BIG DATA

NIVEL 1 WEB SCRAPING
UNA SOLA PÁGINA ESTÁTICA
CON REQUESTS Y LXML

EDUARD LARA

1. INDICE

1. HERRAMIENTAS NIVEL 1

Para hacer web scraping hay claramente dos procesos:

1) Requerimiento al servidor para obtener la estructura HTML de la página web a extraer datos
2) Parseo de la respuesta para obtener los datos

Para este tipo de extracción de una sola página estática, vamos a manejar 4 librerías diferentes

1) **Para realizar los requerimientos de las páginas HTML → Requests**. Es la más sencilla de utilizar y aprender además es muy sencillo manejar autenticación y encabezados.

   instalación → pip install requests

   uso en Python → import requests

1. HERRAMIENTAS NIVEL 1

2) **Para parsear los requerimientos y extraer la información del árbol HTML → LXML y beautifulSoup** dos librerías excelentes para recorrer y parsear el árbol HTML de diferentes maneras entre ellas utilizando expresiones XPath

3) **También utilizaremos Scrapy. Sirve para poder hacer ambos procesos: requerimiento de información y Paseo de información**. Es una de las librerías más poderosas para hacer web scraping extático no sólo de una sino también de varias páginas que es el siguiente nivel.

1. HERRAMIENTAS NIVEL 1

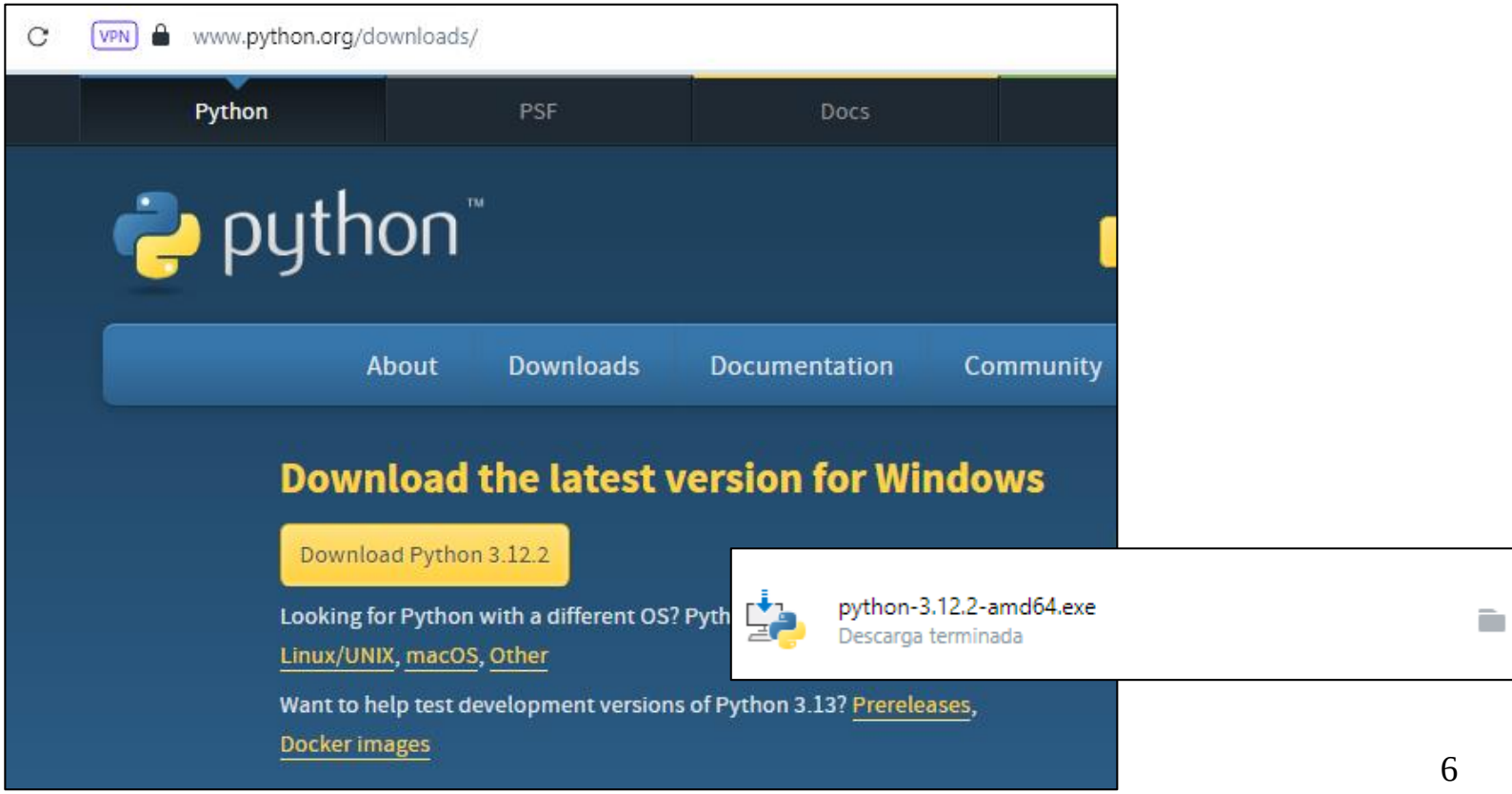
Proceso a seguir

a) Empezaremos utilizando Requests con el XML para extraer datos de wikipedia
b) Despues utilizaremos Requests con BeutifulSoup para extraer datos de stackoverflow
c) Finalmente utilizaremos Scrapy para extraer datos de stackoverflow y el Diario EL UNIVERSO

Contrastaremos los usos de cada una de estas librerías y veremos por qué en unos casos sirve de mejor manera y en otros otras me sirven de mejor manera.

