**WEB SCRAPPING (BEAUTIFULSOUP)**

Web scrapping can extract a large amount of data from websites. Here the data on the website are unstructured format. Web scrapping helps to collect unstructured data and store it in a structured format. Scrapped data can be stored in any local file in your computer or any Database.

Most of the time STARTUP companies prefer web scrapping because it's a cheap and very effective way to get a large amount of data at a time without any partnership with other companies.

web scraping is also known as web data extraction or web harvesting.

All websites can't allow web scrapping some websites allow it. If you want to know whether a website allows web scrapping or not. If you want to scrap the data. In URL after the domain name or server name, we can append the "/robots.txt," file.

Take an example to clearly understand. Here I am scrapping the Instagram website.

https://www.instagram.com/robots.txt

Here we can't scrap phone pay it shows a 404 error page not found error.

https://www.phonepay.com/robots.txt

**Beautiful Soup:**

Beautiful Soup is the python library that is used to pull the data that are HTML files and XML files. (It can also include malformed markup means missing the closed tags). It's useful for web scrapping. In web scraping, we can use this package called Beautiful Soup. By installing this package, we can use this command.

pip install beautifulsoup4

The latest version of the beautiful soup is 4.11.1

**Scrape data from the website:**

To extract data using web scraping python you should follow the below steps.

Find the URL that you want to scrape.

Inspecting the Page (enter F12 or open the web page you want to scrap and right-click on the page it shows inspect option).

Find the data you want to extract.

Write the code.

Run the code and extract the data.

Store the data in the required format.

**Sample program:**

import requests

from bs4 import BeautifulSoup

url = 'https://www.flipkart.com/beautiful-soup-webscraping-python/'

r = requests.get(url)

soup = BeautifulSoup(r.content, 'lxml')

print(soup.prettify())

beautiful soup has a prettify()function.

Using prettify function adds whitespace for indention, here you should not use any reformatting. Using prettify function visualization is very good.

**NUMPY**

NumPy is a python library.

NumPy stands for numerical python.

NumPy is used for working with arrays. NumPy provides functions related to the arrays.

NumPy creates an array object called nd-array (N-Dimensional array).

Working of nd-array is faster than compared to lists in python. Because a list can be used to represent a one-dimensional array. Using NumPy arrays we can create multi-dimensional arrays.

To get the version of NumPy.

import NumPy

print(numpy.\_\_version\_\_)

It can display the version in your system. I am using the latest version which is "1.23.4".

By creating nd-array you can use array (). In the array function, we have to pass the parameters either list or tuples only. For a clear explanation take an example.

import NumPy as np

a = np.array(42)

b = np.array([1, 2, 3, 4, 5])

c = np.array([[1, 2, 3], [4, 5, 6]])

d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(d)

print(a.ndim)

print(b.ndim)

print(c.ndim)

print(d.ndim)

print(type(a)

output:

[[[1 2 3]

[4 5 6]]

[[1 2 3]

[4 5 6]]]

0

1

2

3

<class 'NumPy. nd-array'>

**Difference between list and array:**

[1,2,3,4,5] -- list in comma separated values.

[1 2 3 4 5] -- Array in white space separated values.

**Higher Dimensional Arrays:**

An array can have any number of dimensions.

We create an array you can define the number of dimensions by using the nd-min.

**Example:**

Create an array with 5 dimensions:

import NumPy as np

arr = np.array([1, 2, 3, 4], ndmin=5)

print(arr)

print('number of dimensions:', arr.n-dim)

output:

[[[[[1 2 3 4]]]]]

number of dimensions: 5

**Access 3-D Arrays:**

Access elements from 3-D arrays.

**Example:**

Access the 10 values in the array by using indexing.

import NumPy as np

arr = np. array ([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

print(arr[1, 1, 0])

Output:

10

**Data Types in NumPy:**

NumPy has some different data types that are mentioned below.

• i - integer

• b - boolean

• u - unsigned integer

• f - float

• c - complex float

• m - time delta

• M - date time

• O - object

• S - string

• U - Unicode string

• V - a fixed chunk of memory for other types (void).

**The Difference Between Copy and View:**

In NumPy copy, will creates a new array it can't change the existing array.

In NumPy view is just a view of the original array.

