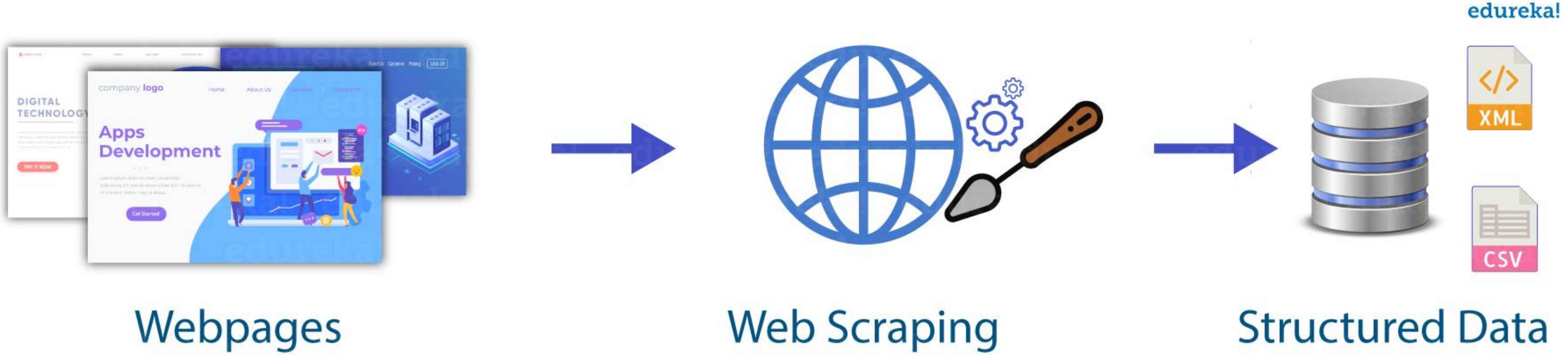
Web Scraping



NAYA College

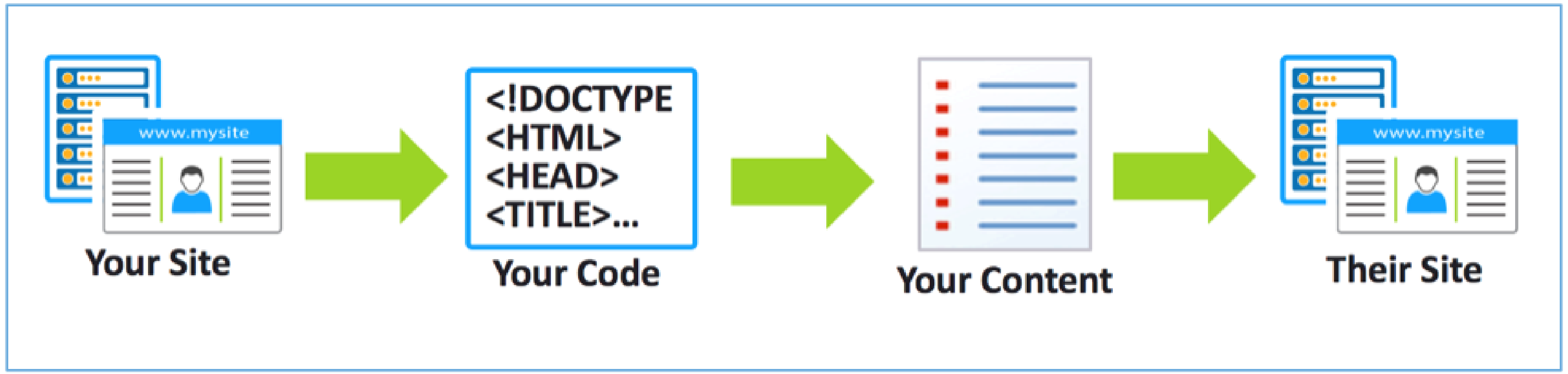
Agenda

- Introduction
- HTTP (HyperText Transfer Protocol)
- The ”urllib” package
- The ”requests” module
- HTML (Hyper-Text Mark-up Language)
- The ”BeatifulSoup” module

Introduction

Web scraping is the process of using bots to extract content and data from a website. Unlike screen scraping, which only copies pixels displayed onscreen, web scraping extracts underlying HTML code and, with it, data stored in a database. The scraper can then replicate entire website content elsewhere.

