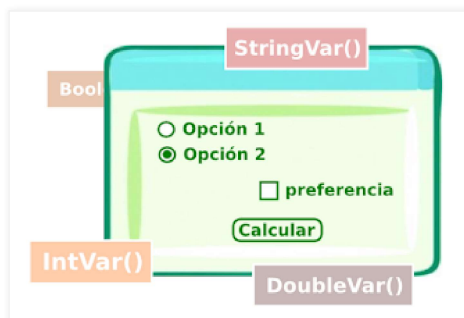


"Simple es mejor que complejo" (Tim Peters)

Python IPython EasyGUI Tkinter JupyterLab Numpy

miércoles, 3 de febrero de 2016

Variables de control en Tkinter



Variables de control

Las variables de control son objetos especiales que se asocian a los widgets para almacenar sus valores y facilitar su disponibilidad en otras partes del programa. Pueden ser de tipo numérico, de cadena y booleano.

Cuando una variable de control cambia de valor el widget que la utiliza lo refleja automáticamente, y viceversa.

Las variables de control también se emplean para conectar varios widgets del mismo tipo, por ejemplo, varios controles del tipo **Radiobutton**. En este caso tomarán un valor de varios posibles.

Declarar variables de control

Las variables de control se declaran de forma diferente en función al tipo de dato que almacenan:

```
entero = IntVar() # Declara variable de tipo entera
flotante = DoubleVar() # Declara variable de tipo flotante
cadena = StringVar() # Declara variable de tipo cadena
booleano = BooleanVar() # Declara variable de tipo booleana
```

También, en el momento de declarar una variable es posible asignar un valor inicial:

```
blog = StringVar(value="Python para impacientes")
```

Método set()

El método **set()** asigna un valor a una variable de control. Se utiliza para modificar el valor o estado de un widget:

```
nombre = StringVar()
id_art = IntVar()
nombre.set("Python para impacientes")
id_art.set(1)
blog = ttk.Entry(ventana, textvariable=nombre, width=25)
arti = ttk.Label(ventana, textvariable=id_art)
```

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 concebido...

[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...

[Tablas con estilo con Tabulate](#)

El módulo Tabulate , desarrollado por Sergey Astanin, permite imprimir en la salida estándar o escribir en un archivo de texto tabla...

[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...

[Threading: programación con hilos \(I\)](#)

En programación, la técnica que permite que una aplicación ejecute simultáneamente varias operaciones en el mismo espacio de proceso se...

[El módulo random](#)

El módulo random de la librería estándar de Python incluye un conjunto de funciones que permiten obtener de distintos modos números a...

[Sesión interactiva Python](#)

En una sesión interactiva con la consola predeterminada de Python3 podemos ejecutar una sentencia o un bloque de sentencias. Este modo...

Este sitio utiliza cookies de Google para prestar sus servicios y para analizar su tráfico. Tu dirección IP y user-agent se comparten con Google, junto con las métricas de rendimiento y de seguridad, para garantizar la calidad del servicio, generar estadísticas de uso y detectar y solucionar abusos.

MÁS INFORMACIÓN ACEPTAR

```
print('Blog:', nombre.get())
print('Id artículo:', id_art.get())
```

Método trace()

El método **trace()** se emplea para "detectar" cuando una variable es leída, cambia de valor o es borrada:

widget.trace(tipo, función)

El primer argumento establece el tipo de suceso a comprobar: 'r' lectura de variable, 'w' escritura de variable y 'u' borrado de variable. El segundo argumento indica la función que será llamada cuando se produzca el suceso.

En el siguiente ejemplo se define una variable de control de tipo cadena y con el método **trace()** se asocian su lectura y cambio de valor a dos funciones que son llamadas cuando ocurran estos sucesos. Concretamente, cuando se utilice el método **set()** se llamará a la función 'cambia()' y cuando se use **get()** a la función 'lee()'.

```
def cambia(*args):
    print("Ha cambiado su valor")

def lee(*args):
    print("Ha sido leído su valor")

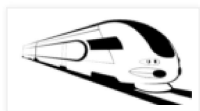
variable = StringVar()
variable.trace("w", cambia)
variable.trace("r", lee)
variable.set("Hola")
print(variable.get())
print(variable.get())
```

Estrategias para validar y calcular datos

Cuando se construye una ventana con varios widgets se pueden seguir distintas estrategias para validar los datos que se introducen durante la ejecución de un programa:

- Una opción posible podría validar la información y realizar los cálculos después de que sea introducida, por ejemplo, después de presionar un botón.
- Otra posibilidad podría ser haciendo uso del método **trace()** y de la opción 'command', para validar y calcular la información justo en el momento que un widget y su variable asociada cambien de valor.