**Example for copy():**

import NumPy as np

arr = np. array ([1, 2, 3, 4, 5])

x = arr.copy()

arr[0] = 42

print(arr)

print(x)

Output:

[42 2 3 4 5]

[1 2 3 4 5]

Copy can't change the existing array. It can change newly created arrays only.

**View example:**

import numpy as np

arr = np.array([1, 2, 3, 4, 5])

x = arr.view()

x[0] = 31

print(arr)

print(x)

Output:

[31 2 3 4 5]

[31 2 3 4 5]

By using view() here both existing and newly created arrays are changed.

**The shape of an array:**

Here the shape of an array can be defined as the number of elements in each dimension. By using the shape attribute, we can get the final output that should be in the tuple format. In that tuple, it gives the length of an array.

Syntax: NumPy.shape(array\_name)

**Example:**

Import NumPy as np

arr = np. array ([[1,3,4,5], [2,4,6,8]])

print(arr.shape)

output:

(2, 4)

**Reshape of an array:**

It can change the shape of an array without changing the input data.

**Example:**

import NumPy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])

new\_arr = arr.reshape(4, 3)

print(new\_arr)

Output:

[[ 1 2 3]

[ 4 5 6]

[ 7 8 9]

[ 10 11 12]]

**Searching an array:**

By using where () you can search an array for a particular value and return the value index that is matched to the certain array.

**Example:**

import NumPy as np

arr = np. array([1, 4, 3, 4, 5, 4, 4])

x = np.where(arr == 4)

print(x)

Output:

(array([1, 3, 5, 6])

In the above example, element four can be searched for the entire array and it gives the fourth value index that is 1, 3, 5, 6.

**Sorting Array:**

An array can be sorted with all the elements in an ordered sequence.

In NumPy they have one pre-defined function called sort ().

**Example:**

import numpy as np

arr = np.array([3, 2, 0, 1])

print (np. sort(arr))

Output:

[0 1 2 3]

**Filtering Arrays:**

The filter should follow the Boolean operator either True or False.

Create a new array using an existing array to get some elements that elements must and should be True. In that case, it gives output. It checks the index of the true value and gives the output as a value.

**Example:**

import NumPy as np

arr = np.array([41, 42, 43, 44])

x = arr[[True, False, True, False]]

print(x)

Output:

[ 41, 43 ]

**Arrange ():**

By using arrange function it can create an array with a collection of space-separated values.

**Difference between range () and arrange () functions:**

By using the range () function you do not have to install any module we can't mention any data type in the range function.

By using arrange () function you have to install the NumPy module. In NumPy specify the data type as d-type in arrange.

**Example:**

import NumPy

a=numpy.arange(0,15,1,dtype='int')

print(a)

output:

[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14]

Arrange function is similar to the range function. 0 -- start

15 -- stop

1 -- step

d-type –- type of the data.

**NumPy u-function:**

create a function using u-function.

Ufunc -– stands for universal function.

First, you have to define a function the same as python after creating functions, then you add these functions NumPy u-function library with from-py-function () function. Frompyfunc () allows the creation of an arbitrary Python function (variable length arguments) and the result is a Numpy u-function.

create a u-function by adding two numbers.

**Example:**

import numpy as np

def myadd(x, y):

return x+y

myadd = np.frompyfunc(myadd, 2, 1)

print(myadd([1, 2, 3, 4], [5, 6, 7, 8]))

Output:

[6 8 10 12]

In the above example, myadd is the function name.

We can give inputs in functions that are parameters.

Call the function by using the function name.

Check if the function is ufunc or not.

A ufunc return <class 'numpy.ufunc'>.

**Summations:**

The difference between addition and summation is addition performs two arguments and summation can perform n number of elements.

**Example for summation:**

import numpy as np

arr1 = np.array([1, 2, 3 ,4, 5])

arr2 = np.array([1, 2, 3, 4, 5])newarr = np.sum([arr1, arr2], axis=1)

print(newarr)

output:

[15 15]

In the above example arr1 elements can be added that is 15 and next add arr2 and finally, it displays the result

**Random number:**

You can find the random number by using the random module.

A random module is a predefined module in python.

A random number generates a random value to a particular range.

**Example:**

from numpy import random

x = random.randint(100)

print(x)

output:

35

**Ufunc in numpy:**

ufunc – stands for universal function.

