Rotman

WEB SCRAPING WITH PYTHON

March 9, 2020   Prepared by Niti / FinHUB

Rotman School of Management
UNIVERSITY OF TORONTO

What you will learn?

To build a script that fetches data from the web and displays the content in your machine in a readable format.

1. Basics of HTTP requests, HTML and CSS

2. Python's **requests** library to make HTTP request

3. Python's **BeautifulSoup** library to handle HTML processings

Rotman

Agenda

1. Web Scrapping

2. Connecting to the Data
   - HTTP requests and responses
   - Python's **requests** library

3. Getting the Data
   - Inspecting your Data
   - HTML
   - CSS
   - Python's **BeautifulSoup**

Rotman

1. Web Scrapping

1.1 What is Web Scrapping?

- "Constructing a program to download, parse and organize data from the web in an automated manner"

- Transfer large amount of data from online source and store it for later use

- Web scrapping focuses on the transformation of unstructured data on the web into a more structured format

Rotman

1.2 Why Web Scrapping is useful?

- Web exposes interesting opportunities:
  - ➤ Reviews
  - ➤ Wikipedia
  - ➤ Social networks
  - ➤ Weather information, etc.

- Google Translate, for instance, utilizes text sources on the web to train and improve itself

Rotman

1.3 What is an API?

- Application Programming Interface (APIs)

- Programs provided by websites to access their data repository in a structured way

- With API, you can avoid parsing messy HTML documents

- The process is generally more stable than web scrapping

- Lack of quality documentation can make it harder to inspect the structure of API

Rotman

1.4 Why Web Scrapping over API?

- no API for that website

- API is not free

- API has rate limits

- API does not provide all the information you want

Rotman

1.5 Word of Caution!
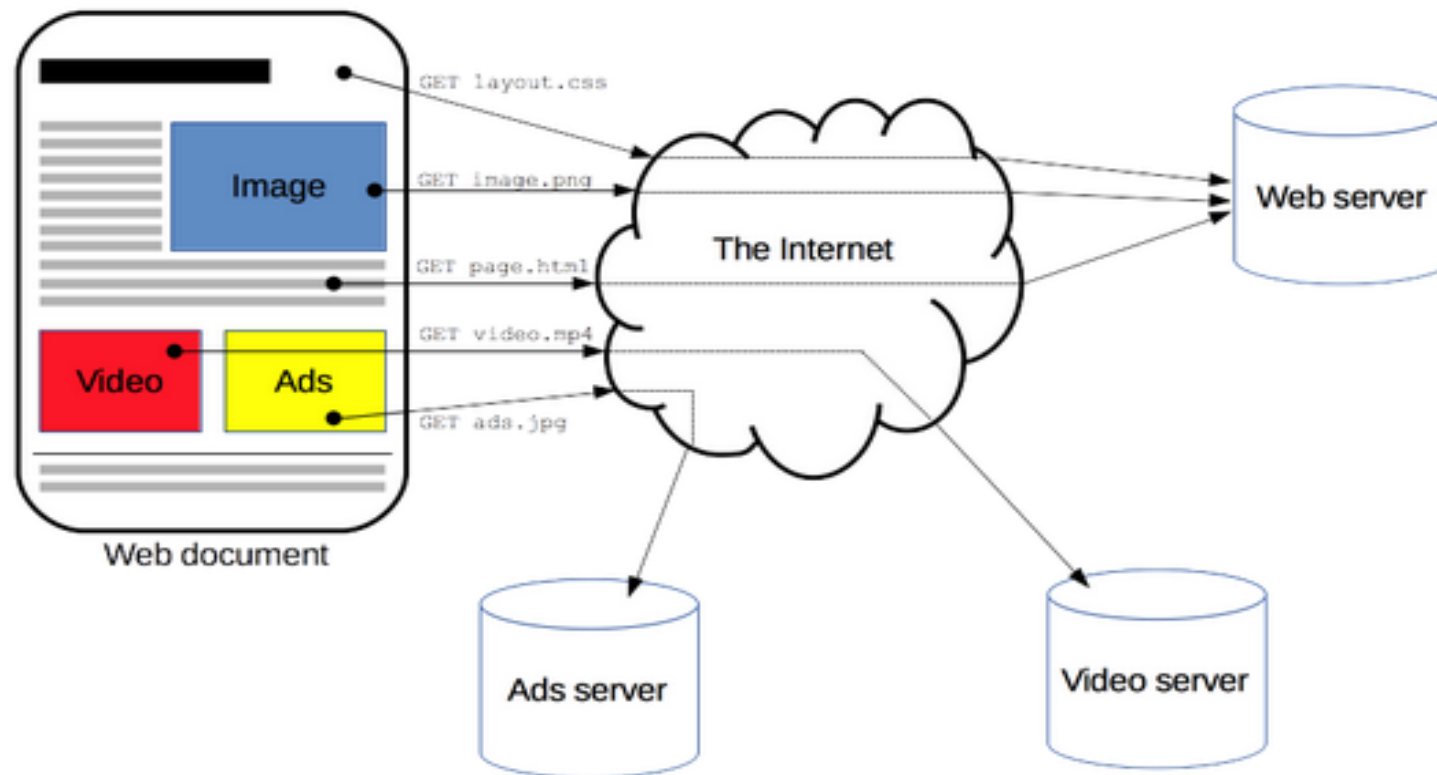
Some websites don't like it when automatic scrapers gather their data

