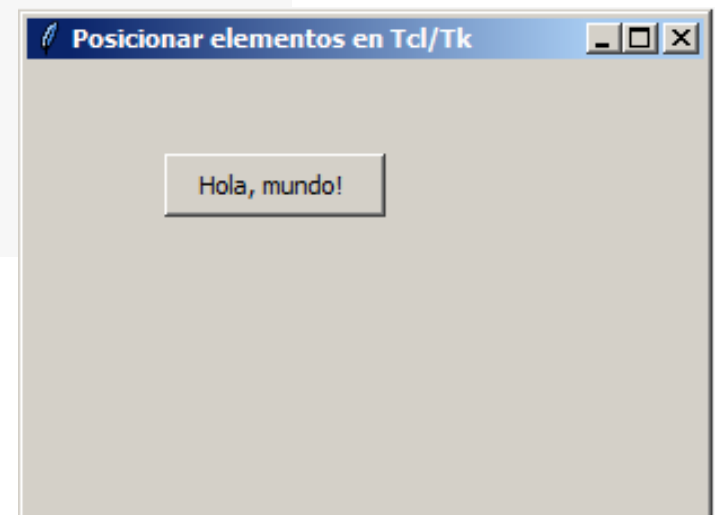
    def __init__(self, main_window):
        super().__init__(main_window)
        main_window.title("Posicionar elementos en Tcl/Tk")

        main_window.configure(width=300, height=200)
        # Ignorar esto por el momento.
        self.place(relwidth=1, relheight=1)

        self.button = ttk.Button(self, text="Hola, mundo!")
        self.button.place(x=60, y=40, width=100, height=30)

main_window = tk.Tk()
app = Application(main_window)
app.mainloop()
```

En un esquema más “reciclable”, la clase Tk se crea fuera de la aplicación, y la clase Aplicación se crea con el tipo de clase Frame,



Ejercicio2. Creando un menú dentro de una clase

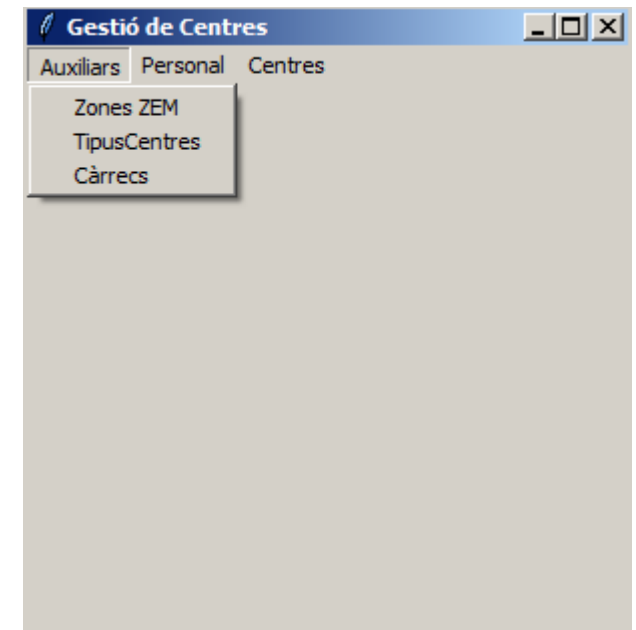
```
class Application(ttk.Frame):

    def __init__(self, window):
        super().__init__(window)
        self.win = window
        self.dimension_pantalla ()
        self.pedir_menu ()

    def dimension_pantalla (self):
        self.win.title("Gestió de Centres")
        self.win.geometry('400x400')
        self.win.minsize(width=300, height=400)
        self.win.maxsize(width=300, height=400)

    def montar_menu (self) :
        pass

#----- Main program
window = tk.Tk()
app = Application(window)
app.mainloop()
```



Ara completamos la función pedir_menu

```
def montar_menu (self) :
```

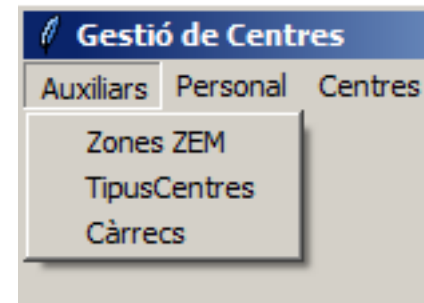
```
    menubar = Menu(self.win)
    self.win.config(menu=menubar)
    auxmenu = Menu(menubar)
    persmenu = Menu(menubar)
    centremenu = Menu(menubar)
```