These ufunc are used to implement vectorization in NumPy. it's faster than iteration.

**Vectorization:**

Iterative statements can Convert into a vector-based operation called vectorization.

**Example:**

import numpy as np

x = [1, 2, 3, 4]

y = [4, 5, 6, 7]

z = np.add(x, y)

print(z)

print(type(np.add))

Output:

[5 7 9 11]

<class, 'numpy.ufunc'>

**ufunc differences:**

The discrete difference means subtracting two elements in a particular array.

**Example:**

import numpy as np

arr = np.array([10, 15, 25, 5])

newarr = np.diff(arr)

print(newarr)

output:

[ 5 10 -20 ]

In the above example first two numbers in the array 10 and 15 can be subtract 10 - 15 = 5, 15 – 25 = 10, 25 – 5 = -20.

**REQUESTS**

Requests is a Python module. You can send all kinds of HTTP requests using the requests module.

Requests help you to make HTTP calls programmatically.

Users can send the request to the web browser. Web browsers can add some facilities, methods, and headers that can be transferred to a web server. The web server gives the response for the user requirement.

The requests module can be used to scrape the data from the website.

HTTP – stands for Hyper Text Transfer Protocol. It is a request-response protocol between a client and a server. or Communication between the client and server is called request-response protocol.

**Download requests module:**

Go to pypi.org and search for requests and download the latest version. Present "2.28.1 "is the latest version it shows one command.

pip install requests

After installation is completed, we can use it in your application.

URL stands for Uniform Resource Locator.

https://www.w3schools.com

HTTP – protocol.

www.w3schools – domain name or server name.

HTTP protocol supports some methods that are mentioned below.

GET

POST

PUT

DELETE

COPY

HEAD

OPTIONS and many more.

**Requests get method:**

In the get method, we are expecting something from the server. By using the get method we can fetch the data.

In python get () method can use to create a dictionary.

In requests, the library calls the URL with requests.get ().

We can't send data by using the HTTP get method. Because it does not have a message body. get method can bookmark means it's like a placeholder to store the data we can use in near future.

**Example:**

import requests

req = requests.get('https://www.flipkart.com/')

# Page encoding

e = req. encoding

print ("Encoding: ", e)

# Response code

s = req. status code

print("Response code: ",s)

# Response Time

t = req.elapsed

print("Response Time: ",t)

t = req.headers['Content-Type']

print("Header: ",t)

z = req.text

print ("\nSome text from the web page:\n", z [0:200])

Output:

Encoding: utf-8

Response code: 200

Response Time: 0:00:01.160887

Header: text/html; charset=utf-8

Some text from the web page:

<!doctype html><html lang="en"><head><link href="https://rukminim1.flixcart.com" rel="preconnect"/><link rel="stylesheet" href="//static-assets-web.flixcart.com/fk-p-linchpin-web/fk-cp-zion/css/app\_mo

**Encoding:**

encode () method encodes the string. if we can't specify the encoding method it can take by default utf – 8.

**UTF - 8:**

UTF – stands for Unicode Transformation Format.

Here 8 means 8 – bit value.

**status code:**

After completing execution status code returns a number for the given URL. The most used status codes are mentioned below.

Syntax: response. status code

Status code Information 200 OK requests successful. 204 No content to return but the response is satisfied. 304 Not modified but the response is satisfied. 400 Bad requests. 402 Payment required. 403 Forbidden (not allowed). 404 It shows the file not found (the property does not exist). 415 Unsupported media type. 500 Internal server error. 501 Not implemented (server does not support the requirement) 507 Insufficient storages.

**Elapsed:**

Save the timestamp at beginning of the code start.

Save the timestamp at end of the code end.

Find the difference between start and end finally it gives the execution time.

Timestamp means the date and time of a particular event.

**Requests Post method:**

By using the post method, we can send some data to the server.

The get and post method is used to transfer data from client to server using the HTTP protocol.

The post method carries request attributes in the message body.

The post-request method doesn't have any restrictions to store the length of the data.

**Example:**

import requests

values = {'username': 'deepika','password': '123'}

res = requests.post('https://httpbin.org/post', data = values)

print(res.text)

Output:

{

"args ": {},

"data": " ",

"files": {},

"form": {

"password": "123",

"username": "deepika"

},

"headers": {

"Accept": "\*/\*",

. . . . . . . .