Web scraping software may access the World Wide Web directly using the Hypertext Transfer Protocol, or through a web browser.

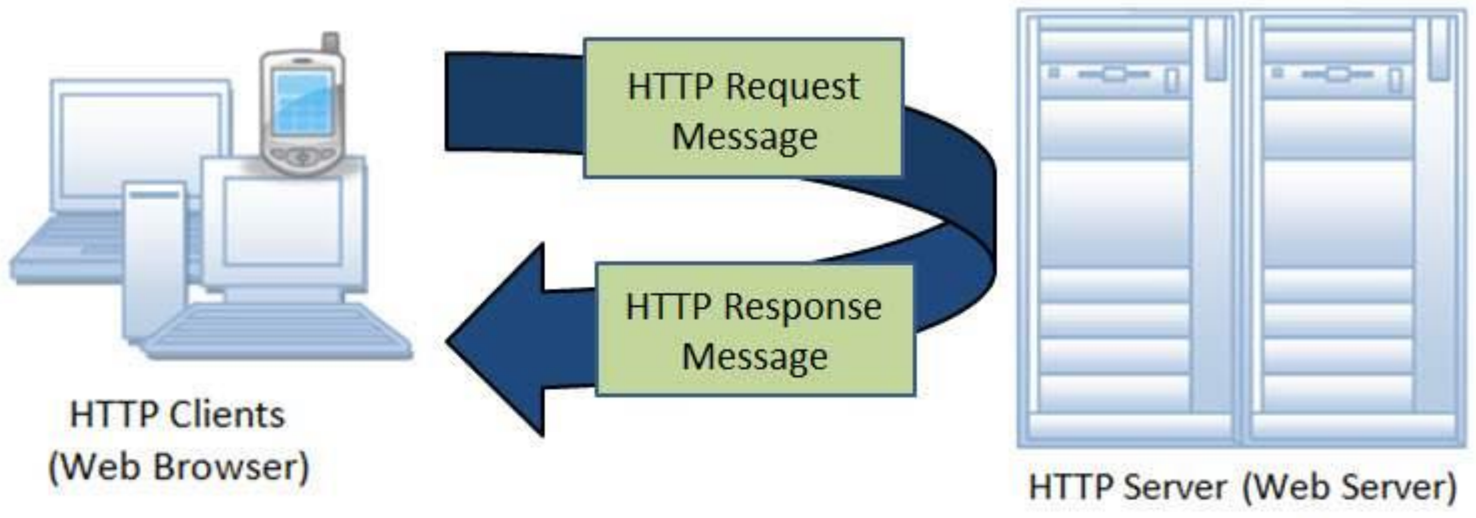


NAYA College

HTTP (HyperText Transfer Protocol)

HTTP is a protocol for distributed, collaborative, hypermedia information systems, and it is the foundation of data communication for the World Wide Web.

As described in the figure below, when an HTTP client wishes to communicate with an HTTP server it sends the appropriate request and gets in return a response. Both requests and responses must comply with the details of the protocol, which can be explored here.



The urllib package

urllib is part of the PSL and is a package that collects several modules for working with URLs (request, error, parse and robotparser).

The *urllib.request* module defines functions and classes which help in opening URLs (mostly HTTP) in a complex world — basic and digest authentication, redirections, cookies and more. The function *urlopen(url)* returns a file-like object which contains the data of the specified url.