2. INSTALACION PYTHON WINDOWS

Paso 1. Vamos https://www.python.org/downloads/. Dando click en el botón amarillo empezará la descarga de la versión idónea para su máquina.

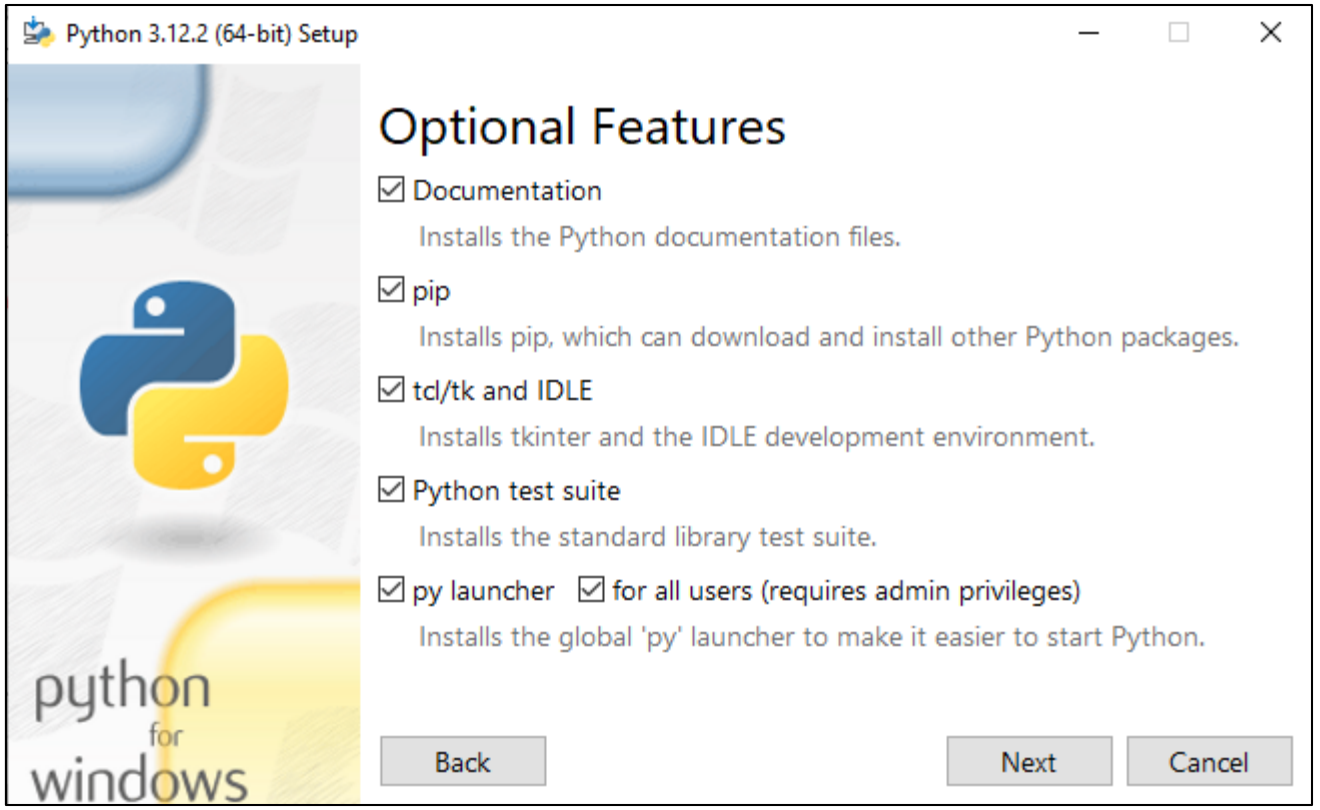


2. INSTALACION PYTHON WINDOWS

Paso 2. Marcamos ambas casillas disponibles y damos click en **Customize Installation** (instalación personalizada)