```
    auxmenu = Menu(menubar, tearoff=0)
    auxmenu.add_command(label="Zones ZEM", command=self.doZones)
    auxmenu.add_command(label="TipusCentres", command=self.doTipusC)
    auxmenu.add_command(label="Càrrecs", command=self.doCarrecs)

    persmenu = Menu(menubar, tearoff=0)
    persmenu.add_command(label="Manteniment", command=self.doMantPers)
    persmenu.add_command(label="Consultes", command=donothing_out)

    centremenu = Menu(menubar, tearoff=0)
    centremenu.add_command(label="Manteniment", command=donothing_out)
    centremenu.add_command(label="Consultes", command=donothing_out)

    menubar.add_cascade(label="Auxiliars", menu=auxmenu)
    menubar.add_cascade(label="Personal", menu=persmenu)
    menubar.add_cascade(label="Centres", menu=centremenu)
```



Añadimos estas funciones dentro de la clase

```
def doZones(self):  
    Button(self.win, text="Zones").pack()  
def doTipusC(self):  
    Button(self.win, text="TipusC").pack()  
def doCarrecs(self):  
    Button(self.win, text="Carrecs").pack()  
def doMantPers(self):  
    pass
```

Y fuera de la definición de clase añadimos

```
def donothing_out():  
    filewin = Toplevel(window)  
    button = Button(filewin, text="Do nothing button")  
    button.pack()
```

Práctica P06

Pandas encapsulado

```
import pandas as pd
import sys
from tkinter import *
from tkinter import filedialog
```

```
class Application (Frame):
```

```
    def __init__(self, window):
```

```
        super().__init__(window)
        window.geometry('880x250')
```

```
        filename = filedialog.askopenfilename(parent=window, title =
"Selecciona ", initialdir = "./dat/", filetypes = (("csv", "*.csv"), ("all
files", "*.*")))
```

```
        df = pd.read_csv(filename, sep=";")
        pd.set_option('display.max_rows', 500)
        pd.set_option('display.max_columns', 500)
        pd.set_option('display.width', 1000)
```

Prueba este programa, que continúa en la página siguiente y que permite mostrar un dataframe sobre una ventana tkinter. Esta usando una estructura tkinter encapsulada.


```
self.txt = Text(window, width = 450, height=10)
self.txt.pack()
sal = sys.stdout
```

```
sys.stdout = self.PrintToTXT(self)
```

```
print ('Llistat de dades')
print (df)
sys.stdout = sal
```

```
class PrintToTXT(object):
    def __init__(self, ap1):
        self.ap1 = ap1
    def write(self, s):
        self.ap1.txt.insert(END, s)
    def flush(self) :
        pass
```

```
if __name__ == '__main__':
    root = Tk()
    app = Application(root)
    app.mainloop()
```

Practica P07.1.Obtener Noticias



Parsing RSS feeds with Python

Estudia este fragmento de código y prueba de incorporarlo a un formulario con un botón y una caja de texto que muestre las noticias.

```
import feedparser
```

```
NewsFeed =
```

```
feedparser.parse("https://timesofindia.indiatimes.com/rssfeedst  
opstories.cms")
```