➢ *Legal Perspectives on Scraping Data From the Modern Web*

➢ *Introduction to robots.txt*

while others don't mind _('.')_/
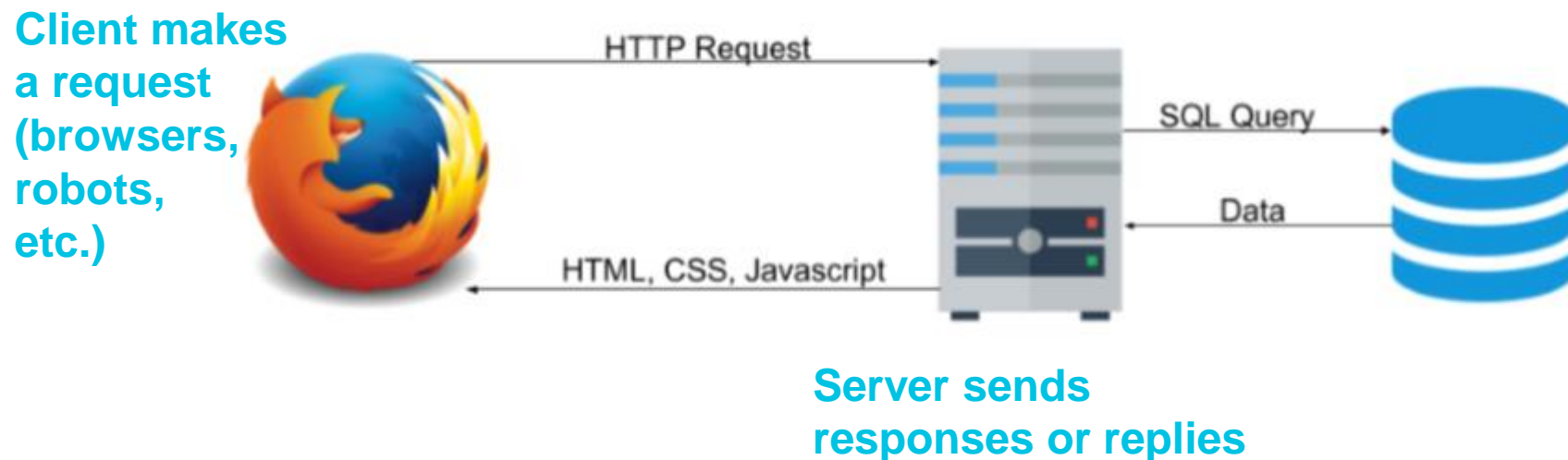
Rotman

2. Connecting to the Data

A massive distributed client/server information system

Source: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

Rotman

2.2 HTTP

- HyperText Transfer Protocol

- Client-server protocol that is the foundation of data exchange on the Web

- HTTP client **sends requests** to an HTTP server, which in turn **returns a response message**.