A continuación, se muestra un mismo caso práctico utilizando ambas técnicas.



El programa de este ejemplo calcula el coste de un viaje en tren teniendo en cuenta el número de viajeros; el tipo de billete (de sólo ida o de ida y vuelta); la clase en la cual se viaja (que puede ser clase turista, primera o lujo); la distancia en kilómetros y el precio por kilómetro (por defecto es 0,10 céntimos de euro).

El cálculo del importe a pagar se realiza multiplicando número de viajeros por km y precio, con los siguientes incrementos:

- Si el viaje es de ida y vuelta se multiplica el total por 1,5
- Si la clase es primera se multiplica el total por 1,2 y si es de lujo se multiplica por 2

Validación y cálculo posterior

rchivo

febrero 2016 (1) ▾

python.org



pypi.org

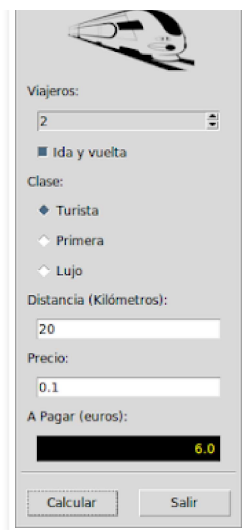


Sitios

- [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](#)

Este sitio utiliza cookies de Google para prestar sus servicios y para analizar su tráfico. Tu dirección IP y user-agent se comparten con Google, junto con las métricas de rendimiento y de seguridad, para garantizar la calidad del servicio, generar estadísticas de uso y detectar y solucionar abusos.