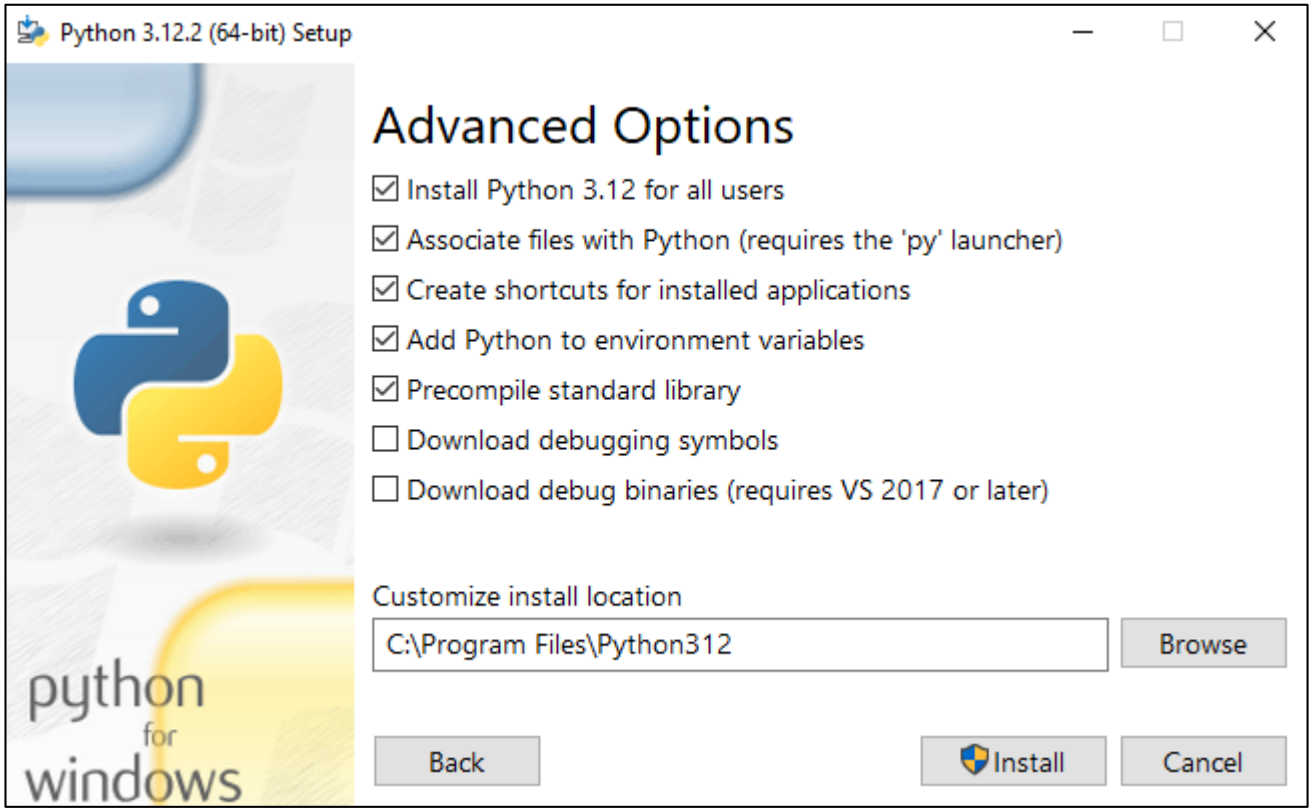
2. INSTALACION PYTHON WINDOWS

Paso 3. En la siguiente pantalla asegurémonos de marcar TODAS las opciones



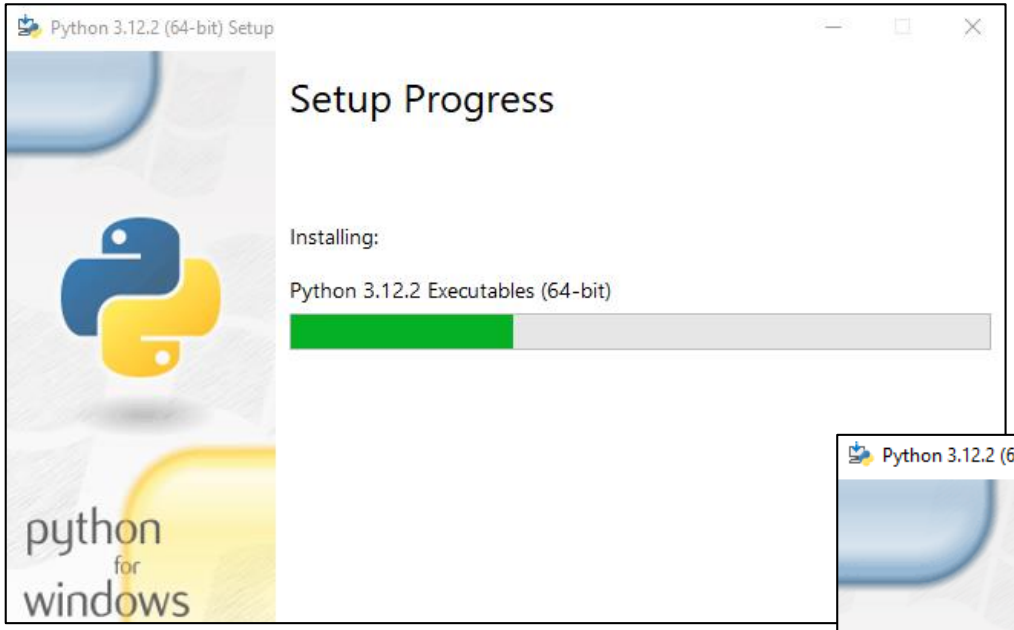
2. INSTALACION PYTHON WINDOWS

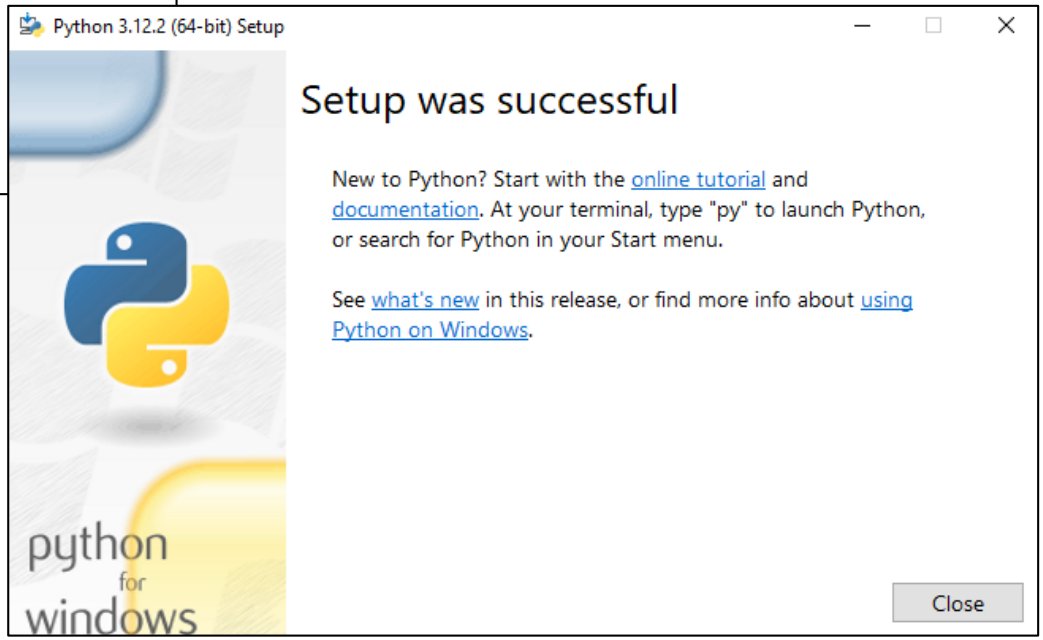
Paso 4. En la siguiente pantalla nos aseguramos de marcar las siguientes opciones. Se recomienda instalar para todos los usuarios (primer Checkbox).