Client makes a request (browsers, robots, etc.)

HTTP Request

SQL Query

Data

HTML, CSS, Javascript

Server sends responses or replies

Rotman

1. Transactional

   ➤ Refers to a single HTTP request and the corresponding HTTP response

2. Stateless (not session-less)

   ➤ The current request does not know what has been done in the previous requests

Rotman

2.3 HTTP Methods

- GET
- POST
- PUT
- HEAD
- DELETE
- PATCH
- OPTIONS

Rotman

2.4 HTTP GET Request

Whenever you enter a URL in the address box of the browser:

Action: Google "HTTP" on google.com

GET /search?hl=en&q=HTTP&btnG=Google+Search HTTP/1.1
Host: http://www.google.com
User-Agent: Mozilla/5.0 Galeon/1.2.0 (X11; Linux i686; U;) Gecko/20020326
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,
     text/plain;q=0.8,video/x-mng,image/png,image/jpeg,image/gif;q=0.2,
     text/css,*/*;q=0.1
Accept-Language: en
Accept-Encoding: gzip, deflate, compress;q=0.9
Accept-Charset: ISO-8859-1, utf-8;q=0.66, *;q=0.66
Keep-Alive: 300
Connection: keep-alive

Rotman

2.4 HTTP GET Request

Request Method

Version of the protocol

Request Line

GET /search?hl=en&q=HTTP&btnG=Google+Search HTTP/1.1
Host: http://www.google.com
User-Agent: Mozilla/5.0 Galeon/1.2.0 (X11; Linux i686; U;) Gecko/20020326
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,
    text/plain;q=0.8,video/x-mng,image/png,image/jpeg,image/gif;q=0.2,
    text/css,*/*;q=0.1
Accept-Language: en
Accept-Encoding: gzip, deflate, compress;q=0.9
Accept-Charset: ISO-8859-1, utf-8;q=0.66, *;q=0.66
Keep-Alive: 300
Connection: keep-alive

Request Headers

Rotman

2.5 HTTP GET Response

What you get in response:

Response of "HTTP" search on google

HTTP/1.1 200 OK
Server: GWS/2.0
Date: Tue, 21 May 2002 12:34:56 GMT
Transfer-Encoding: chunked
Content-Encoding: gzip
Content-Type: text/html
Cache-control: private
Set-Cookie: PREF=ID=58c005a7065c0996:TM=1021283456:LM=1021283456:S=OLJcXi3RhSE;
        domain=.google.com; path=/; expires=Sun, 17-Jan-2038 19:14:07 GMT

(Web content compressed with gzip)

Rotman

2.5 HTTP GET Response

Response
Headers

Response
Message
Header

Blank line separates
header and body

Response
Message
Body

HTTP/1.1 200 OK
Server: GWS/2.0
Date: Tue, 21 May 2002 12:34:56 GMT
Transfer-Encoding: chunked
Content-Encoding: gzip
Content-Type: text/html
Cache-control: private
Set-Cookie: PREF=ID=58c005a7065c0996:TM=1021283456:LM=1021283456:S=OLJcXi3RhSE;
                    domain=.google.com; path=/; expires=Sun, 17-Jan-2038 19:14:07 GMT

(Web content compressed with gzip)

