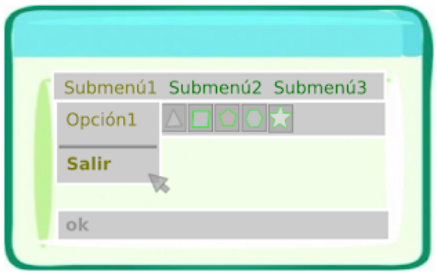
# Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)



**sábado, 26 de marzo de 2016**

## Menús, barras de herramientas y de estado en Tkinter




### Introducción

Tkinter cuenta con widgets específicos para incluir **menús de opciones** de distintos tipos en una aplicación. Además, siguiendo unas pautas en la construcción de ventanas es posible agregar otros elementos como **barras de herramientas y de estado**.

Todos estos componentes se utilizan con frecuencia en el desarrollo de interfaces porque facilitan mucho la interacción de los usuarios con las aplicaciones gráficas.

### Menús de opciones

Los menús pueden construirse agrupando en una barra de menú varios submenús desplegables, mediante menús basados en botones, o bien, utilizando los típicos menús contextuales que cambian sus opciones disponibles dependiendo del lugar donde se activan en la ventana de la aplicación.

Entre los tipos de opciones que se pueden incluir en un menú se encuentran aquellas que su estado representan un valor lógico de activada o desactivada (**add_checkbutton**); las que permiten elegir una opción de entre varias existentes (**add_radiobutton**) y las que ejecutan directamente un método o una función (**add_command**).

Las opciones de un menú pueden incluir iconos y asociarse a atajos o combinaciones de teclas que surten el mismo efecto que si éstas son seleccionadas con un clic de ratón. También, en un momento dado, pueden deshabilitarse para impedir que puedan ser seleccionadas.

### Barras de herramientas

Una aplicación Tkinter además de menús de opciones puede incorporar barras de herramientas construidas a base de botones apilados de manera horizontal o vertical. Estas barras de herramientas suelen situarse, muchas veces, justo debajo de la barra de menú de la aplicación y ejecutan los procesos más utilizados en el sistema.

### Barra de estado

Las barras de estado se utilizan con asiduidad en los programas gráficos para mostrar información que resulte útil a los usuarios. Normalmente, ocupan la última línea de la ventana de aplicación y en algunos casos puede ocultarse para dejar más espacio disponible a dicha ventana.

### PyRemoto, un ejemplo de implementación

**Buscar**

Buscar

**Python para impacientes**

Python
IPython
EasyGUI
Tkinter
JupyterLab
Numpy

**Anexos**

Guía urgente de MySQL
Guía rápida de SQLite3

**Entradas + populares**

Dar color a las salidas en la consola
En Python para dar color a las salidas en la consola (o en la terminal de texto) existen varias posibilidades. Hay un método basado ...

Instalación de Python, paso a paso
Instalación de Python 3.6 A finales de 2016 se produjo el lanzamiento de Python 3.6 . El propósito de esta entrada es mostrar, pas...

Añadir, consultar, modificar y suprimir elementos en Numpy
Acceder a los elementos de un array. [], [,], ... Acceder a un elemento de un array. Para acceder a un elemento se utiliz...

Cálculo con arrays Numpy
Numpy ofrece todo lo necesario para obtener un buen rendimiento cuando se trata de hacer cálculos con arrays. Por como está concebid...

Variables de control en Tkinter
Variables de control Las variables de control son objetos especiales que se asocian a los widgets para almacenar sus valore...

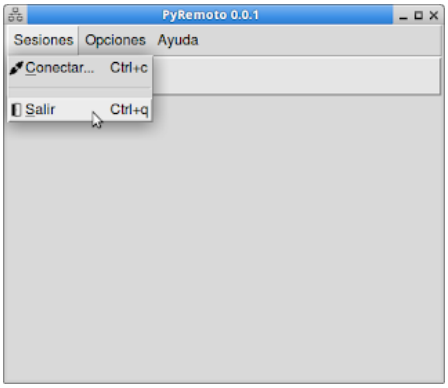
Tkinter: interfaces gráficas en Python
Introducción Con Python hay muchas posibilidades para programar una interfaz gráfica de usuario ( GUI ) pero Tkinter es fácil d...

