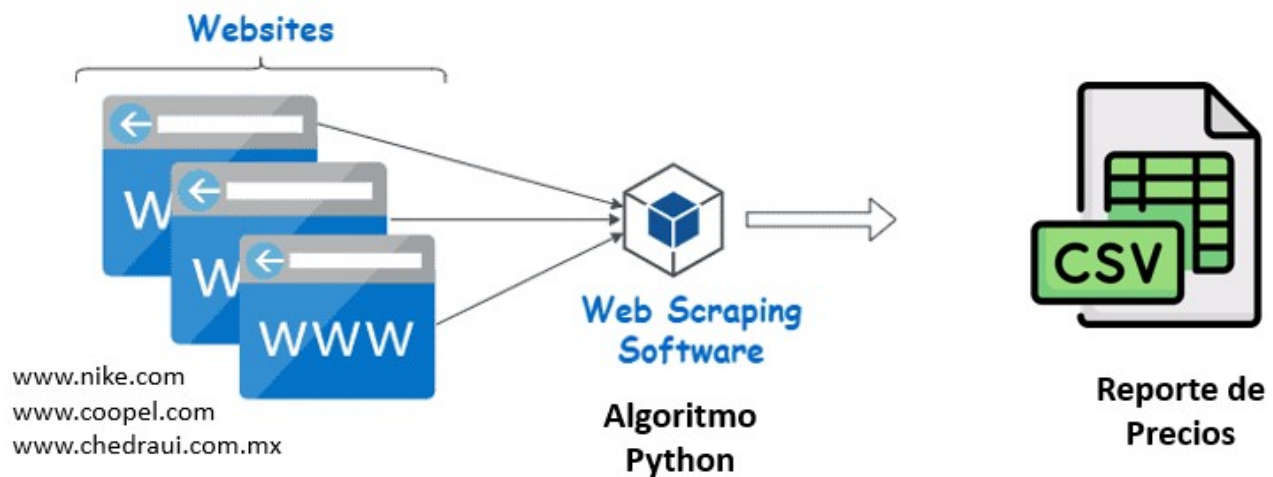


S3.A1 Desarrollar algoritmo para la identificación de precios usando la técnica de Web Scraping

Objetivo:

El objetivo de esta actividad consiste en desarrollar un algoritmo en el lenguaje Python que permita el mapeo y/o identificación de precios de un producto en 3 sitios web diferentes utilizando la técnica de Web Scraping; tal como se visualiza en el siguiente diagrama:



Requisitos de la practica:

- Se debe imprimir los 3 precios e indicar el menor de ellos.
- Impresión de pantalla del reporte de precios en formato CSV.

Empezemos mencionando que es el web Scraping:

Es un proceso de extracción de datos de sitios web. Este utiliza herramientas y técnicas para recolectar información de páginas web de manera automatizada y estructurada.

Además de que es una técnica muy poderosa para obtener datos que no están disponibles de manera fácil en formatos accesibles como APIs o base de datos.

Primero identificaremos los sitios web a los cuales les haremos la extracción de datos.

En este caso utilizaremos la biblioteca Web Scraping de Python llamada BeautifulSoup y request ya que esta nos permite manejar los datos de manera más sencilla y poder realizar peticiones y manejar estas peticiones de manera sencilla para ello necesitamos realizar los siguientes pasos:

la instalar BeautifulSoup y Request :