```
print ('Number of RSS posts :', len(NewsFeed.entries))
```

```
entry = NewsFeed.entries[1]
```

```
print (entry.published)
```

```
print ("*****")
```

```
print (entry.summary)
```

```
print ("-----News Link-----")
```

```
print (entry.link)
```

Practica P07.2 Avanzado - Selectores

CALCULO DE VENTAS

Crea una ventana que lea las ventas de una empresa y permita escoger los datos que se mostrarán y el tipo de gráfico.

Archivo de entrada : caja de texto + botón de selector de fichero
ver diapos 20 y 24)

Agrupar por : Centro / TipoCliente / Distrito : Radio Button :

Desglose por meses o Total anual : Radio Button :

Número de Factura y/o Total Ventas : Check Button.

Calcular Botón

Resultado : Caja de Texto

Gráfico de Líneas o Barras o Tarta : Botones

*De momento cuando pulsen calcular muestra el dataframe por la consola con la función `display(df)` de **IPython.core.display***

Ayudas para la práctica

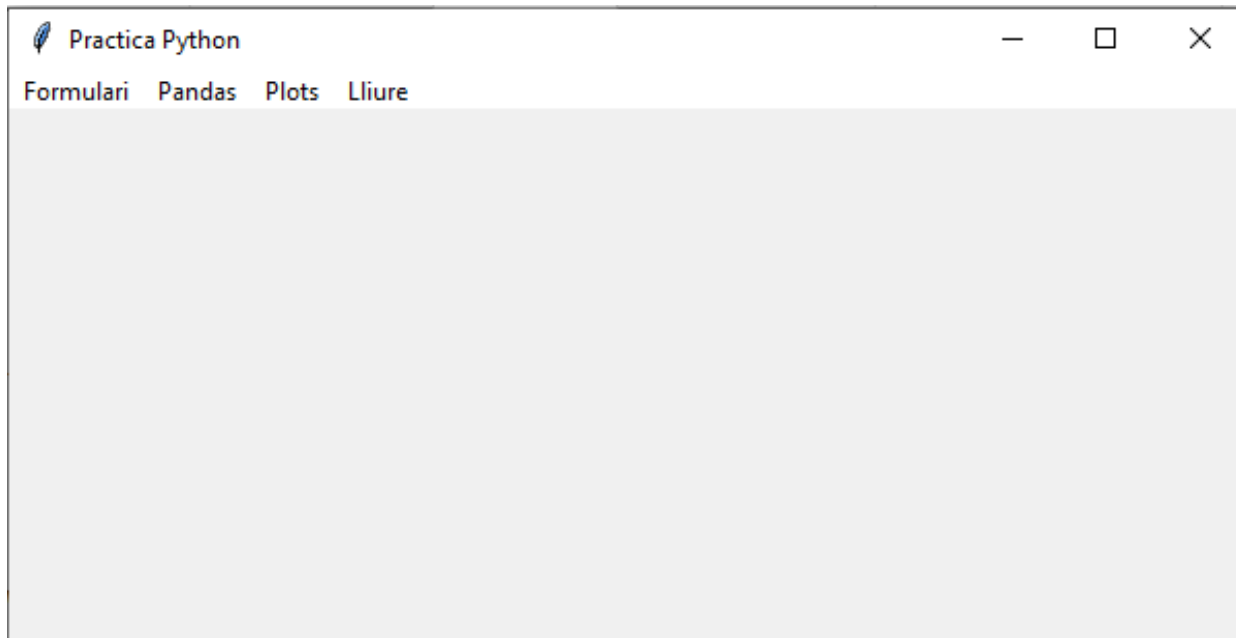
```
filename = filedialog.askopenfilename(parent=root,  
                                     initialdir = "./dat/",  
                                     title = "Selecciona arxiu",  
                                     filetypes = (("csv", "*.csv"),  
                                                ("all files", "*.*")))
```

```
df = pd.read_csv(filename, sep=";")  
pd.set_option('display.max_rows', 500)  
pd.set_option('display.max_columns', 500)  
pd.set_option('display.width', 1000)
```

```
display(df)
```

Practica P08 – Solo leer

Se realizará un menú con 4 opciones, que deben poder ser llamadas importando módulos i llamando a la clase Application del módulo.



Estudiar los modelos que proporciona el tutor.

Módulo menu.py

```
from tkinter import *
from tkinter import ttk

import mod_pandas
import mod_matplot
import form_clientes

class Application(Frame):

    def __init__(self, window):
        super().__init__(window)
        self.win = window
        self.win.title("Practica Python")
        self.win.geometry('600x600')
        self.crear_menu ()
        self.f1 = Frame(width=600, height=300)
        #self.f2 = Frame(width=600, height=300, background="white")

        self.f1.grid(row=0, column=1)
        #self.f2.grid(row=1, column=1)

    def opc_formulari (self):
        w3 = Tk()
        a3 = form_clientes.Application(w3)
        a3.mainloop()

    def opc_pandas (self):
        #pdver.Application(self.f1)
        w1 = Tk()
        ap1 = mod_pandas.Application(w1)
        ap1.mainloop()
```

```
def opc_plots (self):
    w2= Tk()
    start= mod_matplot.mclass (w2)
    w2.mainloop()


def opc_lliuere (self):
    pass

def crear_menu (self) :
    menubar = Menu(self.win)
    self.win.config(menu=menubar)

    menubar.add_command(label="Formulari", command=self.opc_formulari)
    menubar.add_command(label="Pandas", command=self.opc_pandas)
    menubar.add_command(label="Plots", command=self.opc_plots)
    menubar.add_command(label="Lliure", command=self.opc_lliuere)

#----- Main program
window = Tk()
app = Application(window)
app.mainloop()
```

Módulo Formulario

 Gestió de Clients— □ ×

Dni	Nom i Cognoms	Correu
0	Pruebas Usuario de pruebas	pruebas@ejemplo.com
1946-2006	Mario Benedetti Farugia	mario@ejemplo.com
46227542-A	Marta Artigas Font	5589@telefonica.es
37128934-J	Joan Artigas Piqué	joan@dosnoms.cat

+

-

Dni

Nom

Cognom 1

Cognom 2

Correu

Telèfon

cPostal

Cancelar

Enviar