Operaciones con fechas y horas. Calendarios
Los módulos datetime y calendar amplían las posibilidades del módulo time que provee funciones para manipular expresiones de ti...

Convertir, copiar, ordenar, unir y dividir arrays Numpy
Esta entrada trata sobre algunos métodos que se utilizan en Numpy para convertir listas en arrays y viceversa; para copiar arrays d...

Tkinter: Tipos de ventanas
Ventanas de aplicación y de diálogos En la entrada anterior tratamos los distintos gestores de geometría que se utilizan para di...

Threading: programación con hilos (I)
En programación, la técnica que permite que una aplicación ejecute

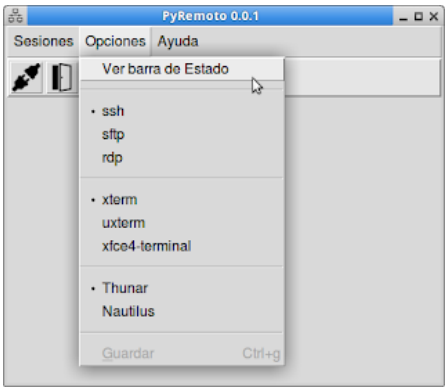



Ventana de aplicación

**PyRemoto** es un ejemplo que pretende mostrar cómo incluir menús y barras en una aplicación basada en Tkinter.

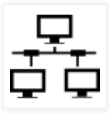
La ventana principal de esta aplicación cuenta con una barra de título, una barra de menú con tres submenús desplegables con distintos tipos de opciones, un menú contextual que se activa haciendo clic con el botón secundario del ratón en cualquier lugar de la ventana, una barra de herramientas con dos botones con imágenes y una barra de estado que es posible ocultar o mostrar mediante la opción "**Ver barra de estado**" del submenú "**Opciones**".


Menú contextual




Submenú "Opciones"

En el submenú "**Opciones**" se encuentra la opción "**Guardar**" que se habilitará cuando se elija alguna opción de entre las disponibles en este submenú; y se deshabilitará cuando se utilice la propia opción "**Guardar**", -prevista- para salvar las preferencias seleccionadas por el usuario.



simultáneamente varias operaciones en el mismo espacio de proceso se...

**Archivo**

marzo 2016 (1)

**python.org**

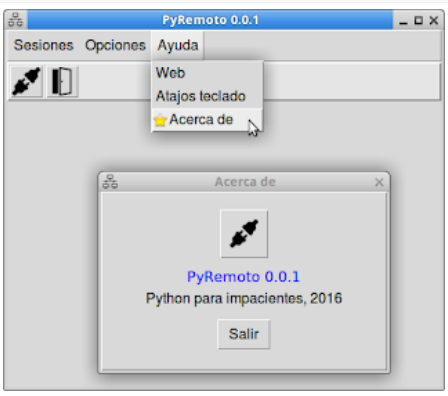
**pypi.org**

**Sitios**

- ActivePython
- Anaconda
- Bpython
- Django
- Flask
- Ipython
- IronPython
- Matplotlib
- MicroPython
- Numpy
- Pandas
- Pillow
- PortablePython
- PyBrain
- PyCharm
- PyDev
- PyGame
- Pypi
- PyPy
- Pyramid
- Python.org
- PyTorch
- SciPy.org
- Spyder
- Tensorflow
- TurboGears



La aplicación de ejemplo utiliza imágenes de diferentes tamaños que se localizan en una carpeta llamada "***imagen***". Estas imágenes son utilizadas en la barra de título, los submenús, la barra de herramientas y en la ventana de diálogo "**Acerca de**" del submenú "**Ayuda**". Han sido obtenidas del sitio flaticon.com que ofrece colecciones de imágenes organizadas por distintas temáticas a programadores y a diseñadores gráficos.




Ventana de diálogo "Acerca de"

También, en el programa se incluyen varias combinaciones de teclas asociadas a distintas opciones del menú, declaradas con el método **bind()**.

**¿Cómo funciona?**

La aplicación la inicia la función **main()** que comprueba que todas las imágenes de la aplicación existen en la carpeta "***imagen***". Si todas las imágenes están disponibles se creará el objeto aplicación mediante la clase **PyRemoto()**. Esta clase cuenta con un método **__init__()** que construye la ventana de la aplicación, los menús, la barra de herramientas y una barra de estado.

