

Python Programming

Módulo 05

Automatizar envío de formularios

Automatizar envío de formulario HTML vía HTTP

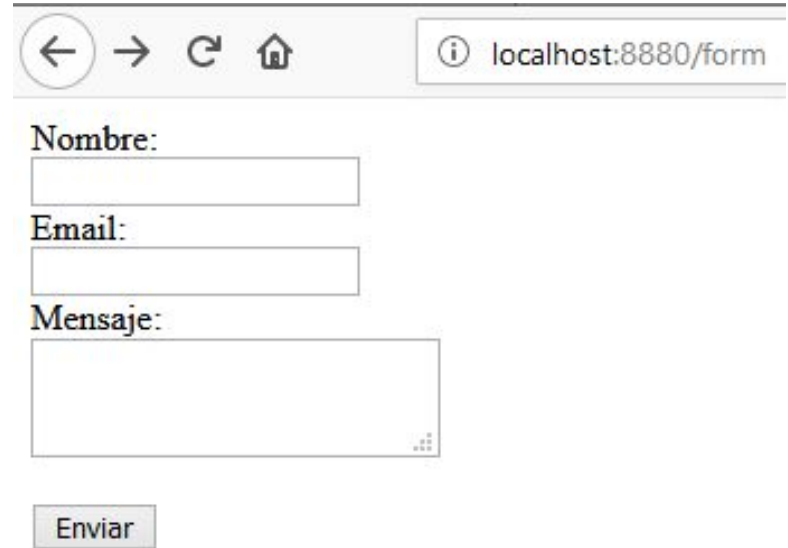
Los formularios son la principal forma de comunicación entre un usuario y un sitio web para enviar información. Por ejemplo, formularios de contacto o para iniciar sesión. Enviar información a través de un formulario usando Python es una operación muy común con el objetivo de automatizar tareas. La misma librería con la que hemos estado trabajando hasta ahora, `requests`, nos permite efectuar dicha operación. De hecho, el método HTTP que usan los formularios HTML para enviar información es `POST`.

Lo primero que necesitamos es un sitio con un formulario. Para eso nuevamente vamos a usar una aplicación creada para este ejemplo. La encuentras para descargar en el Alumni en esta misma sección con el nombre *formulario_web*.

Descargas `formulario_web.py` y guardas en una carpeta. Luego le haces doble click encima de `formulario_web.py`, se abrirá una terminal (*servidor*) muy parecido a lo que hicimos en la sección anterior con el servicio de alumnos.

(Solo en Linux o en iOS ejecutarlo desde la consola con `python3`).

Este pequeño sitio de ejemplo expone el siguiente formulario de contacto en <http://localhost:8880/form>.



The screenshot shows a web browser window with the address bar displaying `localhost:8880/form`. The browser's navigation bar includes back, forward, refresh, and home icons. The contact form consists of three input fields: 'Nombre:' (Name), 'Email:', and 'Mensaje:' (Message). Below these fields is a button labeled 'Enviar' (Send).