Status Line:
1. Version of protocol
2. Status code
3. Status message

Rotman
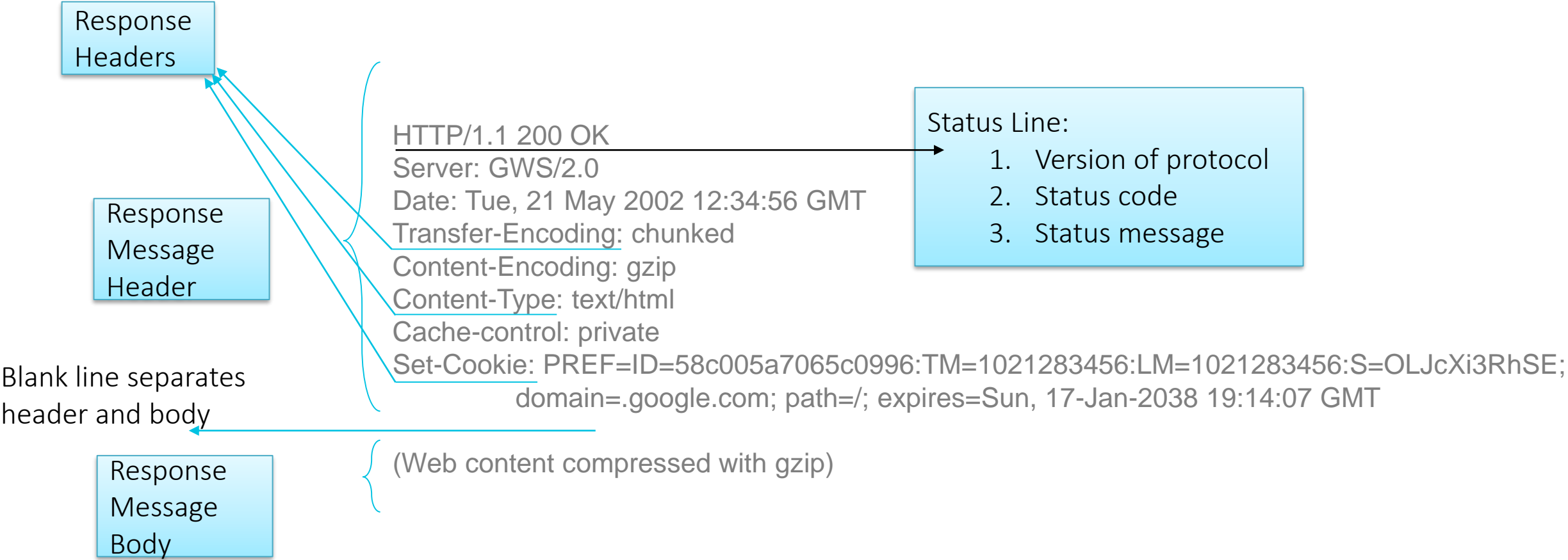
2.6 HTTP Requests with Python

- urllib : built-in Python module

- urllib3 : powerful HTTP client for Python

- requests : simple HTTP library built on top of urllib3
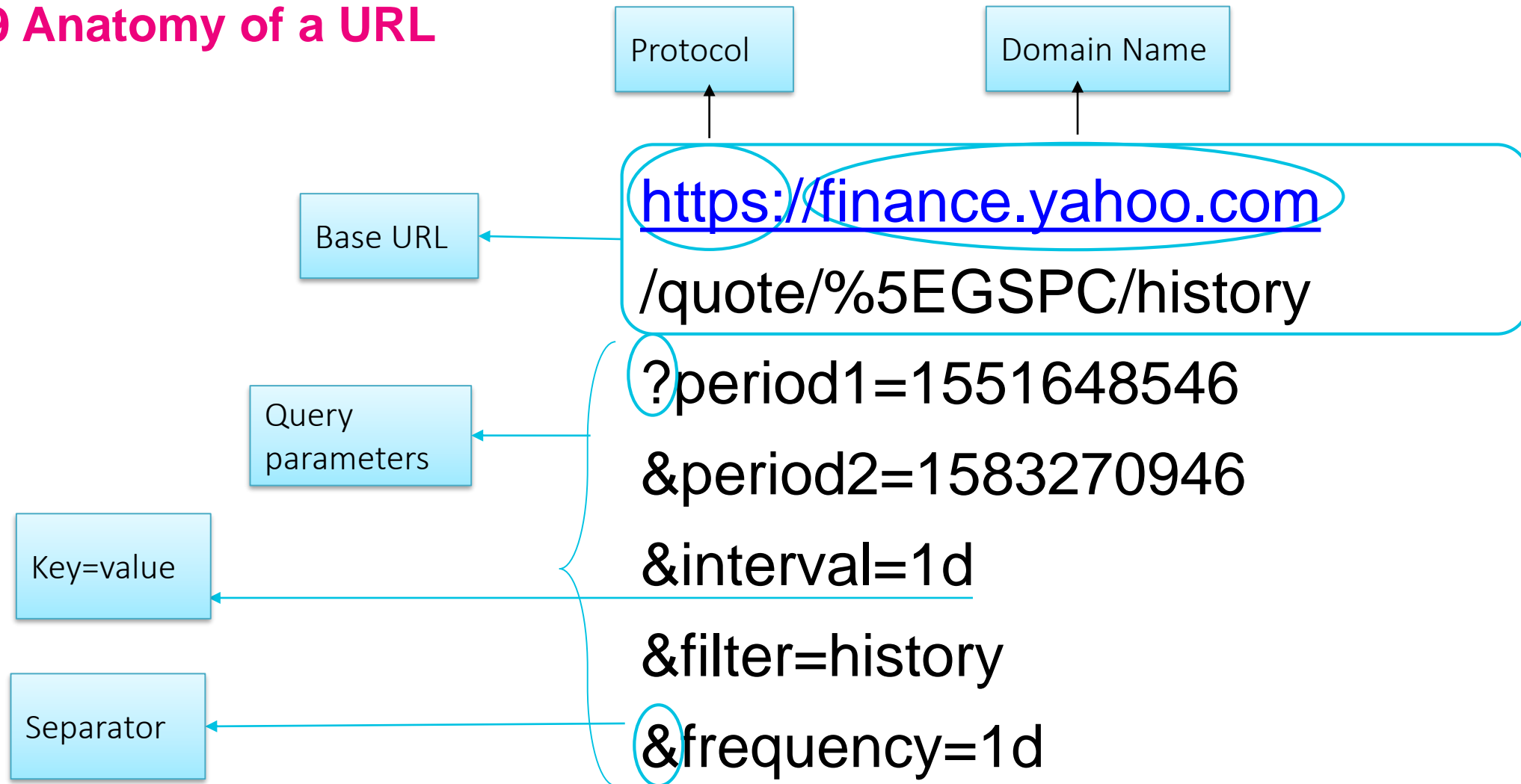
Rotman

2.7 requests Library

- simplifies the process of making HTTP requests

- built on top of "urllib3"

- allows you to tackle the majority of HTTP use cases in code that is short, pretty, and easy to use

- formats a proper HTTP request message in accordance with what we've seen before

- Install request through anaconda: https://anaconda.org/anaconda/requests

Rotman

2.8 URL

- Uniform Resource Locators: address of a given unique resource on the Web

https://finance.yahoo.com/quote/%5EGSPC/history?period1=1551648546&period2=1583270946&interval=1d&filter=history&frequency=1d

Rotman

2.9 Anatomy of a URL

Protocol

Domain Name

Base URL

https://finance.yahoo.com
/quote/%5EGSPC/history
?period1=1551648546
&period2=1583270946
&interval=1d
&filter=history
&frequency=1d

Query parameters

Key=value

Separator

Rotman

3. Getting the Data

3.1 Inspect Your Data Source

- Modern browsers have a powerful suite of developer tools that among other things also inspects currently-loaded HTML, CSS and JavaScript to show which aspects the page has been requested, how long it took to load, etc.

