

APLICACIÓN WEB INTERACTIVA CON PYTHON

CONTENIDO

Que estaremos usando.....	2
Instalación de PyWebIO	2
Crear un entorno virtual: Ejecutar en la terminal	2
Seleccione y active un entorno #	2
Instalar PyWebIO.....	4
Extensiones para trabajar con Visual Studio Code	4
Trabajando con Pywebio	4
Producción.....	5
Crear un generador de PDF	7
Visualización de la información recibida del usuario	9
Instalar fpdf	12
Agregar botones y generar el pdf.....	12
Conclusión	17

QUE ESTAREMOS USANDO

PyWebIO es una biblioteca de Python que proporciona herramientas de entrada y salida que nos permiten escribir fácilmente aplicaciones web interactivas. Hace que tomar la entrada del usuario y manejarla sea muy simple. También simplifica la salida. Según los documentos de PyWebIO:

PyWebIO proporciona una serie de funciones imperativas para obtener la entrada y salida del usuario en el navegador, convirtiendo el navegador en un "terminal de texto enriquecido", y se puede utilizar para crear aplicaciones web simples o aplicaciones GUI basadas en navegador.

PyWebIO se puede utilizar para crear rápidamente aplicaciones simples que no requieren una interfaz de usuario compleja.

INSTALACIÓN DE PYWEBIO

Recuerda que debes haber elegido un entorno. Para saber más:

<https://code.visualstudio.com/docs/python/environments>

CREAR UN ENTORNO VIRTUAL: EJECUTAR EN LA TERMINAL

Para crear un entorno virtual, en la terminal de VSCode utilice el siguiente comando, donde ".venv" es el nombre de la carpeta del entorno:

```
# macOS/Linux

# You may need to run sudo apt-get install python3-venv first

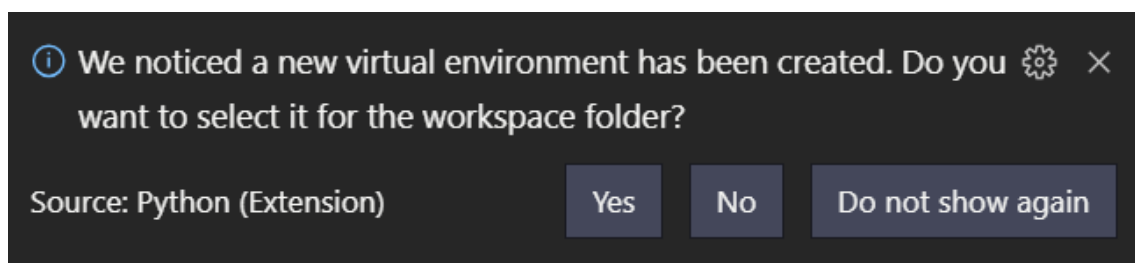
python3 -m venv .venv


# Windows

# You can also use py -3 -m venv .venv

python -m venv .venv
```

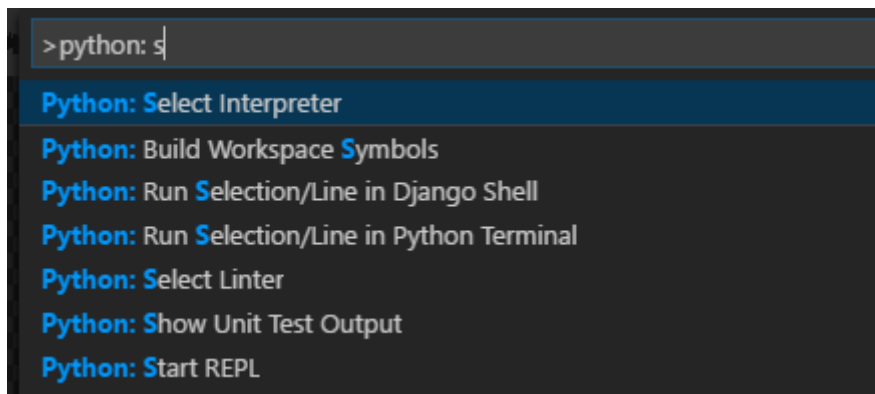
Cuando cree un nuevo entorno virtual, se mostrará un mensaje que le permitirá seleccionarlo para el espacio de trabajo.