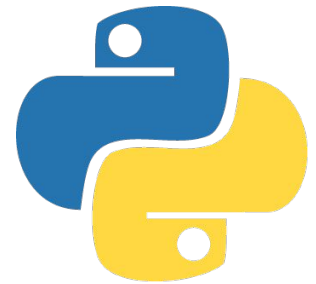
Para automatizar el envío, lo primero que tenemos que identificar es la dirección de URL. Luego, acabamos de decir que la información de un formulario HTML en un sitio web se envía a través del protocolo HTTP usando el método `POST`; entonces, nuestro código de Python comenzará siendo el siguiente:

```
import requests

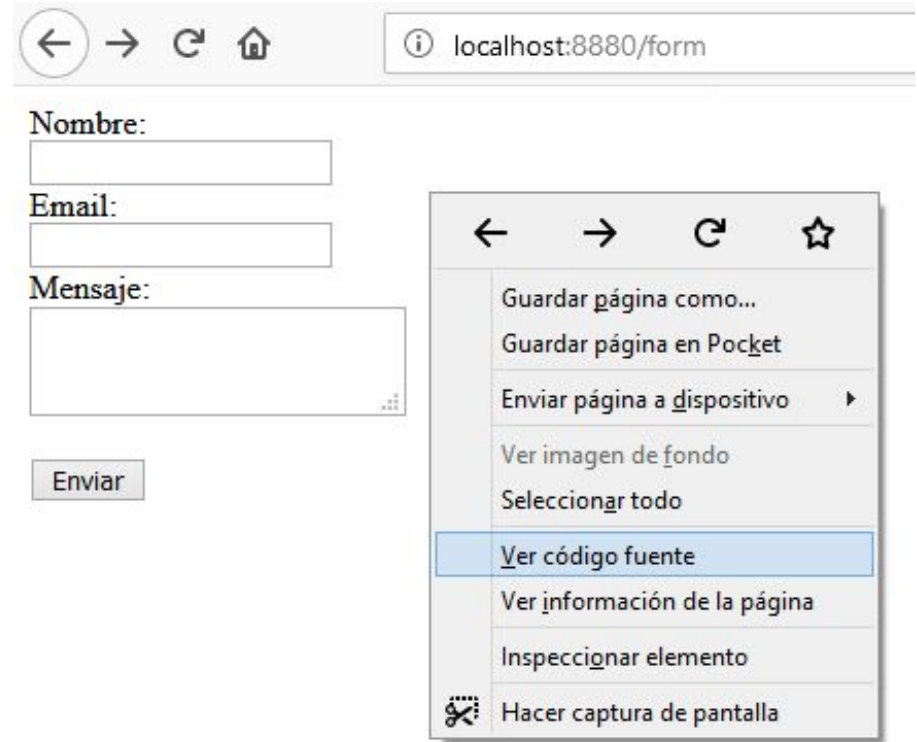
r =
requests.post("http://localhost:8880/form")
```

El siguiente paso es pasar como argumento a `post()` los datos que queremos enviar: el nombre, el correo electrónico y el mensaje mismo. Estos datos tienen que estar contenidos como valores de un diccionario, ¿pero cuáles serían las claves?

Para ello necesitamos ver el código de fuente del formulario. Todos los navegadores contienen una opción para ello en el menú contextual:



Para ello necesitamos ver el código de fuente del formulario. Todos los navegadores contienen una opción para ello en el menú contextual:



Se abrirá una nueva ventana y veremos el siguiente código:

```
<!DOCTYPE html>
<html>
<head><meta charset="utf-8"></head>
<body>
  <form method="post">
    Nombre:<br>
    <input type="text" name="name">
    <br>
    Email:<br>
    <input type="text" name="email">
    <br>
    Mensaje:<br>
    <textarea name="message"></textarea>
    <br><br>
    <input type="submit" value="Enviar">
  </form>
</body>
</html>
```

El fragmento de código que configura el formulario HTML está entre las etiquetas `<form>` y `</form>`. Lo que nos interesa dentro de él son los nombres que el sitio le ha asignado a los componentes del formulario (esto es, las cajas de texto para ingresar datos) vía el atributo `name`. Los marcamos en rojo a continuación:

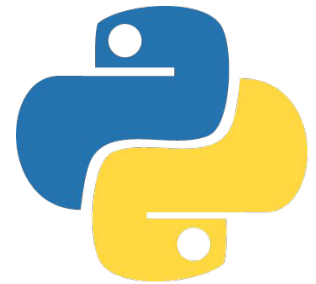
```
<!DOCTYPE html>
<html>
<head><meta charset="utf-8"></head>
<body>
  <form method="post">
    Nombre:<br>
    <input type="text" name="name">
    <br>
    Email:<br>
    <input type="text" name="email">
    <br>
    Mensaje:<br>
    <textarea name="message"></textarea>
    <br><br>
    <input type="submit" value="Enviar">
  </form>
</body>
</html>
```


Vemos que los nombres para los campos de nombre, correo electrónico y mensaje son, respectivamente, `name`, `email` y `message`. Pues bien, estos corresponderán, entonces, a las clave de nuestro diccionario de datos:

```
datos = {  
    "name": "Mariano",  
    "email": "mariano@ejemplo.com",  
    "message": "¡Hola, mundo!"  
}  
r = requests.post("http://localhost:8880/form", data=datos)
```

(Aquí el argumento para `post()` es `data` y no `json`, a diferencia del apartado anterior, ya que al enviar datos en un formulario raramente se codifican en formato JSON).