2. INSTALACION PYTHON WINDOWS

Paso 5. Esperamos que la instalación termine





2. INSTALACION PYTHON WINDOWS

Paso 6. Abrimos un CMD en Windows buscaremos en la barra de herramientas de Windows "CMD" y daremos click para abrir esta aplicación.
Asegurémonos que la instalación halla sido exitosa intentando ejecutar alguno de los siguientes comandos en la terminal o CMD:

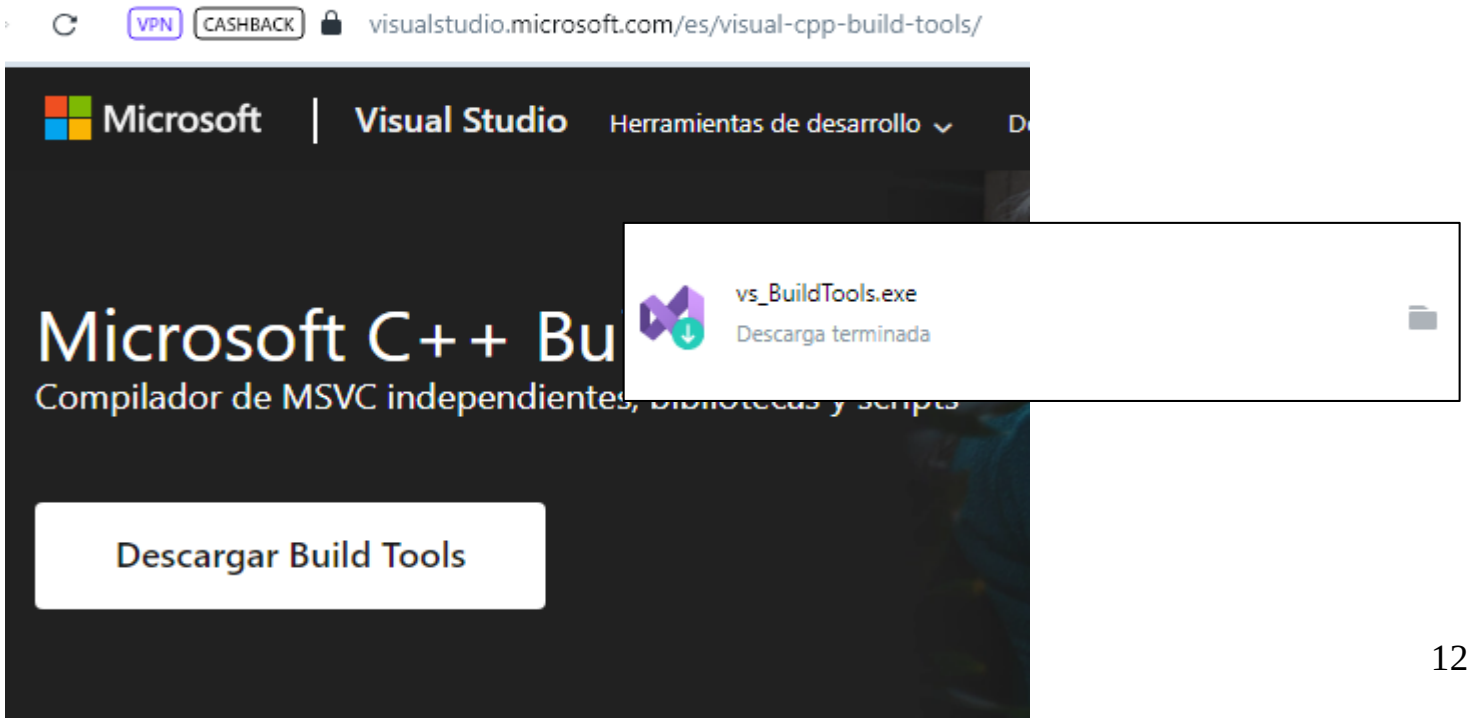
```
C:\Users\eduar>pip -V
pip 24.0 from C:\Program Files\Python312\Lib\site-packages\pip (python 3.12)

C:\Users\eduar>pip3 -V
pip 24.0 from C:\Program Files\Python312\Lib\site-packages\pip (python 3.12)
```