Esto agregará la ruta al intérprete de Python desde el nuevo entorno virtual a la configuración de su espacio de trabajo.

SELECCIONE Y ACTIVE UN ENTORNO

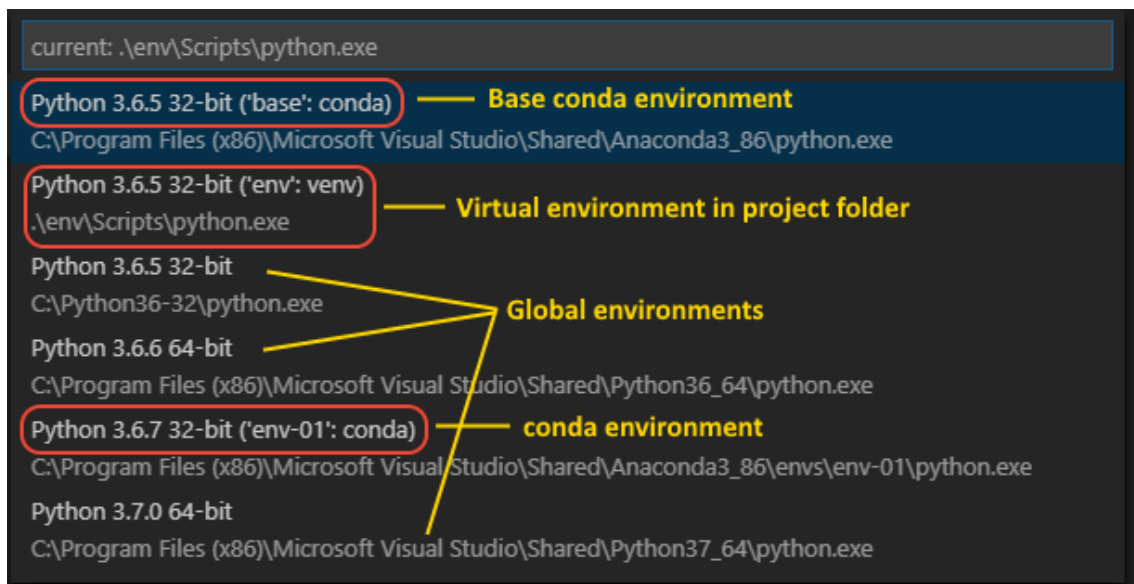
De forma predeterminada, la extensión de Python busca y usa el primer intérprete de Python que encuentra en la ruta del sistema. Para seleccionar un entorno específico, use el comando **Python: Seleccionar intérprete** de la **paleta de comandos** (**Ctrl + Shift + P**).



Nota : Si la extensión de Python no encuentra un intérprete, emite una advertencia. En macOS, la extensión también emite una advertencia si está utilizando el intérprete de Python instalado en el sistema operativo, porque normalmente desea utilizar un intérprete que instala directamente.

Puede cambiar de entorno en cualquier momento; El cambio de entornos le ayuda a probar diferentes partes de su proyecto con diferentes intérpretes o versiones de biblioteca según sea necesario.

El comando **Python: Seleccionar intérprete** muestra una lista de entornos globales, entornos conda y entornos virtuales disponibles. (Consulte la sección [Dónde busca entornos la extensión](#) para obtener detalles, incluidas las distinciones entre estos tipos de entornos). La siguiente imagen, por ejemplo, muestra varias instalaciones de Anaconda y CPython junto con un entorno conda y un entorno virtual (**env**) que se encuentra dentro la carpeta del espacio de trabajo:

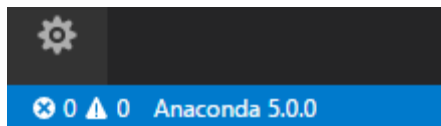


Nota: En Windows, VS Code puede tardar un poco en detectar los entornos de conda disponibles. Durante ese proceso, es posible que vea "(en caché)" antes de la ruta a un entorno. La etiqueta indica que VS Code está trabajando actualmente con información almacenada en caché para ese entorno.

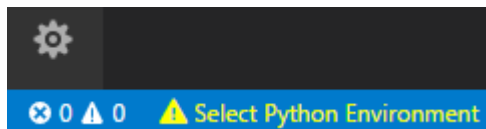
Si tiene una carpeta o un espacio de trabajo abierto en VS Code y selecciona un intérprete de la lista, la extensión de Python almacenará esa información internamente para que se use el mismo intérprete una vez que vuelva a abrir el espacio de trabajo.