La aplicación está incompleta pero enseña cómo organizar los métodos que son llamados desde las distintas opciones de los menús y barras.

Por último, el ejemplo incluye la ventana de diálogo "**Acerca de**", que es de tipo modal, para mostrar el modo de abrir este tipo de ventanas desde una opción del menú.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
# name:        pyremoto.py (Python 3.x).
# description: Acceso Remoto a equipos por ssh, sftp y rdp
# purpose:     Construcción de menús, barras de herramientas
#              y de estado
# author:      python para impacientes
#
#------------------------------------------------------------

'''PyRemoto: Acceso Remoto a equipos por ssh, sftp y rdp '''

__author__ = 'python para impacientes'
__title__= 'PyRemoto'
__date__ = ''
__version__ = '0.0.1'
__license__ = 'GNU GPLv3'

import os, sys, webbrowser, platform
from tkinter import *
from tkinter import ttk, font, messagebox

# PYREMOTO: ACCESO REMOTO A EQUIPOS POR SSH, SFTP y RDP

class PyRemoto():
    ''' Clase PyRemoto '''
```

```
    # DECLARAR MÉTODO CONSTRUCTOR DE LA APLICACIÓN

    def __init__(self, img_carpeta, iconos):
        ''' Definir ventana de la aplicación, menú, submenús,
            menú contextual, barra de herramientas, barra de
            estado y atajos del teclado '''

        # INICIALIZAR VARIABLES

        self.img_carpeta = img_carpeta
        self.iconos = iconos

        # DEFINIR VENTANA DE LA APLICACIÓN:

        self.raiz = Tk()

        # ESTABLECER PROPIEDADES DE LA VENTANA DE APLICACIÓN:

        self.raiz.title("PyRemoto " + __version__)  # Título
        self.icono1= PhotoImage(file=self.iconos[0])  # Icono app
        self.raiz.iconphoto(self.raiz, self.icono1)  # Asigna icono app
        self.raiz.option_add("*Font", "Helvetica 12")  # Fuente predeterminada
        self.raiz.option_add('*tearOff', False)  # Deshabilita submenús flotantes
        self.raiz.attributes('-fullscreen', True)  # Maximiza ventana completa
        self.raiz.minsize(400,300)  # Establece tamaño minimo ventana

        # ESTABLECER ESTILO FUENTE PARA ALGUNOS WIDGETS:

        self.fuente = font.Font(weight='normal') # normal, bold, etc...

        # DECLARAR VARIABLES PARA OPCIONES PREDETERMINADAS:
        # (Estos valores se podrían leer de un archivo de
        # configuración)

        self.CFG_TIPOCONEX = IntVar()
        self.CFG_TIPOCONEX.set(1) # shh
        self.CFG_TIPOEMUT = IntVar()
        self.CFG_TIPOEMUT.set(1) # xterm
        self.CFG_TIPOEXP = IntVar()
        self.CFG_TIPOEXP.set(1) # thunar

        # DECLARAR VARIABLE PARA MOSTRAR BARRA DE ESTADO:

        self.estado = IntVar()
        self.estado.set(1)  # Mostrar Barra de Estado

        # DEFINIR BARRA DE MENÚ DE LA APLICACION:

        barramenu = Menu(self.raiz)
        self.raiz['menu'] = barramenu

        # DEFINIR SUBMENÚS 'Sesiones', 'Opciones' y 'Ayuda':

        menu1 = Menu(barramenu)
        self.menu2 = Menu(barramenu)
        menu3 = Menu(barramenu)
        barramenu.add_cascade(menu=menu1, label='Sesiones')
        barramenu.add_cascade(menu=self.menu2, label='Opciones')
        barramenu.add_cascade(menu=menu3, label='Ayuda')

        # DEFINIR SUBMENÚ 'Sesiones':

        icono2 = PhotoImage(file=self.iconos[1])
        icono3 = PhotoImage(file=self.iconos[2])

        menu1.add_command(label='Conectar...',
                          command=self.f_conectar,
                          underline=0, accelerator="Ctrl+c",
                          image=icono2, compound=LEFT)
        menu1.add_separator()  # Agrega un separador
        menu1.add_command(label='Salir', command=self.f_salir,
                          underline=0, accelerator="Ctrl+q",
                          image=icono3, compound=LEFT)