**Requests Put method:**

If the resource is not available put method can create a resource.

By using the put method you can update resources available on the server.

The put method overwrites the existing resource.

After overwriting the existing resource.

the existing resource can be removed.

**Requests Delete method:**

The delete () method sends a DELETE request to the given URL.

**Requests Copy method:**

Copy method creates a duplicate resource both source and destination, destination resource can be identified by the request URI in the destination header. We can check whether the destination header is present or not. The destination header must be present.

Copy method behavior depends on the source resource.

URI – stands for Uniform Resource Locator.

**Headers:**

we can use headers in the get method by using the python request library. While creating a header in python. First take a dictionary with both key and its value {key: value} the key represents the name of the header.

The value represents the content of the header.

**Text:**

Here text can display everything present in the given URL. It can be in HTML format.

**Download a file using the request method:**

Download a file using the python request library using the shuttle, this shuttle method collects the files and handles high-level files easily.

Copy the content of one file object to another file object.

**Example:**

import shutil

import requests

url = 'https://reqbin.com/echo/get/json'

response = requests.get(url, stream=True)

with open('sample.json', 'wb') as out\_file:

shutil.copyfileobj(response.raw, out\_file)

print('The file was saved successfully')

In the above example file is in byte code because we can give access mode wb.

**REQUESTS-HTML**

First, you know the basic Html and go to requests-Html.

**HTML:**

HTML – stands for Hyper Text Markup Language.

By using Html, we can create web pages.

In Html, the title can display in a web browser and the body can display the Html page.

The purpose of a web browser is to read Html documents after reading them it can display them.

Web browsers can't display Html tags, but they can use them to display the document.

**Html tags:**

<html> -- root element, <head> -- meta information, <title> -- title of the Html page, <body> -- container (content of the Html page), <h1> -- large heading, <h2>, <h3>, <h4>, <h5>, <h6> (h1 large heading then compared to h2 it decreases the size one by one), <p> -- paragraph.

Element is defined by starting tag, some useful information, and an ending tag.

<starting tag> useful information. . .. <ending tag>

Some Html elements do not have any content.

These elements are called empty elements in Html.

An empty element does not have any end tag.

<br> -- line break.

< > -- this is anchor brackets. In between anchor brackets, we can write Html tags.

<head> -- this is the opening tag Html content</head> -- this is the closing tag in front of they have forward slash.

These tags can also be nested in Html.

**Requests-html:**

The requests-Html library is an Html parser.

Html parser uses CSS selectors and XPath selectors to pull the information from the web page.

**HTML-Parser:**

Html parser is a software. By using an Html parser data can pull after that it can leave the tags.

**Example:**

Input: <h1> This is my Html content </h1> After the h1 tag, I enter one tab space. Here the parser ignores white space.

<h1> -- It is the open tag and </h1> -- It is the closed tag.

Output:

This is my Html content.

**CSS**

CSS – stands for cascading style sheet.

CSS is the language used to style a Web page colorful.

CSS syntax consists of a selector and declaration block.

You can't forget to put a semi-colon in between two declarations.

**CSS syntax:**

h1 {colour: blue; font-size: 12px;}

h1 --- selector.

Colour: blue --- declaration.

Font-size: 12px --- declaration.

Colour --- property.

Blue --- value.

**CSS Selector:**

We use CSS selectors to select Html elements that you want to style.

**XPath Selectors:**

XPath stands for XML path.

XML stands for an extensible markup language.

XPath selector means the path of the particular Html tags.

Web scraping is the best example for requests-Html.

In the requests Html library, we use the requests library to call the URL by making HTTP requests to a web server.

Here you install both requests and requests-HTML library.

pip install requests

pip install requests-HTML

**Html session method:**

The Html session method starts the get requests and the. get () function requests to call URL from scratch.

pip install requests-HTML

**requests-Html get method:**

This module provides a session object to manage and persist settings.

**Example:**

using requests Html library

import requests

from requests\_Html import HTMLSession import requests

session = HTMLSession ()

r = session.get('https://python.org/')

print(r.text)

Output: Output is in the form of an Html page.