BICENTENARIO PROVINCIA DE BUENOS AIRES

INSTITUTO PROVINCIAL DE LA ADMINISTRACIÓN PÚBLICA (IPAP)

2022

IPAP | SUBSECRETARÍA DE EMPLEO PÚBLICO Y GESTIÓN DE BIENES | MINISTERIO DE JEFATURA DE GABINETE DE MINISTROS

GOBIERNO DE LA PROVINCIA DE BUENOS AIRES

DESARROLLO DE APLICACIONES DE ESCRITORIO PYTHON

Nombre y apellido docente: Diego Lanciotti

Año 2022

Contenido del Curso

PySimpleGUI™

Módulo 1

Historia de Python

Instalación de Python

Frameworks

Primera aplicación

Módulo 2

Layouts

Eventos

Ventanas persistentes

Elementos básicos y temas

Módulo 3

Estructura de un proyecto.

Archivos json

Keys para identificar elementos

Módulo 4

Componentes más usados

Actualizar componentes de forma dinámica

Módulo 5

Ventajas y desventajas de las aplicaciones de escritorio

Crear ejecutables

Recomendaciones

Módulo 2 - Primeras aplicaciones y elementos

Creación de ventanas

LINK AL MATERIAL DEL MÓDULO 2:
https://github.com/cosme12/DESARROLLO-DE-APLICACIONES-DE-ESCRITORIO-PYTHON/tree/modulo2

PySimpleGUI™

Creación de ventanas

PySimpleGUI se basa en una estructura de ventanas, donde cada una de estas tiene su “layout” con los elementos que la componen. Múltiples ventanas (o solo una) forman una aplicación.

El objeto window tiene muchas propiedades, como por ej: tamaño, estilo, título, margen, etc.

Podés encontrar la lista completa de atributos en la documentación: https://www.pysimplegui.org/en/latest/call%20reference/#window-the-window-object

PySimpleGUI™

Layout

Un Layout contiene todos los elementos de una ventana, como por ejemplo texto, botones o imágenes.

Un layout es una lista de listas. Podemos imaginar el layout de una ventana como un conjunto de filas, donde cada elemento de cada fila se renderiza en la ventana en el orden que los agregamos.

Ej: fila1 = texto

fila2 = campo para ingresar datos

fila3 = texto, boton, texto

PySimpleGUI™

```
# Definimos el contenido de la ventana
layout = [  [sg.Text("Como te llamas?")],     # Parte 2 - El Layout
            [sg.Input()],
            [sg.Button('Ok')] ]

# Creamos la ventana
window = sg.Window('Titulo de la ventana', layout)  # Parte 3 - Definimos la ventana
```

Layout

En el ejemplo anterior podemos ver una ventana, donde cada elemento se renderiza de forma ordenada siguiendo la estructura de filas y columnas definido en el layout de la misma.

Una vez definido el layout, creamos la ventana con su título y le asignamos el layout a la misma.

PySimpleGUI™

Ejemplo completo

Veamos un ejemplo completo de como trabajar con ventanas:

1. Importamos PySimpleGUI
2. Definimos el layout de nuestra ventana
3. Creamos la ventana
4. Dibujamos y leemos los eventos de la ventana
5. Imprimimos un mensaje y cerramos la ventana

Ver app.py en el repositorio del material de módulo 2

PySimpleGUI™

```python
import PySimpleGUI as sg                        # Parte 1 - Importamos

# Definimos el contenido de la ventana
layout = [  [sg.Text("Como te llamas?")],     # Parte 2 - El Layout
            [sg.Input()],
            [sg.Button('Ok')] ]

# Creamos la ventana
window = sg.Window('Titulo de la ventana', layout)  # Parte 3 - Definimos la ventana

# Dibujamos la ventana y leemos los eventos
event, values = window.read()                   # Parte 4 - Loop de eventos

# Hacemos algo con la información obtenida
print('Hola', values[0], "! Gracias por sumarte al curso de PySimpleGUI")

# Cerramos la ventana
window.close()                                  # Parte 5 - Cerramos la ventana
```

Eventos

Al ejecutar el código anterior se nos abre una ventana que nos solicita nuestro nombre y un botón OK para enviar los datos.

Todas las acciones en una ventana quedan guardados dentro las variables “event” y “values”.

Si imprimimos el contenido de estas variables al momento de enviar los datos vamos a ver lo siguiente:

event = “Ok”

values = {0: “diego”}

PySimpleGUI™

Eventos

Los eventos nos permiten identificar acciones que se generen en nuestra ventana, como por ejemplo el click sobre un botón, la seleccion de un elemento en una lista, click derecho, etc.

Los values almacenan los datos actuales de todos los campos de una ventana en forma de diccionario.