3. INSTALACION LIBRERIAS WEB SCRAPING

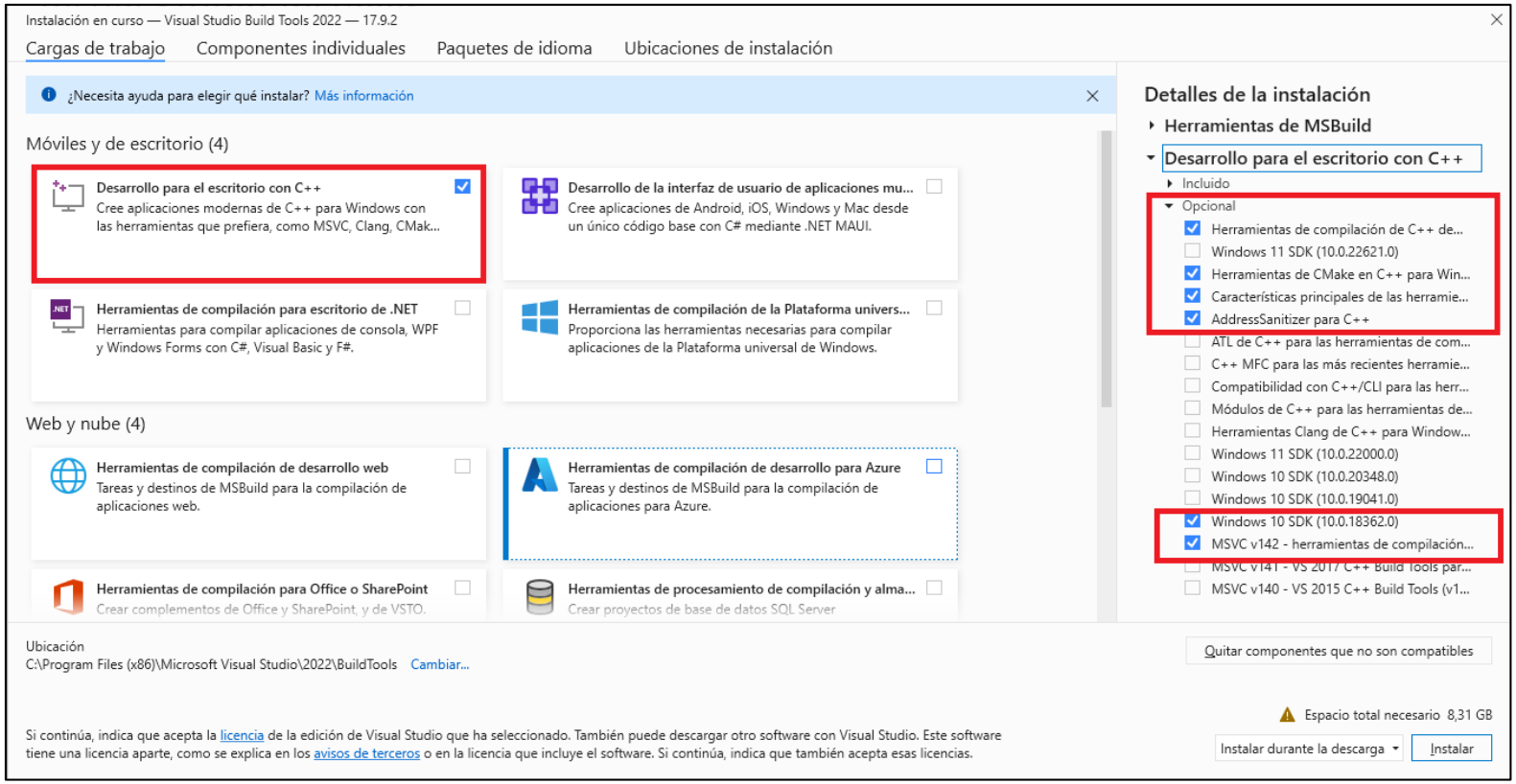
Paso 1. En Windows necesitamos instalar algunos prerequisitos para que la librería llamada Scrapy funcione correctamente. Instalaremos Microsoft C++ Build Tools entrando en el siguiente link:

https://visualstudio.microsoft.com/visual-cpp-build-tools/



3. INSTALACION LIBRERIAS WEB SCRAPING

Paso 2. En la primera ventana, nos aseguramos de tener las siguientes opciones marcadas: Desktop development with C++, MSVC v142. Dependiendo del S.O. marcar Windows 10 SDK o Windows 11 SDK. Hacer click en Install:



3. INSTALACION LIBRERIAS WEB SCRAPING

Paso 3. En Linux necesitamos idealmente tener instaladas algunas dependencias (especialmente para que la librería Scrapy funcione):

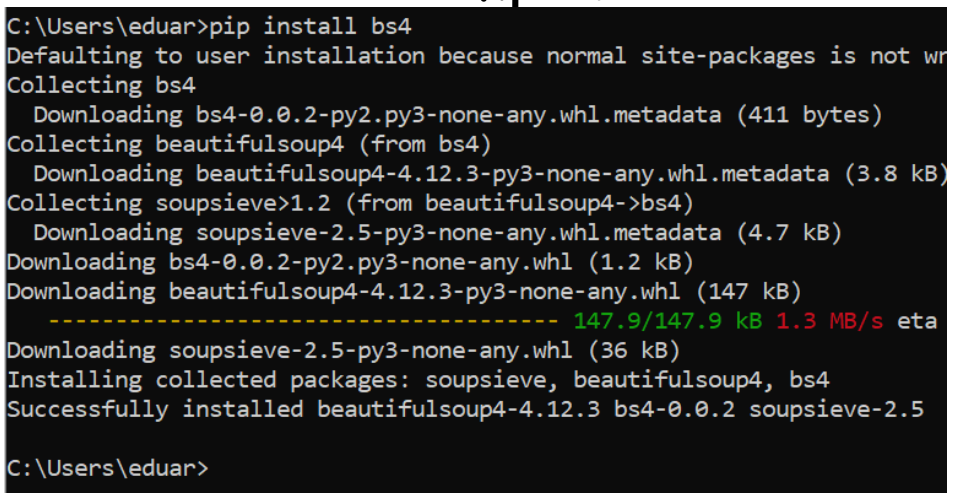
sudo apt-get install python3-dev libxml2-dev libxslt1-dev zlib1g-dev libffi-dev libssl-de