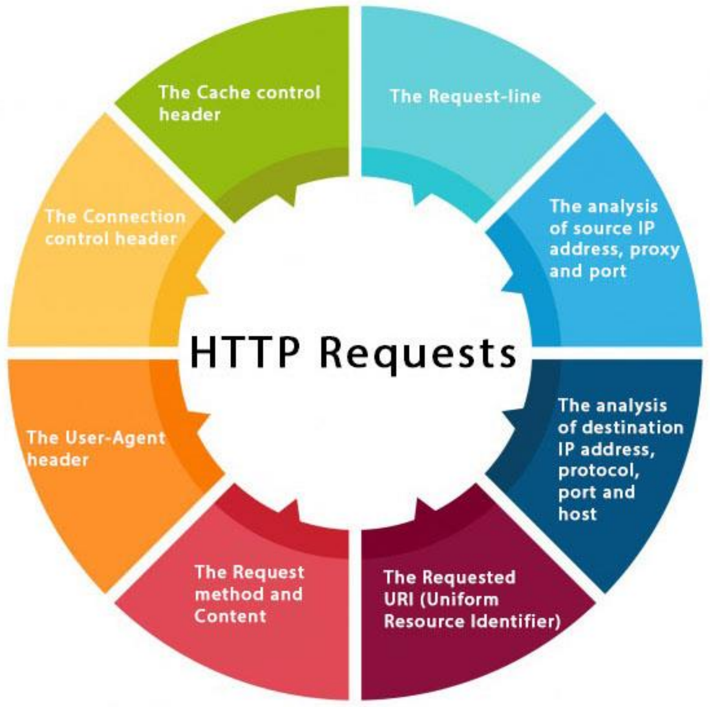
NOTE: The textual content of a webpage, a.k.a. **source code**, can be (usually) seen by clicking the "View page source" option in the right-click menu (Ctrl+U in Chrome).

```python
import urllib.request
```

The urllib package

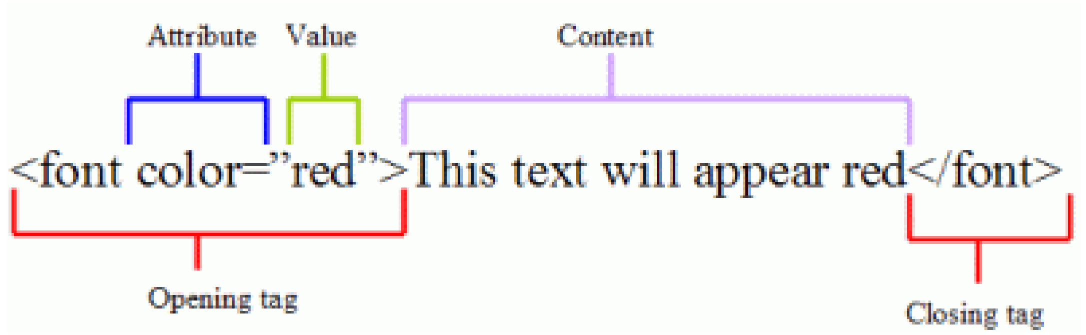
The requests module

The *requests* module is the *de facto* standard for working with HTTP in Python, and even the formal documentation of *urllib* recommends to use it. Unlike the *urllib* module, the *requests* module is not part of the PSL and should be installed separately. It is, however, included in the Anaconda package.


HTTP Requests
The Request-line
The analysis of source IP address, proxy and port
The analysis of destination IP address, protocol, port and host
The Requested URI (Uniform Resource Identifier)
The Request method and Content
The User-Agent header
The Connection control header
The Cache control header

HTML (Hyper-Text Mark-up Language)

HTML documents are textual, yet not conveniently readable by humans. The reason for that is the way this language is structured. HTML documents contain a variety of (possibly nested) **HTML elements** with various purposes, e.g.: text and paragraph formatting, elements embedding and hyperlinks.



The requests module & HTTP

The requests module & HTTP

Exresices:

The Wikipedia page Academy Award for Best Picture gives the details about the winners and nominees of the Oscar awards since its debut in 1929.
Use the source-code of this page to make a list of all the winning movies.

The BeatifulSoup module

Parsing an HTML document is an extremely tedious task, even when using regular expressions. Although the PSL contains a module named *HTMLParser*, the *de facto* module used for parsing HTML documents is an external module called *BeautifulSoup*, which is also included in the Anaconda package (it is a sub-module of the *bs4* module, as seen by the *import* statement).

Due to the recursive nature of its elements, the HTML document is regarded as a "tree" of elements. The module's then facilitates **navigating** the tree and **searching** it using the various elements parts, as we will see. The main tree has a class of its own, surprisingly called *BeautifulSoup*, and the tree is traditionally called ***soup***.