- Developer tools can help understand the structure of a website

- In Firefox, you can access it as:

Menu ➤ Web Developer ➤ Toggle Tools

Tools ➤ Web Developer ➤ Toggle Tools

Rotman

3.2 Structure of the Web Content

- Hypertext Markup Language (HTML)

- Cascading Style Sheets (CSS)

- JavaScript

Rotman

3.3 HTML

- Defines how a webpage is structured and formatted

- "Hypertext" refers to links that connect web pages

- "markup" annotates text, images and other content to display

- HTML consists of a series of elements, which can be used to enclose or wrap different parts of content for which tags are used

Rotman

3.4 HTML tags

- Tags may or may not come in pair

- Tags can be nested inside each other

- Paired tags have content

- Tags may have attributes such as class that provide additional information about an element

Rotman

3.5 Anatomy of HTML

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <img src="images/firefox-icon.png" alt="My test image">
  </body>
</html>
```

HTML Elements - Paired Tags

3.5 Anatomy of HTML

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <img src="images/firefox-icon.png" alt="My test image">
  </body>
</html>
```

HTML Elements - Unpaired Tags

Rotman

3.5 Anatomy of HTML

```html
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <title>My test page</title>
 </head>
 <body>
  <img src="images/firefox-icon.png" alt="My test image">
 </body>
</html>
```
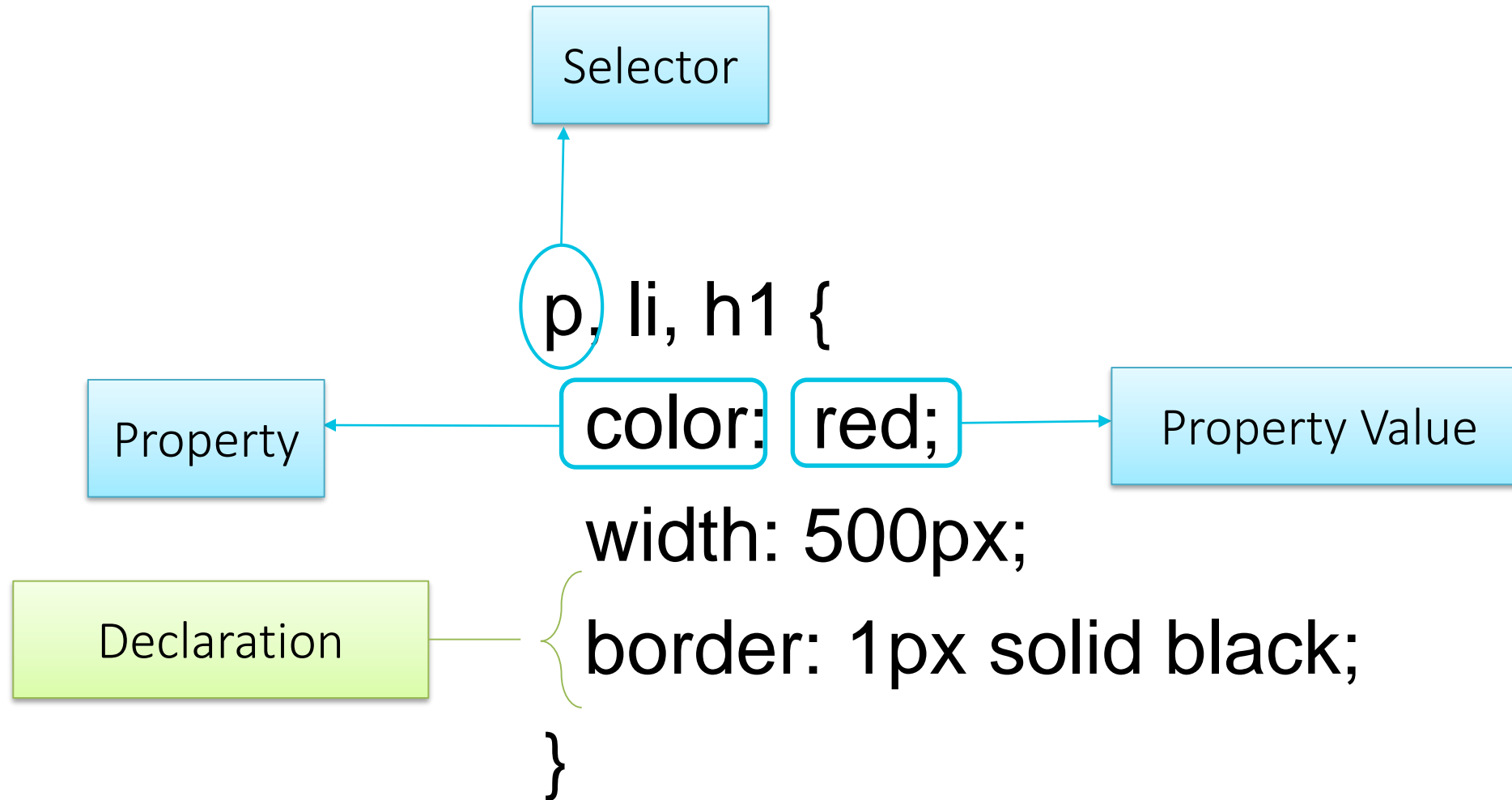
HTML Elements - Attributes