[MÁS INFORMACIÓN](#) [ACEPTAR](#)



```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from tkinter import *
from tkinter import ttk

# Calcula coste de viaje con validación y cálculo posterior

class Aplicacion():
    def __init__(self):
        self.raiz = Tk()
        self.raiz.title("Alta Velocidad")

        # Declara variables de control

        self.num_via = IntVar(value=1)
        self.ida_vue = BooleanVar()
        self.clase = StringVar(value='t')
        self.km = IntVar(value=1)
        self.precio = DoubleVar(value=0.10)
        self.total = DoubleVar(value=0.0)

        # Carga imagen para asociar a widget Label()

        tren = PhotoImage(file='tren-128x64.png')

        # Declara widgets de la ventana
        # Se incluye el widget de tipo Button 'Calcular' que utiliza
        # la opción 'command' para validar datos y calcular el
        # importe a pagar cuando sea presionado

        self.imagen1 = ttk.Label(self.raiz, image=tren,
                                anchor="center")
        self.etiq1 = ttk.Label(self.raiz, text="Viajeros:")
        self.viaje = Spinbox(self.raiz, from_=1, to=20, wrap=True,
                             textvariable=self.num_via,
                             state='readonly')
        self.idavue = ttk.Checkbutton(self.raiz, text='Ida y vuelta',
                                      variable=self.ida_vue,
                                      onvalue=True, offvalue=False)
        self.etiq2 = ttk.Label(self.raiz, text="Clase:")
        self.clase1 = ttk.Radiobutton(self.raiz, text='Turista',
                                      variable=self.clase, value='t')
        self.clase2 = ttk.Radiobutton(self.raiz, text='Primera',
                                      variable=self.clase, value='p')
        self.clase3 = ttk.Radiobutton(self.raiz, text='Lujo',
                                      variable=self.clase, value='l')
        self.etiq3 = ttk.Label(self.raiz,
                                text="Distancia Kilómetros:")
        self.dist = ttk.Entry(self.raiz, textvariable=self.km,
                              width=10)
```

```
self.separ1 = ttk.Separator(self.raiz, orient=HORIZONTAL)

self.boton1 = ttk.Button(self.raiz, text="Calcular",
                        command=self.calcular)
self.boton2 = ttk.Button(self.raiz, text="Salir",
                        command=quit)

self.imagen1.pack(side=TOP, fill=BOTH, expand=True,
                  padx=10, pady=5)
self.etiq1.pack(side=TOP, fill=BOTH, expand=True,
                padx=10, pady=5)
self.viaje.pack(side=TOP, fill=X, expand=True,
                padx=20, pady=5)
self.idavue.pack(side=TOP, fill=X, expand=True,
                 padx=20, pady=5)
self.etiq2.pack(side=TOP, fill=BOTH, expand=True,
                padx=10, pady=5)
self.clase1.pack(side=TOP, fill=BOTH, expand=True,
                 padx=20, pady=5)
self.clase2.pack(side=TOP, fill=BOTH, expand=True,
                 padx=20, pady=5)
self.clase3.pack(side=TOP, fill=BOTH, expand=True,
                 padx=20, pady=5)
self.etiq3.pack(side=TOP, fill=BOTH, expand=True,
                padx=10, pady=5)
self.dist.pack(side=TOP, fill=X, expand=True,
                padx=20, pady=5)
self.etiq4.pack(side=TOP, fill=BOTH, expand=True,
                padx=10, pady=5)
self.coste.pack(side=TOP, fill=X, expand=True,
                 padx=20, pady=5)
self.etiq5.pack(side=TOP, fill=BOTH, expand=True,
                padx=10, pady=5)
self.etiq6.pack(side=TOP, fill=BOTH, expand=True,
                padx=20, pady=5)
self.separ1.pack(side=TOP, fill=BOTH, expand=True,
                 padx=5, pady=5)
self.boton1.pack(side=LEFT, fill=BOTH, expand=True,
                 padx=10, pady=10)
self.boton2.pack(side=RIGHT, fill=BOTH, expand=True,
                 padx=10, pady=10)
self.raiz.mainloop()

def calcular(self):

    # Función para validar datos y calcular importe a pagar

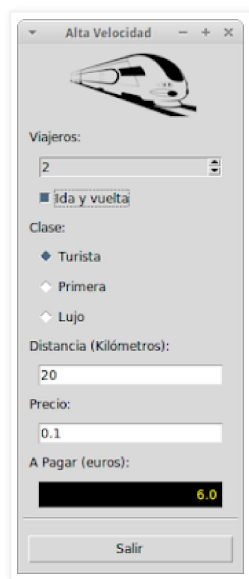
    error_dato = False
    total = 0
    try:
        km = int(self.km.get())
        precio = float(self.precio.get())
    except:
        error_dato = True
    if not error_dato:
        total = self.num_via.get() * km * precio
        if self.ida_vue.get():
            total = total * 1.5
        if self.clase.get() == 'p':
            total = total * 1.2
        elif self.clase.get() == 'l':
            total = total * 2
        self.total.set(total)
    else:
        self.total.set("¡ERROR!")

def main():
    mi_app = Aplicacion()
    return 0

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

Este sitio utiliza cookies de Google para prestar sus servicios y para analizar su tráfico. Tu dirección IP y user-agent se comparten con Google, junto con las métricas de rendimiento y de seguridad, para garantizar la calidad del servicio, generar estadísticas de uso y detectar y solucionar abusos.

[MÁS INFORMACIÓN](#) [ACEPTAR](#)



```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from tkinter import *
from tkinter import ttk

# Calcula coste de viaje con validación y cálculo inmediato

class Aplicacion():
    def __init__(self):
        self.raiz = Tk()
        self.raiz.title("Alta Velocidad")

        # Declara variables de control

        self.num_via = IntVar(value=1)
        self.ida_vue = BooleanVar()
        self.clase = StringVar(value='t')
        self.km = IntVar(value=1)
        self.precio = DoubleVar(value=0.10)
        self.total = DoubleVar(value=0.0)

        # Define trazas con variables de control de los widgets Entry()
        # para detectar cambios en los datos. Si se producen cambios
        # se llama a la función 'self.calcular' para validación y para
        # calcular importe a pagar

        self.km.trace('w', self.calcular)
        self.precio.trace('w', self.calcular)

        # Llama a función para validar y calcular

        self.calcular()

        # Carga imagen para asociar a widget Label()

        tren = PhotoImage(file='tren-128x64.png')

        # Declara widgets de la ventana
        # En los widgets de tipo Spinbox, Checkbutton y Radiobutton
        # se utiliza la opción 'command' para llamar a la función
        # 'self.calcular' para validar datos y calcular importe a
        # pagar de forma inmediata

        self.imagen1 = ttk.Label(self.raiz, image=tren,
                                anchor="center")
        self.etiq1 = ttk.Label(self.raiz, text="Viajeros:")
        self.viaje = Spinbox(self.raiz, from_=1, to=20, wrap=True,
```