It worth mentioning that *BeautifulSoup* objects are a special case of the more general class *tag*, that we will explore immediately.

The BeatifulSoup module

- Import Package

```
from bs4 import BeautifulSoup as bs
```

- Get bseatifulSoup Class

```
soup = bs(resp.text, 'html.parser')
```

- *BeautifulSoup* objects support a very nice method that helps to visualize the content of the HTML document.

```
soup.prettify()
```

The BeatifulSoup module

The main tool of the BeautifulSoup module is the tag class. Each **HTML element** is assigned by the BeautifulSoup() constructor a corresponding tag instance, which contains the entire data of the element. tag elements support many attributes and methods, but we will focus on the method findall() and the attribute string.

- **name** is the name of an html tag type (‘a’, ‘p’, ‘div’, etc.)
- **attrs** is a dictionary of key-value pairs where the key is an html attribute name and the value is the value you want to match.
- **recursive** controls whether to find descendents (the default) or just children (recursive=False)
- **string** allows you to find NavigableString nodes instead of Tag nodes.
- **limit** controls how many to find, maximum.

```
from bs4 import BeautifulSoup as bs
```

The BeatifulSoup module

- Print soup with ‘a’ class

```
print(soup.a)
Answer: <a class="news" href="http://www.ynet.co.il" id="link1">ynet</a>
```

- Print on attr

```
link = soup.a
print(link.attrs)
Answer: {'href': 'http://www.ynet.co.il', 'class': ['news'], 'id': 'link1'}
```

The BeatifulSoup module

- Find All ‘a’ class

```
links = soup.find_all('a')
print(links)
Answer: [<a class="news" href="http://www.ynet.co.il" id="link1">ynet</a>,
<a class="news" href="http://www.israelhayom.co.il/" id="link2">Israel Hayom</a>,
<a class="news" href="https://www.haaretz.co.il/" id="link3">Haaretz</a>,
<a class="sport" href="http://www.one.co.il/" id="link4">One</a>,
<a class="sport" href="http://www.sport5.co.il/" id="link5">Sport 5</a>,
<a class="economic" href="https://www.themarker.com/" id="link6">TheMarker</a>,
<a class="economic" href="https://www.calcalist.co.il" id="link7">Calcalist</a>]
```

The BeatifulSoup module

- Find All ‘a’ class and Filter by attribute value

```
links = soup.find_all('a', class_="news")
print(links)
Answer:
link1: ynet - http://www.ynet.co.il
link2: Israel Hayom - http://www.israelhayom.co.il/
link3: Haaretz - https://www.haaretz.co.il/
```

The BeatifulSoup module

Find by regex. we can pass a regular expression object, Beautiful Soup will filter against that regular expression using its search() method.
NOTE: a regular expression object is obtain by using the complie function in the re module. see re.compile.

- This code finds all the tags whose names start with the letter “b” but not followed by the letter "r"

```
for tag in soup.find_all(re.compile("^b[^r]+")):
    print(tag.name)
Answer:
body
blockquote
```

The BeatifulSoup module

- Find by string

```
print(soup.find_all(string='item 4’))
Answer: ['item 4']
```

- as with tag names, we can pass a regular expression object

```
print(soup.find_all(string=re.compile(".* goal .*")))
Answer: ['A goal is a dream with a deadline.']
```

- Limit - find_all() returns all the tags and strings that match your filters. This can take a while if the document is large. If you don’t need all the results, you can pass in a number for limit

```
print(soup.find_all('a', limit=5))
```

The BeatifulSoup module

- if you need only single (the first) result or you know there is only signle item of what you are looking for, you can use *find()* (instead of passing limit=1 to find_all)

```
print(soup.find('a', class_='sport'))
```

- do we want Beautiful Soup to search the all tree (children, its children's children, and so on) , or just for direct children?

```
print(soup.body.find_all(class_='list', recursive=False))
```

The BeatifulSoup module

The BeatifulSoup module

Exresices:

The website IMDB show the rating of almost all TV series and movies. take a series you like with more than a
single season, and show a list of all episodes and their rating. you can assume you know the exact link to the serie
page and the number of seasons. for example, Game of Thrones has 8 seasons.

https://www.imdb.com/title/tt0944947/