

Haciendo pruebas sobre Splinter

Esta clase analizaremos cómo automatizar el navegador web, algo que nos puede permitir realizar carga de datos masiva o pruebas de integración.

La herramienta que utilizaremos se llama *Splinter*, y básicamente funciona como un objeto python que hace de intermediario con el navegador.

Modos de uso de splinter

Existen dos grandes casos de uso para una herramienta como splinter:

- Automatizar pruebas de funcionamiento sobre una aplicación web. Ideal para proyectos en pleno desarrollo.
- Utilizarla como herramienta de automatización de tareas: cargar cantidades muy grandes de información en el navegador, reducir tareas repetitivas, extraer datos del navegador automáticamente etc...

En cualquiera de los dos casos, podremos crear un script de python con todas las instrucciones que debe seguir el navegador.

Instalación

Splinter es un módulo externo, así que tenemos que instalarlo con un comando del sistema:

```
sudo pip install splinter
```

Si este comando no funciona, posiblemente necesites ejecutar estos dos comandos antes:

```
sudo apt-get install python-setuptools  
sudo easy_install pip
```

Inicialización

Para comenzar, podrías abrir un intérprete de python e inicializar el objeto *Browser*:

```
import splinter
b = splinter.Browser()
```

La segunda sentencia abrirá una instancia del navegador **Firefox**, que se podrá controlar con el objeto que tiene de nombre *b*. Si quieres abrir otro navegador en su lugar, como *chrome*, podrías escribir

```
b = splinter.Browser('chrome')
```

Ten en cuenta que esta sentencia puede tardar unos segundos en responder...

Nota: en huayra linux notamos que necesitas ejecutar el comando

```
sudo apt-get install iceweasel
```

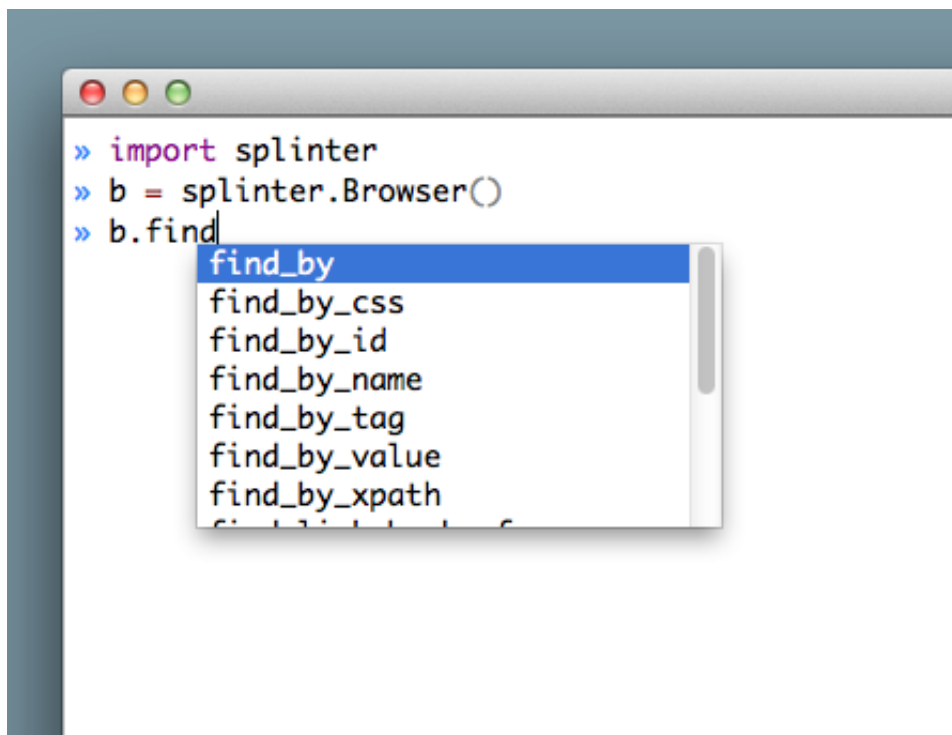
 antes de inicializar *splinter*.

Explorando

Hay varias formas de comenzar a familiarizarse con una biblioteca como *sprinter*. Lo primero que debemos tener en cuenta es que obtuvimos un objeto, llamado `b`, así que podemos conocer sus atributos usando la función `dir` o `help`:

```
dir(b)
help(b)
```

Incluso intérpretes cómo `ipython`, `idle` o `lanas` te podrían servir de utilizar gracias a su autocompletado:



```
>> import splinter
>> b = splinter.Browser()
>> b.find
```

- find_by
- find_by_css
- find_by_id
- find_by_name
- find_by_tag
- find_by_value
- find_by_xpath