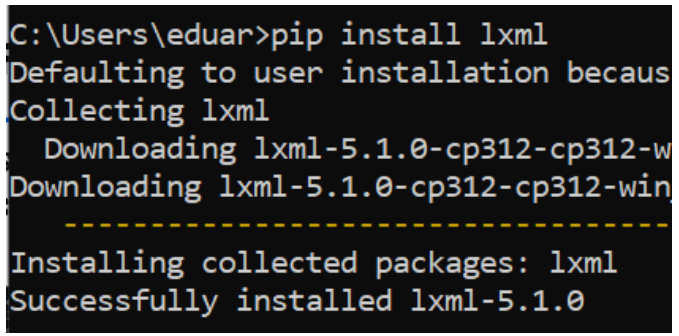
3. INSTALACION LIBRERIAS WEB SCRAPING

Paso 4. Finalmente, en una terminal CMD ejecutamos los siguientes comandos para instalar todas las librerías:

Nota: Si se utiliza pip3, solo se debe de remplazar

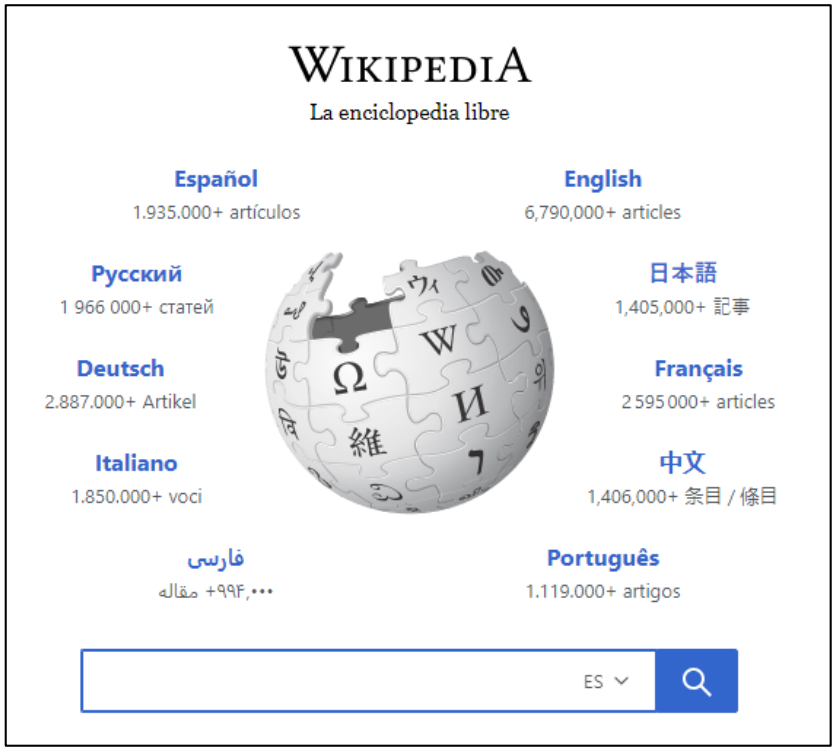
- pip install requests
- pip install lxml
- pip install bs4
- pip install scrapy
- pip install selenium
- pip install webdriver-manager





4. EXTRACCION DATOS DE WIKIPEDIA

Paso 1. Como primera extracción vamos a acceder a datos de la página principal de Wikipedia. Vamos a extraer alguno de los lenguajes que Wikipedia pone aquí, entre los cuales están escritos algunos de los artículos: inglés, alemán, etc



4. EXTRACCION DATOS DE WIKIPEDIA

Paso 2. Tenemos dos fases

a) Primera fase → **Requests** para hacer el requerimiento al servidor y obtener el HTML

b) Segunda fase → **LXML** para parsear la respuesta al requerimiento para extraer los datos deseados

4. EXTRACCION DATOS DE WIKIPEDIA

Paso 3. Creamos un proyecto en Visual Studio Code, y escribimos el siguiente código en wikipedia.py

```
import requests
```
→ Para importar requests en Python para hacer requerimientos web

```
url = 'https://www.wikipedia.org/'
```
→ URL de Wikipedia, por la que vamos a empezar. No vamos a movernos a otras URLs.

Paso 4. Para hacer el requerimiento a esta URL dentro de Python, usamos **requests.get(url)**, que recibe como parámetro la URL a la cual le voy a hacer el requerimiento

```
respuesta = requests.get(url)
```
→ En respuesta ya tenemos el árbol HTML que me devuelve esta URL.

4. EXTRACCION DATOS DE WIKIPEDIA

Paso 5. Antes de hacer el requerimiento, debemos modificar una variable en el encabezado HTTP: **user-agent**
Si no redefinimos esta variable, el valor por defecto que va a usar request es **Robot**, por lo que los servidores pueden identificar/reconocer que estamos intentando hacer web scrapping y nos baneen el acceso.

```
# USER AGENT PARA PROTEGERNOS DE BANEOS
encabezados = {
    "user-agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 "+
    "(KHTML, like Gecko) Ubuntu Chromium/71.0.3578.80 Chrome/71.0.3578.80 Safari/537.36",
}
```

Esta variable le indica al servidor desde qué navegador estamos haciendo el requerimiento y cuál es mi S.O.