```

from tkinter import Tk, Frame, StringVar, Label, Entry, DISABLED, Text, INSERT
from tkinter import ttk
from bd_clients import crea_client, consulta_clients, consulta_client

class Application (ttk.Frame):

    def __init__(self, window):
        super().__init__(window)
        self.win = window
        self.win.title("Gestió de Clients")
        self.win.geometry('600x500')
        self.opcio = ""

        self.frame1= Frame(self.win)      # Crea un frame per mostrar registres de clients
        self.muntar_frame1(self.frame1)

        self.frame2= Frame(self.win)      # Crea un frame pel formulari
        self.muntar_frame2(self.frame2)

```

Aquí se añadirán el resto de funciones o métodos de la clase

```

#----- Main
if __name__ == "__main__":
    window = Tk()
    app = Application(window)
    app.mainloop()

```

```
def muntar_frame1(self, frame1) :
```

```
def mostrar_clients(self) :
```

```
def cmd_afegir_client(self):
```

```
def cmd_esborrar_client(self):
```

```
def muntar_frame2(self, frame2) :
```

```
def frame2_visible(self):
```

```
def frame2_no_visible(self):
```

```
def cmd_enviar(self) :
```

```
def cmd_cancelar(self) :
```

```
def esborra_caixes(self):
```

```
def tracta_event(self, event):
```

Gestió de Clients

