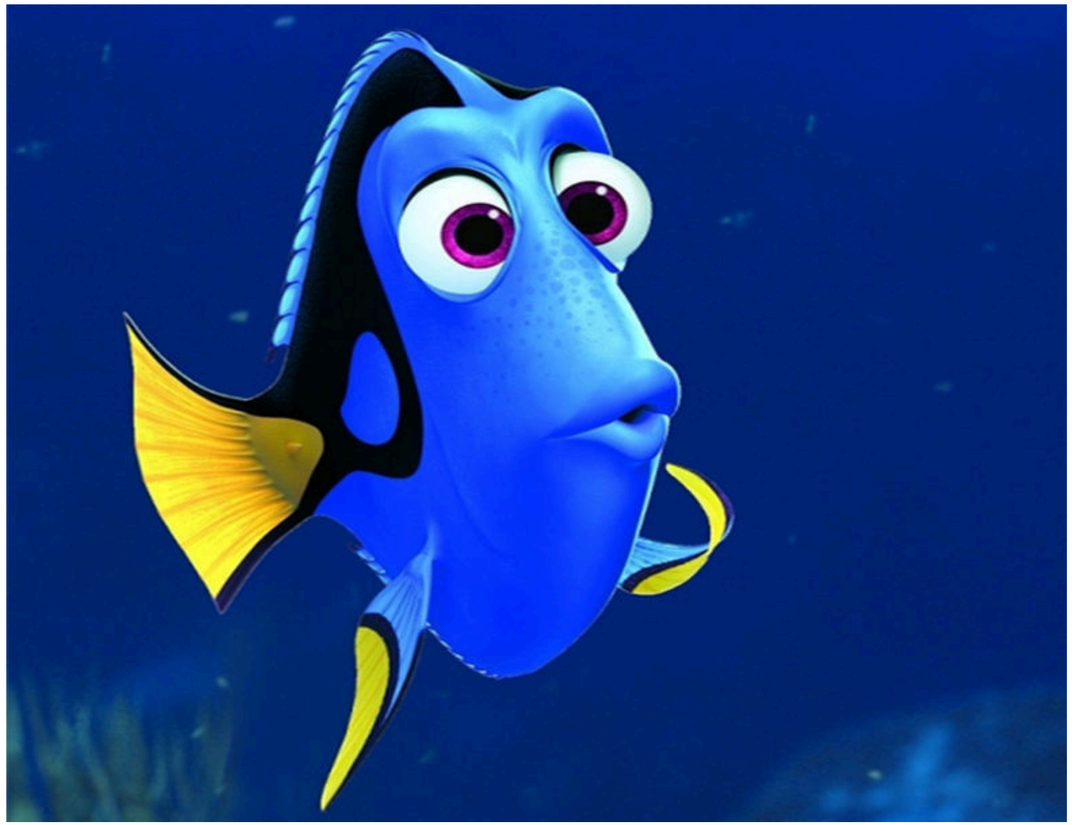
Bienvenidos



y en la clase anterior....

beautifull soup

Beautifull soup

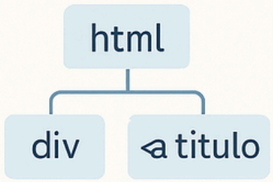
(analizar y recorrer) documentos HTML y XML de forma sencilla

Beautiful Soup 4 (BS4) * última versión

Leonard Richardson

Beautiful Soup te permite:

Parsear documentos HTML/XML:



Convertir /a contenido en un arrôo de objetos fácil de navegar

Acceder a elementos por nombre de etiqueta, atributos, clases, etc.

```
soup.find('div')
```

- soup, find_all (class_="titulo')
- IDs, textos, y más

Modificar o navegar en la estructura:

```
<html
<head>
<body
</html>
```

Extratar, reeplacer, mover o elimitar nodos del DOM

Lidiar con HTML mal formado:

```
<p><b>.../p>
```

Beautiful Soup está diseñado para arreglar errores comunes en HTML incorrecto

1

Usa parsers (analizadores) para interpretar HTML. Por defecto usa el parser de Python, pero puede trabajar con:

- html.parser (integrado en Python)
- lxml (más rápido, necesita instalación)
- html5lib (más tolerante con HTML incorrecto)

2

Crear objeto beautifull soup

```
soup = BeautifulSoup(response.content, 'html.parser')
```

Búsqueda básica con etiquetas

3

```
soup.title.text
soup.find_all('div', class_='quote')
```

Navegar por los elementos

4

```
quote.find('span', class_='text').get_text(strip=True)
```

Explicación: Se extraen enlaces y etiquetas asociadas a cada cita.

5

```
author_about['href']
quote.find_all('a', class_='tag')
```

manipulación de atributos

6

```
author_about['href']
quote.find_all('a', class_='tag')
```

código

```
if response.status_code == 200:
    print("Conexión exitosa. Código de estado:", response.status_code)
```

Se comprueba si la conexión al sitio web respondió con el código HTTP 200 (éxito). Si no, se imprime un error.

código

```
soup = BeautifulSoup(response.content, 'html.parser')
```

Se utiliza BeautifulSoup para convertir el contenido HTML en un objeto navegable y manipulable.

código

```
page_title = soup.title.text
```

Se utiliza BeautifulSoup para convertir el contenido HTML en un objeto navegable y manipulable.

código

Obtener el título de la página:

```
all_quotes = soup.find_all('div', class_='quote')
```

Encontrar elementos div con clase quote:

```
all_quotes = soup.find_all('div', class_='quote')
```

Extraer texto de las primeras 3 citas

```
quote_text = quote.find('span', class_='text').get_text(strip=True)
```

Obtener el primer autor:

```
first_author = soup.find('small', class_='author')
```

código

Filtro por autor (Albert Einstein)

```
def author_filter(tag): ...
einstein_quotes = soup.find_all(author_filter)
```

código

Filtrar citas que contienen la palabra “love”

```
love_quotes = soup.find_all(text=lambda text: 'love' in text.lower() if text else False)
```

código

Manipulación de atributos

```
author_about['href']
quote.find_all('a', class_='tag')
)
```

código

Funcionalidad 3 - Extracción de atributos HTML

```
author_links.append({
    'cita': quote_text,
    'autor': author,
    'enlace': author_link,
    'etiquetas': tag_list
})
```

código

Obtener el padre del primer div.quote:

parent = first_quote.parent

Listar hijos directos de una cita:

children = list(first_quote.children)

Obtener hermanos y navegar hacia arriba/abajo en el árbol HTML:

next_sibling = author_elem.find_next_sibling()
quote_div = text_elem.find_parent('div', class_='quote')

código

Se extrae toda la información de las citas: texto, autor y etiquetas.
Se crea un DataFrame de pandas:

```
df = pd.DataFrame(all_quotes)
```