La extensión Python usa el entorno seleccionado para ejecutar código Python (usando el comando **Python: Run Python File in Terminal**), proporcionando servicios de lenguaje (autocompletar, verificación de sintaxis, linting, formateo, etc.) cuando tiene un `.py` archivo abierto en el editor y abriendo una terminal con el comando **Terminal: Create New Terminal**. En este último caso, VS Code activó automáticamente el entorno seleccionado.

La barra de estado siempre muestra el intérprete actual.



La barra de estado también refleja cuando no se selecciona ningún intérprete.



En cualquier caso, hacer clic en esta área de la barra de estado es un atajo conveniente para el comando **Python: Seleccionar intérprete**.

INSTALAR PYWEBIO

```
pip install -U pywebio
```

EXTENSIONES PARA TRABAJAR CON VISUAL STUDIO CODE

Además para nuestro Visual Studio Code necesitaremos tener instaladas las siguientes herramientas:

<https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer>

<https://marketplace.visualstudio.com/items?itemName=formulahendry.code-runner>

<https://marketplace.visualstudio.com/items?itemName=Zignd.html-css-class-completion>

<https://marketplace.visualstudio.com/items?itemName=ms-toolsai.jupyter>

<https://marketplace.visualstudio.com/items?itemName=ms-python.vscode-pylance>

<https://marketplace.visualstudio.com/items?itemName=VisualStudioExptTeam.vscodintellicode>

TRABAJANDO CON PYWEBIO

Es muy fácil registrar varias formas de respuestas del usuario usando PyWebIO. A continuación, se muestran algunas de las formas en las que puede realizar entradas con PyWebIO.

```

from pywebio.input import *

input("This is a simple text input")

select("This is a drop down menu", ['Option1', 'Option2'])

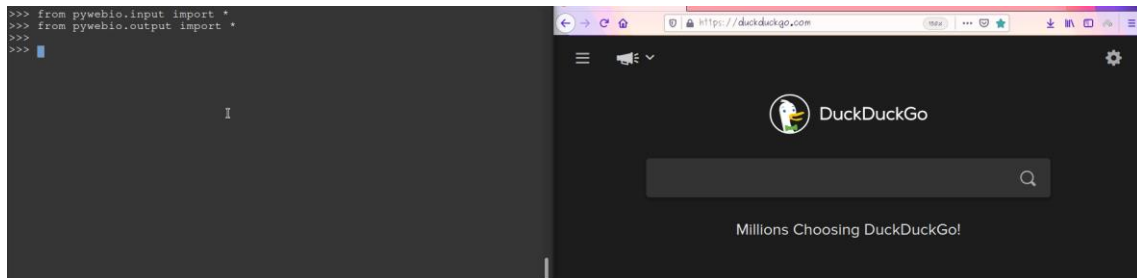
checkbox("Multiple Choices!", options=['a', 'b', 'c', 'd'])

radio("Select any one", options=['1', '2', '3'])

textarea('Text Area', rows=3, placeholder='Multiple line text input')

```

El GIF a continuación muestra cómo se ven estas opciones.



PRODUCCIÓN

PyWebIO también proporciona varias opciones de salida. Algunos de ellos se muestran a continuación.

```

from pywebio.output import *

from pywebio import session

put_text("Hello friend!")

put_table([
    ['Object', 'Unit'],
    ['A', '55'],
    ['B', '73'],
])