Dni	Nom i Cognoms	Correu
0	Pruebas Usuario de pruebas	pruebas@ejemplo.com
1946-2006	Mario Benedetti Farugia	mario@ejemplo.com
46227542-A	Marta Artigas Font	5589@telefonica.es
37128934-J	Joan Artigas Piqué	joan@dosnoms.cat

cmd_esborrar_client

cmd_afegir_client

+

-

Dni

Nom

Cognom 1

Cognom 2

Correu

Telèfon

cPostal

Cancelar

Enviar

cmd-cancelar

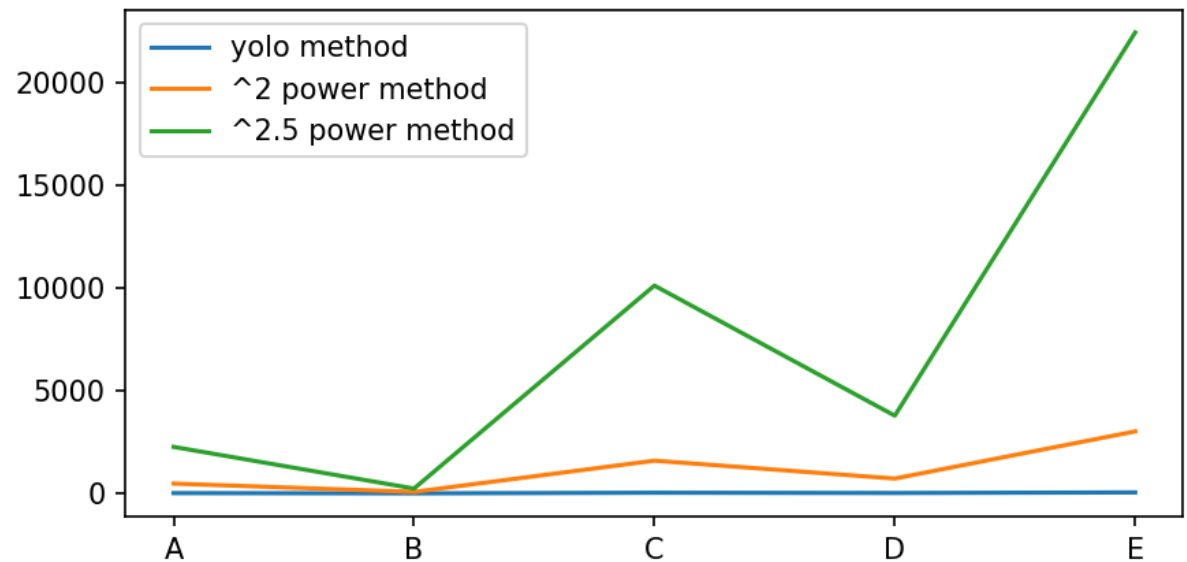
cmd_enviar

Pandas date range of 8 values in 1 timestamp column adjacent to a numpy random float array of 8 rows and 4 columns, displayed in a Tkinter table

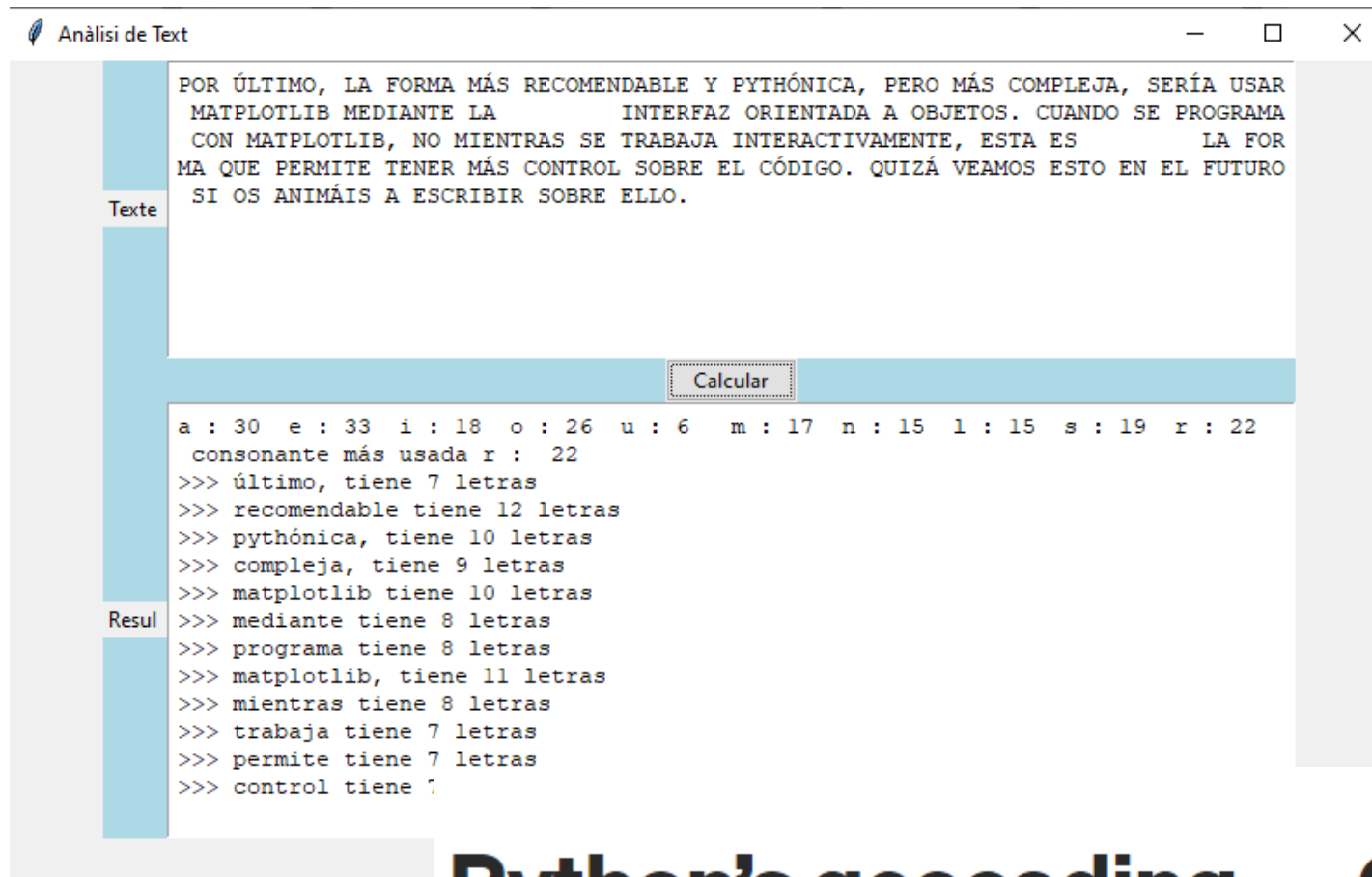
	A	B	C	D
2021-01-01	-1.432976	-1.779824	0.513862	1.330559
2021-01-02	-0.836686	0.869177	-1.232452	-0.534653
2021-01-03	-0.367641	-0.287856	0.369439	-0.866879
2021-01-04	-0.299382	0.629578	0.496779	-0.830201
2021-01-05	2.194129	1.182662	0.484524	0.682912
2021-01-06	0.630824	-0.353600	0.252639	1.413426
2021-01-07	-0.351649	0.904099	-0.987975	-0.817918
2021-01-08	-1.028977	0.763978	0.586111	-0.857585

Módulo Pandas

Módulo Matplotlib



Puede ser análisis de texto ...



Python's geocoding — Convert a list of addresses into a map

O cualquier otra cosa

How to work with geolocations APIs to receive data you need for plotting maps of your customers, factories, car fleet, and other subjects.