Pesenta: B18022, Jose Colombio Gonzalez Perez

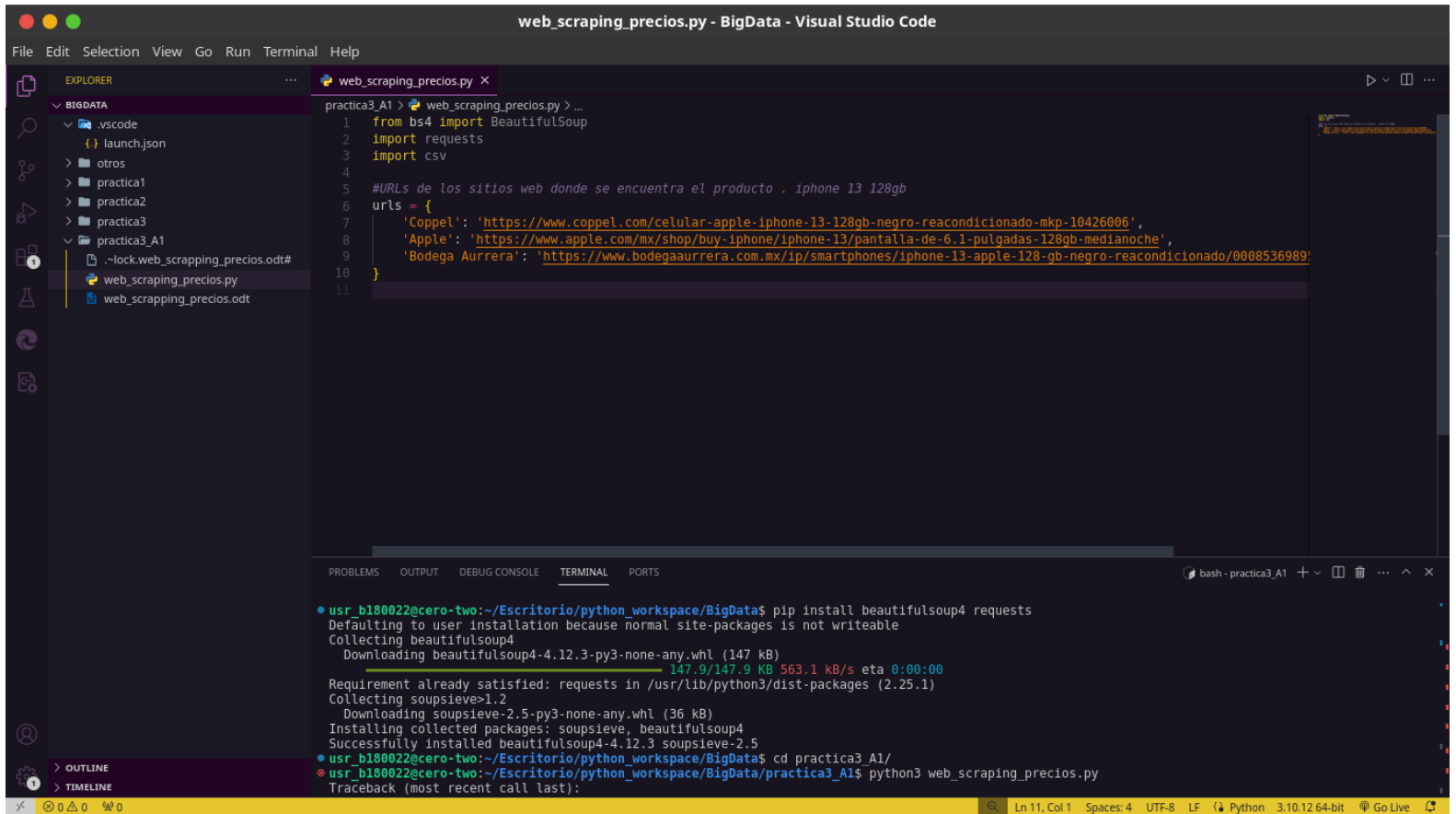
pip install beautifulsoup4 requests

importamos las librerias en nuestro programa.

Y procedemos al desarrollo de nuestro codigo:

paso 1:

Definimos un array que contendra nuestras URLs.