```

```
put_markdown('~~~PyWebIO~~~')
```

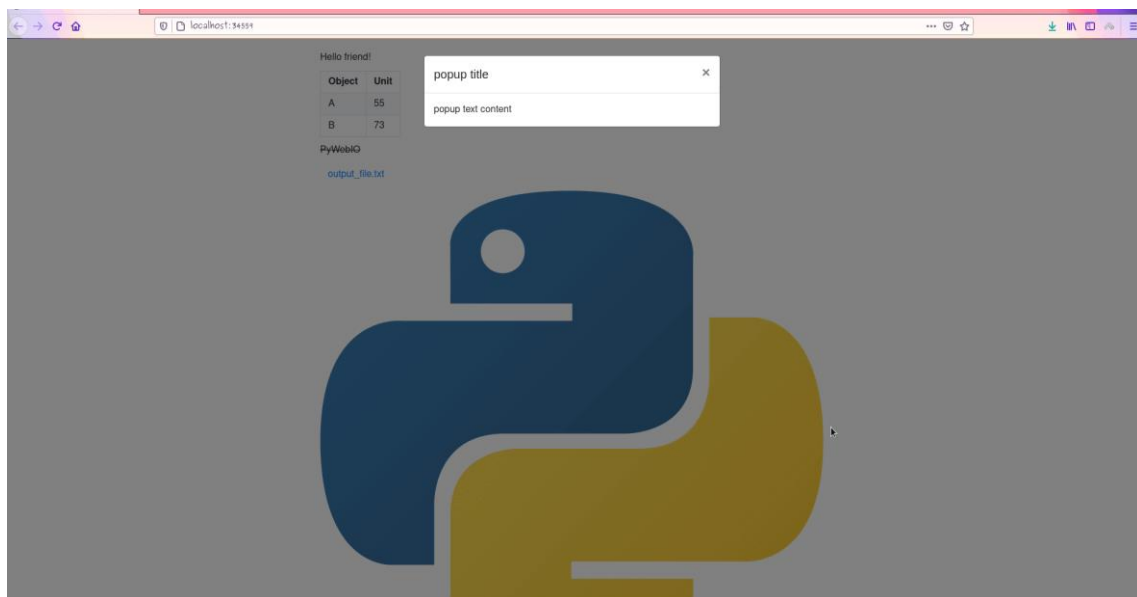
```
put_file('output_file.txt', b'You can put anything here')
```

```
put_image(open('python_logo.png', 'rb').read())
```

```
popup('popup title', 'popup text content')
```

```
session.hold()
```

Así es como se ve el código anterior cuando se ejecuta.



En el código anterior, usamos:

- `put_text` para crear un texto plano
- `put_table` para crear una mesa
- `put_markdown` escribir rebajas
- `put_file` para generar un enlace para descargar un archivo
- `put_image` para dar salida a la imagen
- `popup` para mostrar una ventana emergente
- `session.hold` Se necesita una función para mantener la sesión abierta. Sin él, el servidor se desconecta después de poner todos los resultados y el archivo no se puede descargar.

Es sorprendente lo mucho que podemos lograr con unas pocas líneas de código. Ahora, para comprender mejor esta biblioteca, crearemos una aplicación simple.

La aplicación:

- Tomar información del usuario.
- Guárdelo en un archivo pdf en un formato agradable y presentable.
- Permita que el usuario descargue el pdf y lo guarde.

CREAR UN GENERADOR DE PDF

En primer lugar, crearemos un formulario que puede tomar información del usuario.

```
from pywebio.input import *

from pywebio.output import *

from pywebio import start_server

def app():

    put_markdown('# Professional Info')

    #group all the inputs to make a form
    data = input_group("Basic info",[

        input('Input your name', name='name', required=True),

        checkbox("Languages of choice", ['JAVA', 'Python', 'C', 'C++'], name='loc'),

        select("Experience", ['0 - 1', '1 - 2', '2+'], name='yoe', required=True),

        radio("Gender", options=['Male', 'Female', 'Other'],name='gndr', required=True),

        textarea('Tell something about yourself', rows=3, name='abt', required=True),

        file_upload('Profile Image',placeholder='Choose file',accept='image/*',name='dp',
        required=True)

    ])

    ))
```

```
# main function

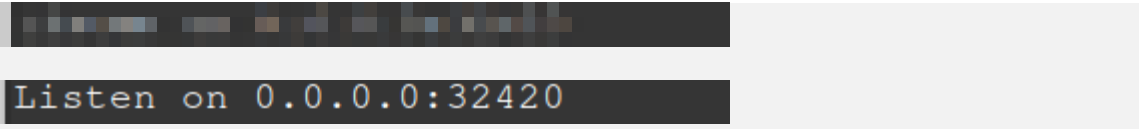
if __name__ == '__main__':

    start_server(app, port=32420, debug=True)
```

Explicación del código anterior:

- `input_group`: nos permite tomar múltiples entradas juntas. El usuario envía solo una vez y todos los datos se devuelven en forma de un diccionario de Python cuya clave es el nombre del elemento de entrada y cuyo valor es el valor del elemento de entrada.
- `input` : crea un cuadro de entrada de texto.
- `name`: nombre del elemento de entrada. Se utilizará para acceder al elemento de datos del diccionario devuelto por `input_group`.
- `required` : especifica si la entrada es necesaria.
- `checkbox`: crea una entrada de opción múltiple. Se pueden seleccionar varios valores.
- `select`: crea un menú desplegable.
- `radio`: crea un botón de opción.
- `textarea` : crea un cuadro de entrada de texto de varias líneas.
- `file_upload`: permite al usuario cargar un archivo.
- `accept`: esto le dice al programa qué tipos de archivos aceptar
- `start_server`: Inicie un servidor para proporcionar la aplicación PyWebIO como un servicio web. `debug=True` le dice al servidor que se recargue automáticamente cuando cambie el código.

Al ejecutar el código, deberíamos ver algo como a continuación en la terminal:



```
Listen on 0.0.0.0:32420
```

Vaya a <http://0.0.0.0:32420/> para jugar con la aplicación. ¡Ahora tenemos una aplicación web simple con un formulario de entrada como el siguiente!

Professional Info

Basic info

Input your name

Languages of choice

☐ JAVA
☐ Python
☐ C
☐ C++

Experience

0 - 1

Gender

☐ Male
☐ Female
☐ Other

Tell something about yourself

Profile Image

Choose file

Browse

Submit

Reset

Ahora que sabemos cómo crear un formulario de entrada para tomar la entrada del usuario, anotemos el código que imprimirá toda la información insertada por el usuario de una manera agradable.

VISUALIZACIÓN DE LA INFORMACIÓN RECIBIDA DEL USUARIO

```
from pywebio.input import *

from pywebio.output import *

from pywebio import start_server


def app():

    put_markdown('# Professional Info')

    #group all the inputs to make a form

    data = input_group("Basic info",[

        input('Input your name', name='name', required=True),
```

```
checkbox("Languages of choice", ['JAVA', 'Python', 'C', 'C++'], name='loc'),

select("Experience", ['0 - 1', '1 - 2', '2+'], name='yoe', required=True),

radio("Gender", options=['Male', 'Female', 'Other'], name='gndr', required=True),

textarea('Tell something about yourself', rows=3, name='abt', required=True),

file_upload('Profile Image', placeholder='Choose file', accept='image/*', name='dp',
required=True)

))
```

```
#extracting image from input
```

```
img = data['dp']
```

```
with open('img.png', 'wb') as file:
```

```
    file.write(img['content'])
```

```
#display image
```

```
put_image(img['content'])
```

```
put_markdown("____")
```

```
# making a list of chosen languages
```

```
loc = "<ul>"
```

```
for i in data['loc']:
```

```
    loc += '<li>' + i + '</li>'
```

```
loc += '</ul>'
```

```
# formatting the data with html
```

```
res = f"""
```

```
<b>Name:</b> {data['name']}<br>
```

```
<b>Languages of choice:</b> {loc}<br>
```

```
<b>Experience:</b> {data['yoe']}<br>
```

```
<b>Gender:</b> {data['gndr']}<br>
```

```
<b>About:</b> {data['abt']}<br>"""
```

```
# displaying the data
```

```
put_html(res)
```

```
# main function
```

```
if __name__ == '__main__':
```

```
start_server(app, port=32420, debug=True)
```

Las líneas **21** a **26**, etiquetadas como "extraer imagen de la entrada", se utilizan para tomar la imagen cargada por el usuario y almacenarla en un archivo. Como se mencionó anteriormente, los datos devueltos por `input_group` se almacenan en un diccionario de Python llamado `data`. El `file_upload` mismo devuelve un diccionario, donde el contenido del archivo cargado se almacena bajo la `content` clave. No es necesario almacenar la imagen en el archivo si solo desea volver a mostrar la imagen o usarla solo en el programa, pero será necesario para generar el pdf, como veremos más adelante.

`put_img` se utiliza para mostrar la imagen cargada por el usuario.

El `checkbox` devuelve una lista de valores. Las líneas **31** a **34** se utilizan para poner estos valores en una lista desordenada HTML.

A continuación, en la línea **37** todos los datos se almacenan en una cadena junto con algún código HTML para formatearlo.

Luego `put_html` se usa para mostrar la información en la pantalla.