Rotman

3.6 CSS

- Defines the style and layout of a webpage

- Ex. alter the font, color, size, spacing of content, etc.

- Describes how elements should be rendered on screen

- Allows selective application of styles to HTML elements

Rotman

3.7 Anatomy of CSS

Selector

Property

Property Value

Declaration

```
p, li, h1 {
    color: red;
    width: 500px;
    border: 1px solid black;
}
```

Rotman

3.6 JavaScript

- JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions

- It is most well-known as the scripting language for Web pages

- HTTP requests to dynamic websites often return JS instead of HTML document

Rotman

3.6 BeautifulSoup

- Usually, we do not need the entire webpage but only certain contents of the webpage

- So we have to parse the HTML document to extract the content we want

- BeautifulSoup makes navigating HTML pages easy
    - Provides intuitive functions to parse structured data

Rotman

3.7 BeautifulSoup

- The tag name you are looking to find on the webpage
- Pass a string or a list of tags

find(name, attrs, recursive, string, **keywords)

find_all(name, attrs, recursive, string, **keywords, limit,)

Rotman

- Attributes to matches HTML elements
- Pass a Python dictionary of attributes

find(name, attrs, recursive, string, **keywords)

find_all(name, attrs, recursive, string, **keywords, limit,)

Rotman

3.7 BeautifulSoup

> - Depth of the search
> - If True (default), will look into children, children's children and so on
> - If False, looks at direct child elements only

find(name, attrs, recursive, string, **keywords)

find_all(name, attrs, recursive, string, **keywords, limit,)

Rotman

3.7 BeautifulSoup

- Match based on the test content of elements

find(name, attrs, recursive, string, **keywords)

find_all(name, attrs, recursive, string, **keywords, limit,)

Rotman

3.7 BeautifulSoup

- Limit the number of elements that are retrieved
- Find is same as find_all with limit set to 1,
  - ➢ except that find returns the element and find_all return a list of elements

find(name, attrs, recursive, string, **keywords)

find_all(name, attrs, recursive, string, **keywords, limit)

Rotman

3.7 BeautifulSoup

- Add extra named arguments, which will be used as attribute filters
- *find('id'='myid')* is same as *find(attrs={'id': 'myid'}*
- Cannot use class and name as a keyword

find(name, attrs, recursive, string, **keywords)

find_all(name, attrs, recursive, string, **keywords, limit,)

Rotman

Future Learnings

- Python's [Scrapy](#) library

- Scrapping dynamic websites that return JavaScript

    - Selenium

- Web crawling, search engine bot

Rotman

Questions?

Thank you