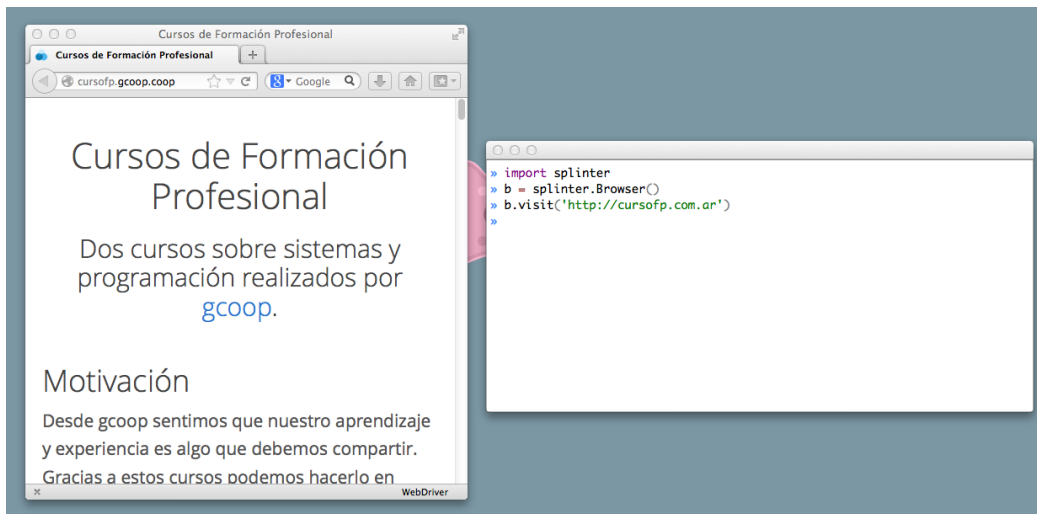
Mensajes básicos

Vamos a construir un script de python que visite el sitio del curso, luego haga click en un link con el texto "gcoop" y que por último haga una búsqueda.

Comencemos diciéndole a sprinter que visite el sitio del curso:

```
b.visit('http://cursofp.com.ar')
```

Listo, en pantalla deberías ver que el navegador cambió de ruta:



Clicks

Para poder hacer clicks sobre elementos, tenemos que realizar dos pasos:

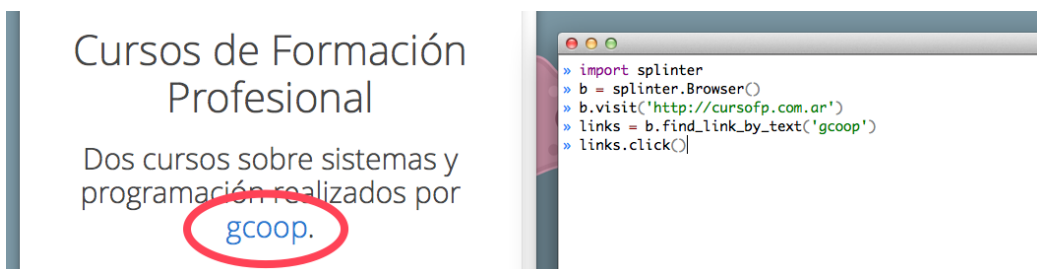
- Buscar y obtener la referencia a los elementos que queremos pulsar.
- Usar la referencia y enviarle el mensaje "click".

Realizaremos estos pasos en dos líneas de código:

```
links = b.find_link_by_text('scoop')
links.click()
```

La primer línea identifica todos los links dentro de la página que dicen exactamente 'gcoop'.

Mientras que la segunda línea hace click sobre el primero de los elementos que identificó.



Ten en cuenta que el objeto `link` sirve mientras no cambiemos de página. Si le envías el mensaje `click`, la página cambia. *splinter* no te dejará

hacer nada mas con ese objeto `link`. Tendrías que volver a generarlo.

Completando campos de texto

Ejecutando el código anterior, el navegador nos tendría que mostrar el sitio de *gcoop*.

Observa que en la parte superior derecha aparece un cuadro de texto con un botón al lado que dice *buscar*.



Es decir, si estuviéramos usando el mouse y el teclado, la forma de realizar una búsqueda en este sitio es hacer click con el mouse sobre el campo de texto, escribir lo que queremos buscar y luego pulsar el botón "Buscar".

Hagamos esto desde el mismo intérprete:

Necesitamos identificar esa caja de texto, pero a diferencia del link que pulsamos anteriormente, aquí no tenemos texto para buscar, el elemento HTML está vacío, necesitamos identificarlo de otra forma.