        # DEFINIR SUBMENÚ 'Opciones':

        self.menu2.add_checkbutton(label='Barra de Estado',
                          variable=self.estado, onvalue=1,
                          offvalue=0,
                          command=self.f_verestado)
        self.menu2.add_separator()
```

```
        self.menu2.add_radiobutton(label='ssh',
                        variable=self.CFG_TIPOCONEX,
                        command=self.f_cambiaropc,
                        value=1)
        self.menu2.add_radiobutton(label='sftp',
                        variable=self.CFG_TIPOCONEX,
                        command=self.f_cambiaropc,
                        value=2)
        self.menu2.add_radiobutton(label='rdp',
                        variable=self.CFG_TIPOCONEX,
                        command=self.f_cambiaropc,
                        value=3)
        self.menu2.add_separator()
        self.menu2.add_radiobutton(label='xterm',
                        variable=self.CFG_TIPOEMUT,
                        command=self.f_cambiaropc,
                        value=1)
        self.menu2.add_radiobutton(label='uxterm',
                        variable=self.CFG_TIPOEMUT,
                        command=self.f_cambiaropc,
                        value=2)
        self.menu2.add_radiobutton(label='xfce4-terminal',
                        variable=self.CFG_TIPOEMUT,
                        command=self.f_cambiaropc,
                        value=3)
        self.menu2.add_separator()
        self.menu2.add_radiobutton(label='Thunar',
                        variable=self.CFG_TIPOEXP,
                        command=self.f_cambiaropc,
                        value=1)
        self.menu2.add_radiobutton(label='Nautilus',
                        variable=self.CFG_TIPOEXP,
                        command=self.f_cambiaropc,
                        value=2)
        self.menu2.add_separator()
        self.menu2.add_command(label='Guardar',
                        command=self.f_opcionguardar,
                        state="disabled", underline=0,
                        accelerator="Ctrl+g")

        # DEFINIR SUBMENÚ 'Ayuda':

        menu3.add_command(label='Web', command=self.f_web)
        menu3.add_command(label='Atajos teclado',
                        command=self.f_atajos)
        icono4 = PhotoImage(file=self.iconos[3])
        menu3.add_command(label="Acerca de",
                        command=self.f_acerca,
                        image=icono4, compound=LEFT)

        # DEFINIR BARRA DE HERRAMIENTAS:

        self.icono5 = PhotoImage(file=self.iconos[4])
        icono6 = PhotoImage(file=self.iconos[5])

        barraherr = Frame(self.raiz, relief=RAISED,
                        bd=2, bg="#E5E5E5")
        bot1 = Button(barraherr, image=self.icono5,
                    command=self.f_conectar)
        bot1.pack(side=LEFT, padx=1, pady=1)
        bot2 = Button(barraherr, image=icono6,
                    command=self.f_salir)
        bot2.pack(side=LEFT, padx=1, pady=1)
        barraherr.pack(side=TOP, fill=X)

        # DEFINIR BARRA DE ESTADO:
        # Muestra información del equipo

        info1 = platform.system()
        info2 = platform.node()
        info3 = platform.machine()

        # Otro modo de obtener la información:
        # (No disponible en algunas versiones de Windows)

        #info1 = os.uname().sysname
        #info2 = os.uname().nodename
        #info3 = os.uname().machine

        mensaje = " " + info1+ ": "+info2+" - "+info3
        self.barraest = Label(self.raiz, text=mensaje,
                              bd=1, relief=SUNKEN, anchor=W)
```

```
        self.barraest.pack(side=BOTTOM, fill=X)

        # DEFINIR MENU CONTEXTUAL

        self.menucontext = Menu(self.raiz, tearoff=0)
        self.menucontext.add_command(label="Conectar",
                                     command=self.f_conectar)
        self.menucontext.add_command(label="Salir",
                                     command=self.f_salir)

        # DECLARAR TECLAS DE ACCESO RAPIDO:

        self.raiz.bind("<Control-c>",
                       lambda event: self.f_conectar())
        self.raiz.bind("<Control-g>",
                       lambda event: self.f_guardar())
        self.raiz.bind("<Control-q>",
                       lambda event: self.f_salir())
        self.raiz.bind("<Button-3>",
                       self.f_mostrarmenucontext)
        self.raiz.mainloop()