PySimpleGUI™

Ventanas persistentes

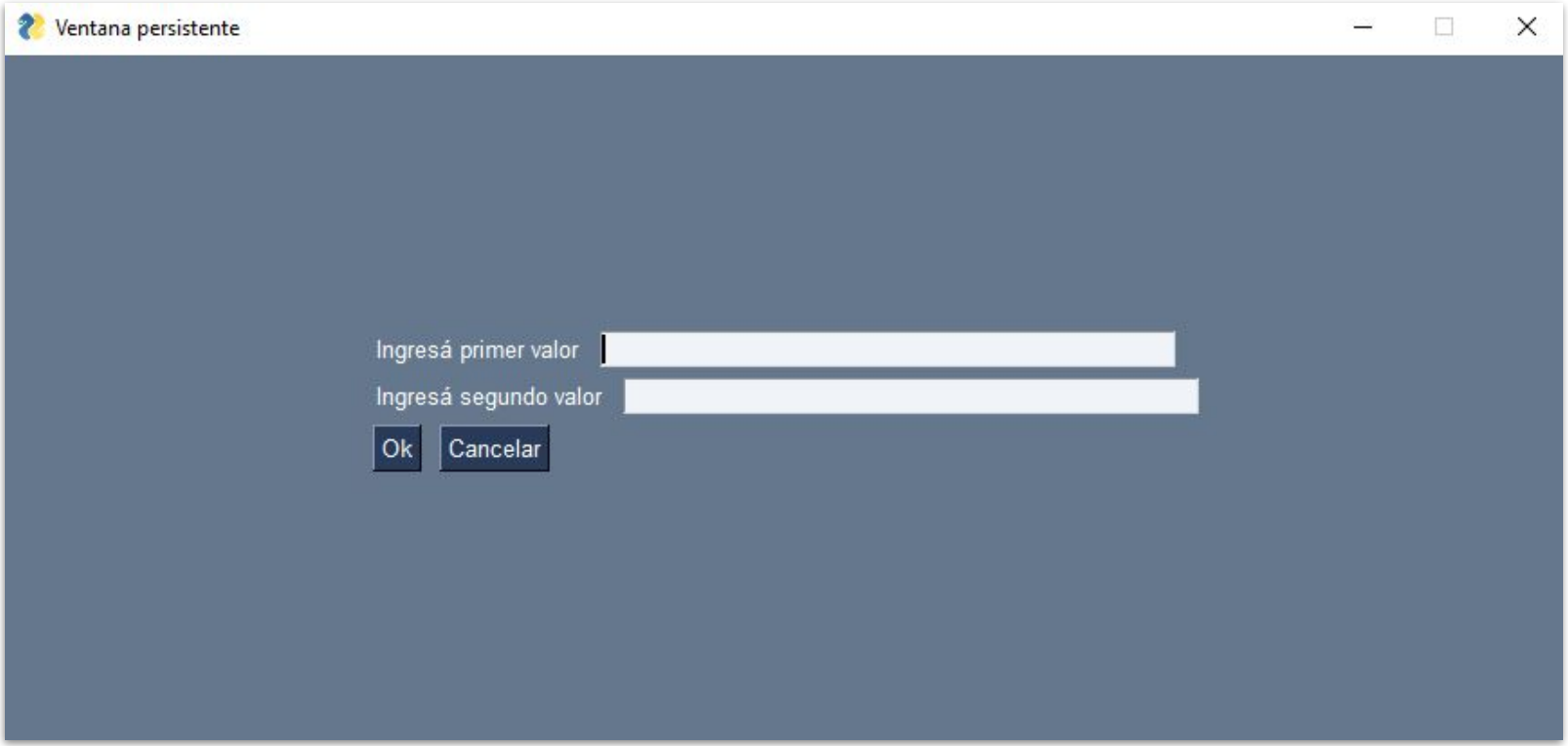
Hasta ahora nuestras ventanas solo pueden reaccionar a un solo evento. Para solucionar esto, vamos a crear ventanas persistentes que se mantengan abiertas y respondan a las acciones del usuario sin cerrarse.

Ver ventana_persistente.py en el repositorio del material de módulo 2

PySimpleGUI™

```python
import PySimpleGUI as sg

layout = [ [sg.Text('Ingresá primer valor'), sg.InputText()],
        [sg.Text('Ingresá segundo valor'), sg.InputText()],
        [sg.Button('Ok'), sg.Button('Cancelar')]
    ]

window = sg.Window("Ventana persistente", layout, margins=(200, 150))


while True:
    event, values = window.read()

    if event == "Cancelar" or event == sg.WIN_CLOSED:
        break

    print('Datos ingresados: ', values)

window.close()
```

PySimpleGUI™



Ventanas persistentes

De la misma forma que hicimos en el ejemplo anterior, definimos el layout y se lo asignamos a nuestra ventana.

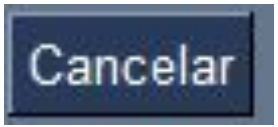
Luego creamos un loop infinito donde podamos leer los eventos y valores sin que la ventana se cierre. Noten que el programa no avanza hasta que “window.read()” no detecte algún evento.

Si el evento fue “Cancelar” entonces rompe el loop y cierra la ventana, sino, imprime el contenido de values.

PySimpleGUI™

Elementos básicos

Botones: *sg.Button("Cancelar")*



Texto: *sg.Text("Texto a mostrar")*

Input: *sg.InputText()*

PySimpleGUI™

Temas y estilos

Por defecto PySimpleGUI viene con un tema de botones y pantalla por defecto. Esto se puede configurar agregando esta línea antes de definir el layout de la ventana:

sg.theme(“SystemDefault”)

Dónde SystemDefault es el nombre del tema. La lista completa de tema puede encontrarse en:
https://user-images.githubusercontent.com/46163555/70382042-796da500-1923-11ea-8432-80d08cd5f503.jpg

PySimpleGUI™

Temas y estilos

PySimpleGUI™

Próximo módulo

PySimpleGUI™

En el siguiente módulo vamos a ver cómo **estructurar proyectos grandes, ventanas - controladores - handlers, convenciones, aplicaciones multiventanas** y trabajar con **archivos json**.

🚧 ¡Si tenés algún problema no dudes en consultar en el foro!

BICENTENARIO
PROVINCIA DE
BUENOS AIRES

ipap.gba.gob.ar

IPAP | SUBSECRETARÍA DE EMPLEO PÚBLICO Y GESTIÓN DE BIENES | MINISTERIO DE JEFATURA DE GABINETE DE MINISTROS

GOBIERNO DE LA PROVINCIA DE
BUENOS AIRES