¡Perfecto! Ahora, ¿cómo sabemos que el formulario se ha enviado correctamente? Primero intentemos enviarlo manualmente usando el navegador. Luego de completar los datos veremos que la página responde con lo siguiente:



Podemos acceder a la respuesta del sitio desde Python vía `r.text`:

```
datos = {  
    "name": "Mariano",  
    "email": "mariano@ejemplo.com",  
    "message": "¡Hola, mundo!"  
}  
  
r = requests.post("http://localhost:8880/form", data=datos)  
contenido = r.text  
print(contenido)
```

Esto imprime:

```
<!doctype html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <title>Enviado</title>
    <meta http-equiv="refresh" content="3";URL="http://localhost:8880/form">
  </head>
  <body>
    Mensaje enviado
  </body>
</html>
```

Eso quiere decir que podemos hacer una comprobación desde nuestro código para determinar si el formulario se envió correctamente, igualando `contenido` a la cadena anterior. Pero dado que el resultado en HTML suele ser muy voluminoso, simplemente podemos buscar alguna parte de la cadena, por ejemplo, "Mensaje enviado". Para ello usamos el operador `in`, como vimos en las colecciones:

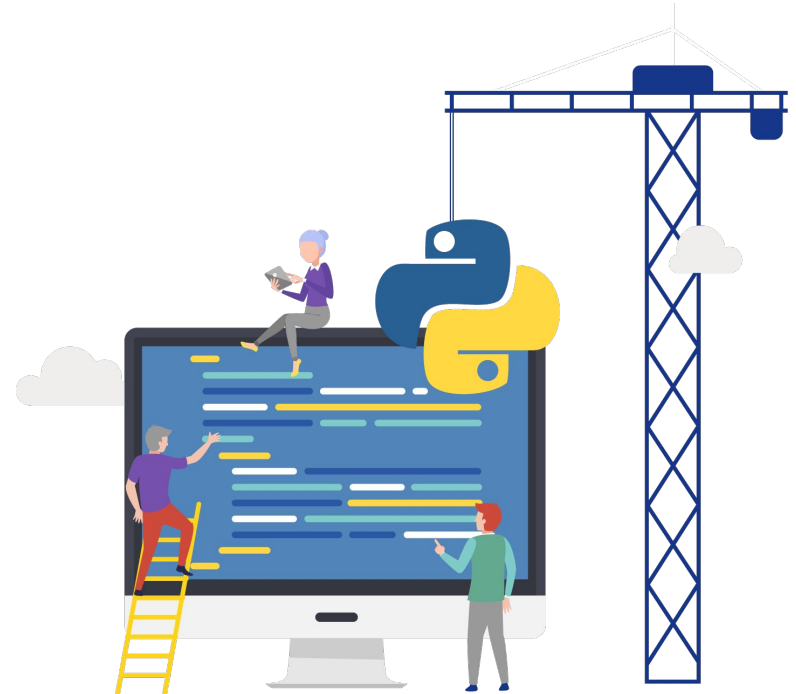
```
datos = {
    "name": "Mariano",
    "email": "mariano@ejemplo.com",
    "message": "¡Hola, mundo!"
}
r = requests.post("http://localhost:8880/form",
data=datos)

contenido = r.text

if "Mensaje enviado" in contenido:
    print("¡Formulario enviado!")
else:
    print("Ocurrió un error.")
```

O bien usando la función `find()` según lo visto en la sección de operaciones sobre cadenas:

```
if contenido.find("Mensaje enviado") > -1:  
    print(";Formulario enviado!")  
else:  
    print("Ocurrió un error.")
```



¡Sigamos trabajando!