```
self.etiq2 = ttk.Label(self.raiz, text="Clase:")
self.clase1 = ttk.Radiobutton(self.raiz, text='Turista',
                              variable=self.clase, value='t',
                              command=self.calcular)
self.clase2 = ttk.Radiobutton(self.raiz, text='Primera',
                              variable=self.clase, value='p',
                              command=self.calcular)
self.clase3 = ttk.Radiobutton(self.raiz, text='Lujo',
                              variable=self.clase, value='l',
                              command=self.calcular)

self.etiq3 = ttk.Label(self.raiz,
                      text="Distancia (Kilómetros):")
self.dist = ttk.Entry(self.raiz, textvariable=self.km,
                      width=10)
self.etiq4 = ttk.Label(self.raiz, text="Precio:")
self.coste = ttk.Entry(self.raiz, textvariable=self.precio,
                      width=10)
self.etiq5 = ttk.Label(self.raiz, text="A Pagar (euros):")
self.etiq6 = ttk.Label(self.raiz, textvariable=self.total,
                      foreground="yellow", background="black",
                      borderwidth=5, anchor="e")
self.separ1 = ttk.Separator(self.raiz, orient=HORIZONTAL)

self.boton1 = ttk.Button(self.raiz, text="Salir",
                        command=quit)

self.imagen1.pack(side=TOP, fill=BOTH, expand=True,
                  padx=10, pady=5)
self.etiq1.pack(side=TOP, fill=BOTH, expand=True,
                padx=10, pady=5)
self.viaje.pack(side=TOP, fill=X, expand=True,
                padx=20, pady=5)
self.idavue.pack(side=TOP, fill=X, expand=True,
                 padx=20, pady=5)
self.etiq2.pack(side=TOP, fill=BOTH, expand=True,
                padx=10, pady=5)
self.clase1.pack(side=TOP, fill=BOTH, expand=True,
                 padx=20, pady=5)
self.clase2.pack(side=TOP, fill=BOTH, expand=True,
                 padx=20, pady=5)
self.clase3.pack(side=TOP, fill=BOTH, expand=True,
                 padx=20, pady=5)
self.etiq3.pack(side=TOP, fill=BOTH, expand=True,
                padx=10, pady=5)
self.dist.pack(side=TOP, fill=X, expand=True,
                padx=20, pady=5)
self.etiq4.pack(side=TOP, fill=BOTH, expand=True,
                padx=10, pady=5)
self.coste.pack(side=TOP, fill=X, expand=True,
                 padx=20, pady=5)
self.etiq5.pack(side=TOP, fill=BOTH, expand=True,
                 padx=10, pady=5)
self.etiq6.pack(side=TOP, fill=BOTH, expand=True,
                 padx=20, pady=5)
self.separ1.pack(side=TOP, fill=BOTH, expand=True,
                 padx=5, pady=5)
self.boton1.pack(side=RIGHT, fill=BOTH, expand=True,
                 padx=10, pady=10)
self.raiz.mainloop()

def calcular(self, *args):

    # Función para validar datos y calcular importe a pagar

    error_dato = False
    total = 0
    try:
        km = int(self.km.get())
        precio = float(self.precio.get())
    except:
        error_dato = True
    if not error_dato:
        total = self.num_via.get() * km * precio
        if self.ida_vue.get():
            total = total * 1.5
```

Este sitio utiliza cookies de Google para prestar sus servicios y para analizar su tráfico. Tu dirección IP y user-agent se comparten con Google, junto con las métricas de rendimiento y de seguridad, para garantizar la calidad del servicio, generar estadísticas de uso y detectar y solucionar abusos.

MÁS INFORMACIÓN ACEPTAR

```
def main():  
    mi_app = Aplicacion()  
    return 0  
  
if __name__ == '__main__':  
    main()
```

Anterior: [Tkinter: tipos de ventanas](#)

Siguiente: [Menús, barras de herramientas y de estado en Tkinter](#)

[Ir al índice del tutorial de Python](#)

Publicado por Pherkad en [4:51](#)

Etiquetas: [Python3](#), [Tkinter](#)

[Entrada más reciente](#)

[Inicio](#)

[Entrada antigua](#)