Es importante que UserAgent siempre la reescribamos en cada proceso web scraping

4. EXTRACCION DATOS DE WIKIPEDIA

Paso 6. Ahora al hacer el requerimiento a la URL, le tenemos que pasar el nuevo encabezado a través del parámetro headers. Le indicamos al servidor que el requerimiento se va a realizar como si estuviéramos desde un navegador.

```
respuesta = requests.get(url, headers=encabezados)
print(respuesta.text)
```

En respuesta tenemos todo el árbol HTML.

Lo imprimimos con la función text, que simplemente convierte respuesta a un texto plano.

4. EXTRACCION DATOS DE WIKIPEDIA

Paso 7. Lo ejecutamos y vemos todo el código Wikipedia.org en la consola de Visual Studio

```
        }
    };
    window.setTimeout( wmL10nVisible.makeVisible, 1000 )
</script>
<script src="portal/wikipedia.org/assets/js/index-24c3e2ca18.js"></script>
<script src="portal/wikipedia.org/assets/js/gt-ie9-ce3fe8e88d.js"></script>
<style>
.styled-select {
        display: block;
    }
</style>
</body>
</html>
                                                >
PS C:\Users\eduar\OneDrive\Desktop\scraping>
```

4. EXTRACCION DATOS DE WIKIPEDIA

Paso 8. Ahora usaremos la segunda librería LXML para parsear todo el árbol HTML y extraer los datos que queremos.

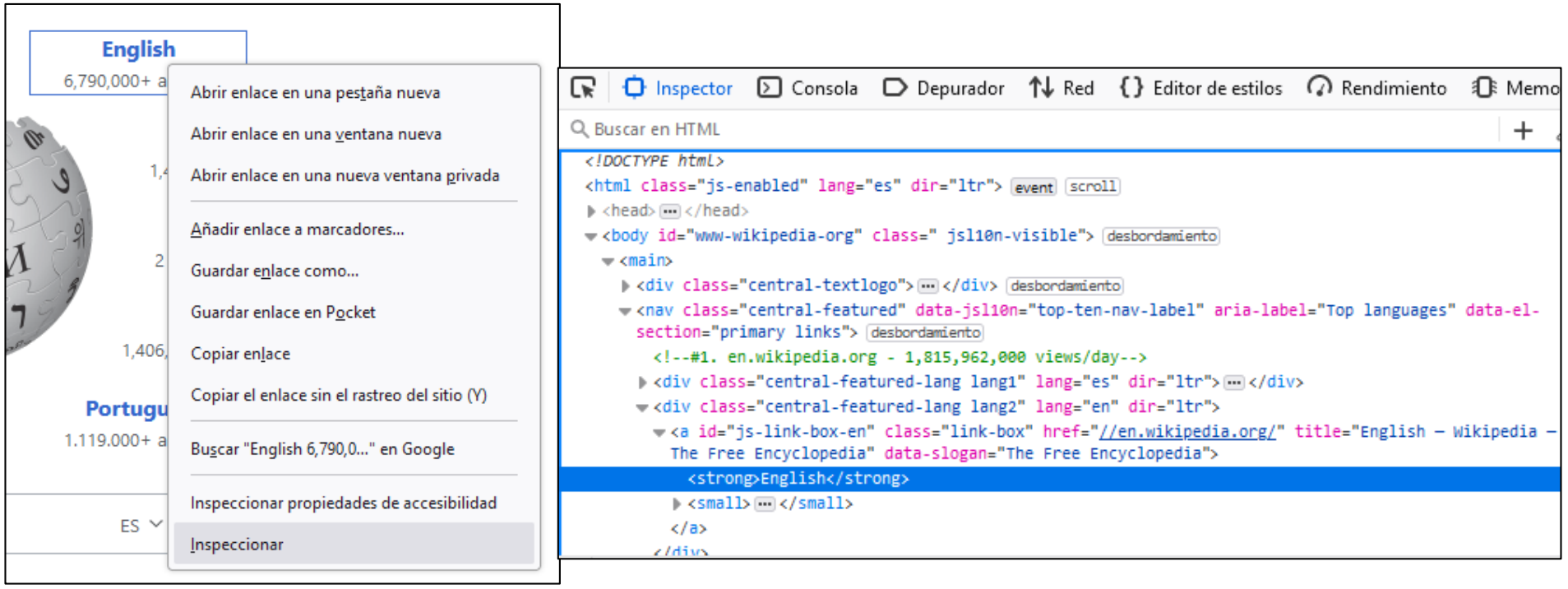
```
from lxml import html
```
→ Importamos la librería LXML

```
parser = html.fromstring(respuesta.text)
parser = html.fromstring(respuesta.content)
```
→ Creamos una variable parser, la cual tiene unos métodos muy interesantes para poder empezar a buscar cosas dentro del árbol HTML

4. EXTRACCION DATOS DE WIKIPEDIA

Paso 9. Para empezar a extraer datos tenemos que revisar la estructura del HTML de la página. Intentaremos extraer primero el texto "English" de wikipedia.org

Sobre el elemento que queremos extraer información hacemos click derecho y seleccionamos Inspeccionar



4. EXTRACCION DATOS DE WIKIPEDIA

Paso 10. Vemos que nos sitúa encima de la palabra English, que es lo que queremos extraer. Vemos que está dentro de un tag a, el cual tiene como atributo **id=js-link-box-en**. Este atributo es único en todo el árbol y es una manera en la cual podemos llegar hasta la información.