Después de que el usuario complete el formulario y presione el botón **Enviar**, la página se verá así:

Professional Info



Name: Apoorv Mishra

Languages of choice:

- JAVA
- Python
- C++

Experience: 0 - 1

Gender: Male

About: I am a python developer

Ahora que el usuario ha visto los datos, queremos preguntarle si quiere generar un pdf o no. En caso afirmativo, queremos darle un pdf que contenga toda su información con el formato adecuado.

INSTALAR FPDF

Para generar un pdf, usaremos la biblioteca PyFPDF. Recuerda que debes instalar en la terminal el fpdf

```
pip instalar fpdf
```

AGREGAR BOTONES Y GENERAR EL PDF

A continuación, se muestra el programa completo con el código para agregar botones para tomar la elección del usuario y luego generar un pdf a partir del mismo incluido.

```
from pywebio.input import *  
  
from pywebio.output import *  
  
from pywebio import start_server  
  
from fpdf import FPDF, HTMLMixin  
  
  
def app():
```

```
put_markdown('# Professional Info')
```

```
#group all the inputs to make a form
```

```
data = input_group("Basic info",[  
    input('Input your name', name='name', required=True),  
    checkbox("Languages of choice", ['JAVA', 'Python', 'C', 'C++'], name='loc'),  
    select("Experience", ['0 - 1', '1 - 2', '2+'], name='yoe', required=True),  
    radio("Gender", options=['Male', 'Female', 'Other'],name='gndr', required=True),  
    textarea('Tell something about yourself', rows=3, name='abt', required=True),  
    file_upload('Profile Image',placeholder='Choose file',accept='image/*',name='dp',  
required=True)  
])
```

```
#extracting image from input
```

```
img = data['dp']
```

```
with open('img.png', 'wb') as file:
```

```
    file.write(img['content'])
```

```
with open('img.png', 'rb') as file:
```

```
    img = file.read()
```

```
#display image
```

```
put_image(img)
```

```
put_markdown("____")
```

```
# making a list of chosen languages
```

```
loc = "<ul>"
```

```
for i in data['loc']:
```

```

        loc += '<li>' + i + '</li>'

loc += '</ul>'

# formatting the data with html

res = f"""
<b>Name:</b> {data['name']}<br>
<b>Languages of choice:</b> {loc}<br>
<b>Experience:</b> {data['yoe']}<br>
<b>Gender:</b> {data['gndr']}<br>
<b>About:</b> {data['abt']}<br>"""

# displaying the data

put_html(res)

# giving user option to choose whether pdf to be generated or not
choice = actions('Generate PDF?', ['Generate', 'Cancel'])

put_markdown("___")

if choice == 'Generate':

    file_name = input('Name of pdf file?')

    generate_pdf(res, file_name)

    put_markdown("***PDF file generated**")

else:

    put_markdown("PDF not generated")


# code to generate pdf

class MyFPDF(FPDF, HTMLMixin):

```

pass

```
def generate_pdf(data,file_name):

    pdf = MyFPDF()

    pdf.add_page()

    pdf.set_font('Arial', 'B', 18)

    pdf.cell(190, 15, 'Professional Info', ln=1, align='C')

    pdf.image('img.png', 85, 30, 40)

    pdf.line(10, 80, 200, 80)

    pdf.cell(60, 60, "", ln=2)

    pdf.write_html(data)

    pdf.output(f'{file_name}.pdf', 'F')


# main function

if __name__ == '__main__':

    start_server(app, port=32420, debug=True)
```

Al ejecutar el código, el formulario se mostrará como antes, pero después del envío, la página aparecerá algo como esto:

Professional Info

Basic info

Input your name

Apoorv Mishra

Languages of choice

☒ JAVA

☒ Python

☐ C

☒ C++

Experience

0 - 1

Gender

☒ Male

☐ Female

☐ Other

Tell something about yourself

I am a python developer.

Profile Image

apoorv.png


Browse

Submit

Reset

El archivo pdf generado se verá así:

Professional Info



Name: Apoorv Mishra

Languages of choice:

- JAVA
- Python
- C++

Experience: 0 - 1

Gender: Male

About: I am a python developer.

CONCLUSIÓN

Al usar PyWebIO, puede crear una aplicación web útil en poco tiempo.

Le animo a leer [la documentación de PyWebIO](#) para encontrar otras cosas interesantes que puede hacer con PyWebIO.