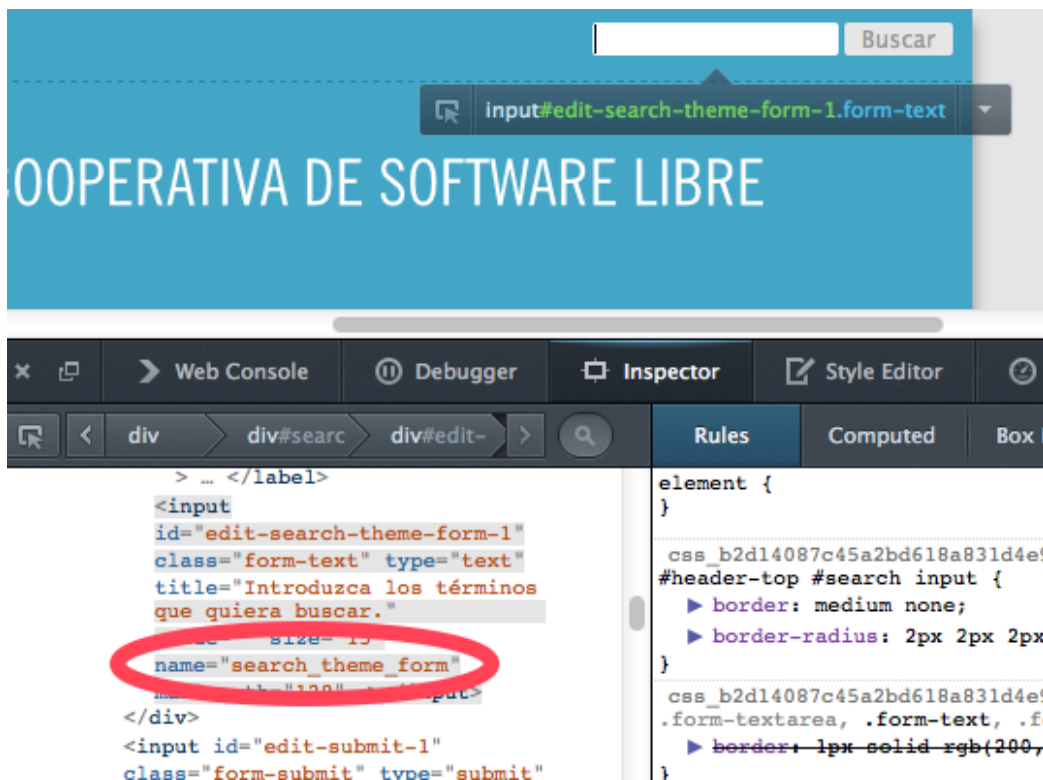
En los sitios webs se suele dar un nombre a cada componente de forma que se lo pueda identificar, así que podemos usar la herramienta de inspección del navegador para llegar exactamente al nombre del elemento:

Pulsa el botón derecho sobre el campo de texto y selecciona "Inspeccionar elemento".

Lo que tendrías que ver en pantalla es el código HTML de ese elemento en particular, y ahí observarás que el nombre de ese componente es

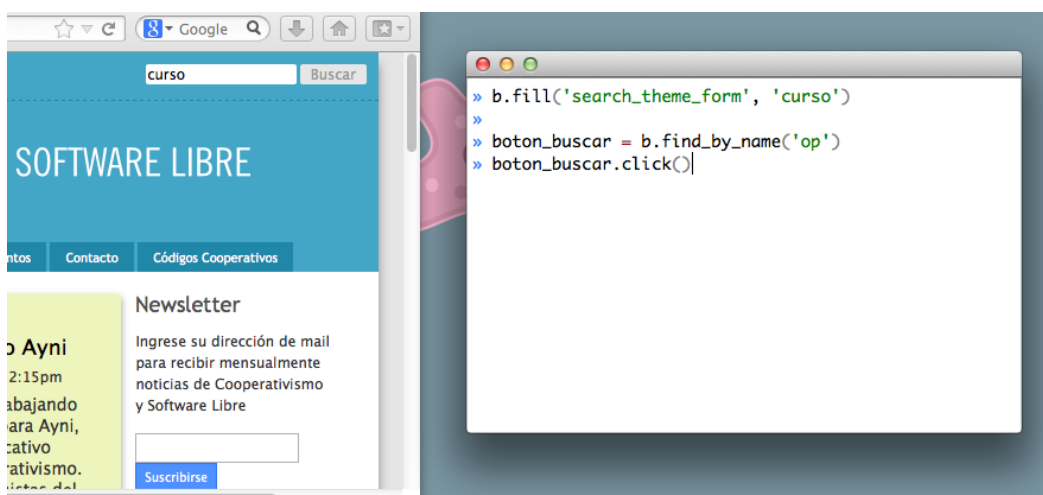
`search theme form`. Ese es el nombre que nos permite identificar al

elemento:



Ahora, solo tenemos que decirle a sprinter que cargue texto en ese elemento y luego pulse el botón buscar:

```
boton_buscar = b.find_by_name('op')
boton_buscar.click()
```



Recursos

Vimos solo unos poquitos métodos de splinter, pero la funcionalidad de splinter es mucho mayor.

Te recomendamos crear un script de automatización para algún sitio web, es una oportunidad práctica para investigar y utilizar splinter.

Estos son algunos links que pueden complementar lo que estuvimos viendo en clase:

- **Sitio de splinter:** <http://splinter.cobrateam.info/>
- **Intérprete lanas:** <https://github.com/hugoruscitti/lanas>