    # DECLARAR OTROS MÉTODOS DE LA APLICACIÓN:

    def f_opcionguardar(self):
        ''' Si opción 'Guardar' está habilitada llama a
            método para guardar opciones de configuración
            de la aplicación '''

        if self.menu2.entrycget(13,"state")=="normal":
            self.menu2.entryconfig(13, state="disabled")
            self.f_guardarconfig()

    def f_guardarconfig(self):
        ''' Guardar opciones de configuración de la aplicación '''
        print("Configuración guardada")

    def f_conectar(self):
        ''' Definir ventana de diálogo para conectar con equipos '''
        print("Conectando")

    def f_cambiaropc(self):
        ''' Habilitar opción 'Guardar' al elegir alguna opción
            de tipo de conexión, emulador de terminar o
            explorador de archivos '''
        self.menu2.entryconfig("Guardar", state="normal")

    def f_verestado(self):
        ''' Ocultar o Mostrar barra de estado '''

        if self.estado.get() == 0:
            self.barraest.pack_forget()
        else:
            self.barraest.pack(side=BOTTOM, fill=X)

    def f_mostrarmenucontext(self, e):
        ''' Mostrar menú contextual '''
        self.menucontext.post(e.x_root, e.y_root)

    def f_web(self):
        ''' Abrir página web en navegador Internet '''

        pag1 = 'http://python-para-impacientes.blogspot.com/'
        webbrowser.open_new_tab(pag1)

    def f_atajos(self):
        ''' Definir ventana de diálogo con lista de
            combinaciones de teclas de la aplicación '''
        pass

    def f_acerca(self):
        ''' Definir ventana de diálogo 'Acerca de' '''

        acerca = Toplevel()
        acerca.geometry("320x200")
        acerca.resizable(width=False, height=False)
        acerca.title("Acerca de")
        marco1 = ttk.Frame(acerca, padding=(10, 10, 10, 10),
                           relief=RAISED)
        marco1.pack(side=TOP, fill=BOTH, expand=True)
        etiq1 = Label(marco1, image=self.icono5,
                      relief='raised')
```

```
        etiq1.pack(side=TOP, padx=10, pady=10,
                   ipadx=10, ipady=10)
        etiq2 = Label(marco1, text="PyRemoto "+__version__,
                      foreground='blue', font=self.fuente)
        etiq2.pack(side=TOP, padx=10)
        etiq3 = Label(marco1,
                      text="Python para impacientes")
        etiq3.pack(side=TOP, padx=10)
        boton1 = Button(marco1, text="Salir",
                        command=acerca.destroy)
        boton1.pack(side=TOP, padx=10, pady=10)
        boton1.focus_set()
        acerca.transient(self.raiz)
        self.raiz.wait_window(acerca)

    def f_salir(self):
        ''' Salir de la aplicación '''
        self.raiz.destroy()

# FUNCIONES DE LA APLICACIÓN

def f_verificar_iconos(iconos):
    ''' Verifica existencia de iconos

    iconos -- Lista de iconos '''

    for icono in iconos:
        if not os.path.exists(icono):
            print('Icono no encontrado:', icono)
            return(1)
    return(0)

def main():
    ''' Iniciar aplicación '''

    # INICIALIZAR VARIABLES CON RUTAS

    app_carpeta = os.getcwd()
    img_carpeta = app_carpeta + os.sep + "imagen" + os.sep

    # DECLARAR Y VERIFICAR ICONOS DE LA APLICACIÓN:

    iconos = (img_carpeta + "pyremoto64x64.png",
              img_carpeta + "conec16x16.png",
              img_carpeta + "salir16x16.png",
              img_carpeta + "star16x16.png",
              img_carpeta + "conec32x32.png",
              img_carpeta + "salir32x32.png")
    error1 = f_verificar_iconos(iconos)

    if not error1:
        mi_app = PyRemoto(img_carpeta, iconos)
    return(0)

if __name__ == '__main__':
    main()
```

Anterior: Variables de control en Tkinter

Ir al índice del tutorial de Python

Publicado por Pherkad en 5:33

Etiquetas: Python3, Tkinter

Entrada más reciente Inicio Entrada antigua