For Web Scraping, we will be using the library called BeautifulSoup

Web Scraping means to gather information from any publicly available website.

- Command to install beautifulsoup4:-

```
pip install beautifulsoup4
```
→ version

We can install it in our virtual environment.

- Next thing to be installed is the parser method from the beautifulsoup library:-

```
pip install lxml
```

When we are working with beautifulsoup, we need to specify the method that we are going to use to parse HTML pages to Python objects.

Among the available parsers, lxml is good and can work with broken HTML codes, unlike the default HTML parser.

- The beautifulsoup library is installed in a folder named bs4. Thus, in our Python file, we import it as:-

```
from bs4 import BeautifulSoup
```

- Now, we need to figure it out how we can access the contents of the webpage.

At first, we will take the webpage, and will access it using file operations.

For beginning purpose, we will take the webpage, and save it in the same directory as the Python file, with name, say home.html

To read any file in Python, following code can be used:

```
with open("home.html", 'r') as html_file:
    content = html_file.read()
    print(content)
```

file handling function to open a file as python object.

function is used to return entire content in the object.

Above code will print the entire code of the webpage.

We will use BeautifulSoup to prettify the HTML content above, and work with HTML tags as Python objects, as follows :-

```
from bs4 import BeautifulSoup

with open('home.html', 'r') as html_file:
    content = html_file.read()

    soup = BeautifulSoup(content, 'lxml')
    print(soup.prettify())
```

location of the file wrt the current working directory in the executing terminal.

file object containing HTML code.

the html parser that we want to use with Beautiful Soup.

This code will show the HTML code in exactly the format as it looks in IDE.

⇒ Now, we will see how to grab some specific info. Say, we want to extract info from all <h5> tags. for this, find() method of the Beautiful Soup object soup, created above.

find() method takes tag name as string, and returns all the specific tags that matches with it.

```
tags = soup.find("h5")
print(tags)
```

→ will return the 1st <h5> tag alongwith content from the webpage as tag object

It will print the 1st <h5> tag alongwith its content, that appears 1st from starting of webpage

⇒ To fetch all the tags with matching tag name, the find_all() method is used.

```
tags = soup.find_all("tagname")
```

It will return a list of html tag elements (alongwith content), with the matching tag name.

Ex:-

```
tags = soup.find_all("h5")
print(tags)
```

Will print a list containg all <h5> tags.

⇒ From the tag objects, if we want to extract (and use) only the content of the tags, then we can

do it as follows :-

```
from bs4 import BeautifulSoup

with open('home.html', 'r') as html_file:
    content = html_file.read()

    soup = BeautifulSoup(content, 'lxml')
    courses_html_tags = soup.find_all('h5')
    for course in courses_html_tags:
        print(course.text)
```

list containing all <h5> tag objects

text is a variable of the tag object, which contains the content of an element as string.

From each tag object obtained from find_all() method, the contents can be extracted/used, using the text variable.

16:25

⇒ The real life webpages are much more complex to understand at a glance.

To gather info of webpage's specific section, we need to inspect the webpage, to find what html tag & element contains specific info.

⇒ Filtering tags based on HTML class names:

⇒ Tags fetched as objects, can be filtered, based upon the HTML classes, using the class_ keyword argument of the soup.find_all() method :-

Syntax :-

takes target class name as string

```
elements = soup.find_all("tagname", class_ = "classname")
```

Ex :-

```
course_cards = soup.find_all("div", class_ = "card")
```

It will fetch all these elements :- ...

Accessing individual sub tag elements inside the fetched objects :-

In the fetched objects, such as above objects, the subtags inside these tags are can attribute / instance member variable of those objects.

For example, suppose the tags are as follows:-

```
<div class = "card ...>
    <h5> Python for beginners </h5>
    <p> <button> Start for $20 </button> </p>
</div>
```

If this tag is stored in "course" named variable, then the elements inside are accessed as:-

course.h5 , course.p

They are also objects now, and any element inside them can be further accessed by same way.

Course name can be accessed as :- course.h5.text

2. A sample scraping script to scrap prices of course and show them in formatted way is follows :-

```
from bs4 import BeautifulSoup

with open('./practice/courses.html','r') as courses_html_file:
    courses_page_content = courses_html_file.read()
    soup = BeautifulSoup(courses_page_content,'lxml')
    course_cards = soup.find_all('div', class_ = 'card')
    for course in course_cards:
        course_name = course.h5.text
        price_text = course.a.text
        course_price = price_text.split()[-1]

        print(f"{course_name}  costs  {course_price}\n")
```

The last word in the text contains price, thus seperating it