The screenshot shows the Visual Studio Code interface with a file named `web_scraping_precios.py` open in the editor. The script imports `BeautifulSoup` from `bs4`, `requests`, and `csv`. It defines a list of URLs for scraping iPhone prices from three different websites. The Explorer sidebar on the left shows the project structure with folders `practica1`, `practica2`, `practica3`, and `practica3_A1`. The Terminal at the bottom shows the execution of `pip install beautifulsoup4 requests` and `python3 web_scraping_precios.py`, with output indicating successful installation and execution.

```
web_scraping_precios.py - BigData - Visual Studio Code
File Edit Selection View Go Run Terminal Help

EXPLORER
BIGDATA
  .vscode
  launch.json
  otros
  practica1
  practica2
  practica3
  practica3_A1
    .lock.web_scraping_precios.odt#
    web_scraping_precios.py
    web_scraping_precios.odt

web_scraping_precios.py
1 from bs4 import BeautifulSoup
2 import requests
3 import csv
4
5 #URLs de los sitios web donde se encuentra el producto . iphone 13 128gb
6 urls = {
7     'Coppel': 'https://www.coppel.com/celular-apple-iphone-13-128gb-negro-reacondicionado-mkp-10426006',
8     'Apple': 'https://www.apple.com/mx/shop/buy-iphone/iphone-13/pantalla-de-6.1-pulgadas-128gb-medianoche',
9     'Bodega Aurrera': 'https://www.bodegaaurrera.com.mx/ip/smartphones/iphone-13-apple-128-gb-negro-reacondicionado/0008536989'
10 }
11

TERMINAL
bash - practica3_A1
• usr_b180022@cero-two:~/Escritorio/python_workspace/BigData$ pip install beautifulsoup4 requests
Defaulting to user installation because normal site-packages is not writeable
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.12.3-py3-none-any.whl (147 kB)
    147.9/147.9 KB 563.1 kB/s eta 0:00:00
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (2.25.1)
Collecting soupsieve>1.2
  Downloading soupsieve-2.5-py3-none-any.whl (36 kB)
Installing collected packages: soupsieve, beautifulsoup4
Successfully installed beautifulsoup4-4.12.3 soupsieve-2.5
• usr_b180022@cero-two:~/Escritorio/python_workspace/BigData$ cd practica3_A1/
• usr_b180022@cero-two:~/Escritorio/python_workspace/BigData/practica3_A1$ python3 web_scraping_precios.py
Traceback (most recent call last):
```

Paso 2:

Ahora procedemos a realizar un metodo para poder realizar una peticion a cada URL y obtener el contenido HTML de la pagina y retornamos el contenido HTML, nos quedaria de la siguiente manera.

```
web_scraping_precios.py X
practica3_A1 > web_scraping_precios.py > ...
1 from bs4 import BeautifulSoup
2 import requests
3 import csv
4
5 #URLs de los sitios web donde se encuentra el producto . iphone 13 128gb
6 urls = {
7     'Coppel': 'https://www.coppel.com/celular-apple-iphone-13-128gb-negro-reacondicionado-mkp-10426006',
8     'Apple': 'https://www.apple.com/mx/shop/buy-iphone/iphone-13/pantalla-de-6.1-pulgadas-128gb-medianoche',
9     'Bodega Aurrera': 'https://www.bodegaurrera.com.mx/ip/smartphones/iphone-13-apple-128-gb-negro-reacondicionado/0008536989'
10 }
11
12 #Metodo para obtener el HTML de la página correspondiente
13 def obtener_html(url):
14     response = requests.get(url)
15     if response.status_code == 200:
16         return response.content
17     else:
18         print('Error al obtener la página:', url)
19         return None
20
```

Paso 3:

Crear métodos para cada URL y extraer el datos del producto, pero solo retornamos su precio pro que es lo que nos interesa.

```
web_scraping_precios.py •
practica3_A1 > web_scraping_precios.py > extraer_datos_coppel
21 #Metodos para extraer datos especificos de cada sitio web
22 def extraer_datos_coppel():
23     url = 'https://www.coppel.com/celular-apple-iphone-13-128gb-negro-reacondicionado-mkp-10426006'
24     response = requests.get(url)
25     if response.status_code == 200:
26         soup = BeautifulSoup(response.content, 'html.parser')
27         nombre = soup.find('h1', class_='productTitle').text.strip()
28         descripcion = soup.find('div', class_='contentBox').text.strip()
29         precio_tag = soup.find('span', class_='price')#tag precio
30
31         if precio_tag:
32             precio = precio_tag.get_text()
33             print("----- Tienda Coppel: -----")
34             print("nombre: ", nombre)
35             print("precio: ", precio)
36             print("Descripcion: ", descripcion)
37             return precio
38         else:
39             return 'Precio no encontrado'
40     else:
41         return 'Error al obtener la página'
42
43 def extraer_datos_apple():
44     url = 'https://www.apple.com/mx/shop/buy-iphone/iphone-13/pantalla-de-6.1-pulgadas-128gb-medianoche'
45     response = requests.get(url)
46     if response.status_code == 200:
47         soup = BeautifulSoup(response.content, 'html.parser')
48         nombre = soup.find('h1', class_='productTitle').text.strip()
49         descripcion = soup.find('div', class_='contentBox').text.strip()
50         precio_tag = soup.find('div', class_='price-display').find('span', class_='current_price')
51         if precio_tag:
52             precio = precio_tag.get_text()
53             print("----- Tienda Apple: -----")
54             print("nombre: ", nombre)
55             print("precio: ", precio)
56             print("Descripcion: ", descripcion)
57             return precio
58         else:
59             return 'Precio no encontrado'
60     else:
61         return 'Error al obtener la página'
```

```
web_scraping_precios.py •
practica3_A1 > web_scraping_precios.py > extraer_datos_apple
43 def extraer_datos_apple():
44     response = requests.get(url)
45     if response.status_code == 200:
46         soup = BeautifulSoup(response.content, 'html.parser')
47         nombre = soup.find('h1', class_='productTitle').text.strip()
48         descripcion = soup.find('div', class_='contentBox').text.strip()
49         precio_tag = soup.find('div', class_='price-display').find('span', class_='current_price')
50         if precio_tag:
51             precio = precio_tag.get_text()
52             print("----- Tienda Apple: -----")
53             print("nombre: ", nombre)
54             print("precio: ", precio)
55             print("Descripcion: ", descripcion)
56         return precio
57     else:
58         return 'Precio no encontrado'
59     else:
60         return 'Error al obtener la página'
61
62 def extraer_datos_bodega_aurrera():
63     url = 'https://www.bodegaaurrera.com.mx/ip/smartphones/iphone-13-apple-128-gb-negro-reacondicionado/00085369895437?from=/s'
64     response = requests.get(url)
65     if response.status_code == 200:
66         soup = BeautifulSoup(response.content, 'html.parser')
67         nombre = soup.find('h1', class_='productTitle').text.strip()
68         descripcion = soup.find('div', class_='contentBox').text.strip()
69         precio_tag = soup.find('span', class_='price-text')
70         if precio_tag:
71             precio = precio_tag.get_text()
72             print("----- Tienda BodegaAurrera: -----")
73             print("nombre: ", nombre)
74             print("precio: ", precio)
75             print("Descripcion: ", descripcion)
76         return precio
77     else:
78         return 'Precio no encontrado'
79     else:
80         return 'Error al obtener la página'
81
82
83
84
```

Paso 4:
Finalmente lo guardamos en un archivo csv:

```
web_scraping_precios.py X
practica3_A1 > web_scraping_precios.py > ...
90 def extraer_datos_bodega_aurrera(html):
110     descripcion = 'Descripción no encontrada'
111
112
113     print("-- Datos Bodega Aurrera")
114     print("nombre", nombre)
115     print("precio", precio)
116     print("Descripción: ", descripcion)
117
118     return nombre, precio, descripcion
119
120 # Ejecutar el proceso para cada sitio web
121 datos_coppel = extraer_datos_coppel(obtener_html(urls['Coppel']))
122 datos_apple = extraer_datos_apple(obtener_html(urls['Apple']))
123 datos_bodega_aurrera = extraer_datos_bodega_aurrera(obtener_html(urls['Bodega Aurrera']))
124
125 # Verificar si se obtuvieron los datos (precio) correctamente y guardar en CSV
126 if datos_coppel[1] != 'Precio no encontrado' and datos_apple[1] != 'Precio no encontrado' and datos_bodega_aurrera[1] != '':
127     datos_precios = [datos_coppel, datos_apple, datos_bodega_aurrera]
128     guardar_datos_csv(datos_precios, 'datos_precios.csv')
129     print('Datos guardados en datos_precios.csv')
130 else:
131     print('No se pudieron obtener todos los precios.')
132
133 # Imprimir precios obtenidos
134 print('\nPrecios obtenidos:')
135 print(f'Coppel: {datos_coppel[1]}')
136 print(f'Apple: {datos_apple[1]}')
137 print(f'Bodega Aurrera: {datos_bodega_aurrera[1]}')
138
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

File "/home/usr_b180022/Escritorio/python_workspace/BigData/practica3_A1/web_scraping_precios.py", line 91, in extraer_datos_bodega_aurrera
soup = BeautifulSoup(html, 'html.parser')
File "/home/usr_b180022/.local/lib/python3.10/site-packages/bs4/_init_.py", line 315, in __init__
elif len(markup) <= 256 and (
TypeError: object of type 'NoneType' has no len()

No se pudo Completar la practica opr que hibo bloqueo por el servidor de la paginas considero que tienen algun mecanismo o proteccion para el ataque de web Scrapping.

El proyecto lo dejo en el repositorio , y este sera actualizado en el futuro.

<https://github.com/pepe1603/repo-BigData.git>