El parser provee técnicas para acceder a los elementos:

- Xpath
- get_element_by_id

```
# EXTRACCION DE IDIOMA INGLES
ingles = parser.get_element_by_id("js-link-box-en")
print (ingles)
```

Guardamos en la variable ingles, el elemento dentro de la página de Wikipedia que tenga este ID y lo imprimimos

4. EXTRACCION DATOS DE WIKIPEDIA

Paso 11. Ejecutamos el programa en Python.

```
PS C:\Users\eduar\OneDrive\Desktop\scraping>
esktop/scraping/wikipedia.py
<Element a at 0x1c3d8bcb020>
PS C:\Users\eduar\OneDrive\Desktop\scraping>
```

El parseador nos muestra la etiqueta contenedora donde esta el id.

Paso 12. Si queremos obtener el contenido del texto que está dentro del elemento HTML que tiene este ID:

```
# EXTRACCION DE IDIOMA INGLES
ingles = parser.get_element_by_id("js-link-box-en")
print (ingles.text_content())
```

```
English
6,790,000+ articles
```

4. EXTRACCION DATOS DE WIKIPEDIA

Paso 13. Podemos obtener este mismo elemento con expresiones XPath de la siguiente manera:

```
<a id="js-link-box-en" class="link-box" href="//en.wikipedia.org/" title="English –
The Free Encyclopedia" data-slogan="The Free Encyclopedia">
    <strong>English</strong>
  ▶ <small> ··· </small>
```

```python
# EXTRACCION MEDIANTE XPATH
ingles = parser.xpath("//a[@id='js-link-box-en']/strong/text()")
print(ingles)
```

En todo el árbol (//), buscamos el tag a cuyo atributo id sea igual a js-link-box-en. Desde ahí vamos al primer hijo directo que tenga el tag strong. Y finalmente obtenemos el texto. El resultado de esto lo guardamos en una variable y la imprimimos.

```
PS C:\Users\eduar\OneDrive\Desktop\scraping>
esktop/scraping/wikipedia.py
['English']
PS C:\Users\eduar\OneDrive\Desktop\scraping>
```

4. EXTRACCION DATOS DE WIKIPEDIA

Paso 14. Ahora vamos a por nuestro objetivo, que es el de sacar todos los idiomas. Lo haremos mediante una expresión XPath. Analizamos el patrón de los idiomas

El patrón se encuentra dentro de los div class central-features-lang. Dentro todos guardan la misma estructura.

4. EXTRACCION DATOS DE WIKIPEDIA

Paso 15. Una posible expresión XPath para buscar los idiomas

```
# EXTRACCION DE TODOS LOS IDIOMAS POR XPATH
idiomas = parser.xpath("//div[contains(@class,'central-featured-lang')]//strong/text()")
print (idiomas)
```

En todo el árbol, búscame todos los divs cuya clase contenga central-featured-lang. Dentro de sus hijos buscamos strong, que es donde se encuentra el texto del idioma y obtengo el texto.

Esta expresión se va a aplicar para todos los elementos que hagan match y nos va a devolver la lista de idiomas

```
PS C:\Users\eduar\OneDrive\Desktop\scraping> & "C:/Program Files/Python312/python.exe" c:/Users/eduar/OneDrive/
esktop/scraping/wikipedia.py
['English']
['English', 'Đ\xa0Ñ\x83Ñ\x81Ñ\x81ĐºĐ¸Đ¹', 'EspaÃ±ol', 'æ\x97¥æ\x9c¬èª\x9e', 'Deutsch', 'FranÃ§ais', 'Italiano',
'ä¸\xadæ\x96\x87', 'PortuguÃªs']
PS C:\Users\eduar\OneDrive\Desktop\scraping>
```

4. EXTRACCION DATOS DE WIKIPEDIA

Paso 16. Como esta es una lista de elementos, la podemos iterar elemento por elemento.

```
# EXTRACCION DE TODOS LOS IDIOMAS POR XPATH
idiomas = parser.xpath("//div[contains(@class,'central-featured-lang')]//strong/text()")
for idioma in idiomas:
    print(idioma)
```

```
esktop/scraping/wikipedia.py
['English']
English
Ð ÑÑÑÐºÐ¸Ð¹
EspaÃ±ol
æ¥æ¬èª§ais
Italiano
ä¸æ
PortuguÃªs
```

Se nos imprime una en cada línea.

4. EXTRACCION DATOS DE WIKIPEDIA

Paso 17. Lo hemos obtenido armando una expresión XPath Podemos hacer lo mismo utilizando otra función que provee LXML, find_class que nos va a encontrar todos los elementos que hagan match con una clase

```
# EXTRACCION DE TODOS LOS IDIOMAS POR CLASE
idiomas = parser.find_class('central-featured-lang')
for idioma in idiomas:
  print(idioma.text_content())
```

Para obtener el texto debemos agregar text_content() ya que en este caso no se trata de XPath

4. EXTRACCION DATOS DE WIKIPEDIA

Paso 18. Si lo ejecutamos, obtenemos los idiomas y el numero de artículos (contenido del texto de cada uno de los divs que contienen esta clase central-featured-lang)

```
FranÃ§ais
2â¯595â¯000+ articles



Italiano
1.850.000+ voci



ä¸ æ
1,406,000+ æØ§Ø±Ø³Û
Û¹Û¹Û´Ù¬Û°Û°Û°+ Ù
ÙØ§Ù
    Ù
```

La función text_content() devuelve el contenido de texto del elemento, incluido el contenido de texto de sus hijos, sin marcado.