5/29 - Web Scraping

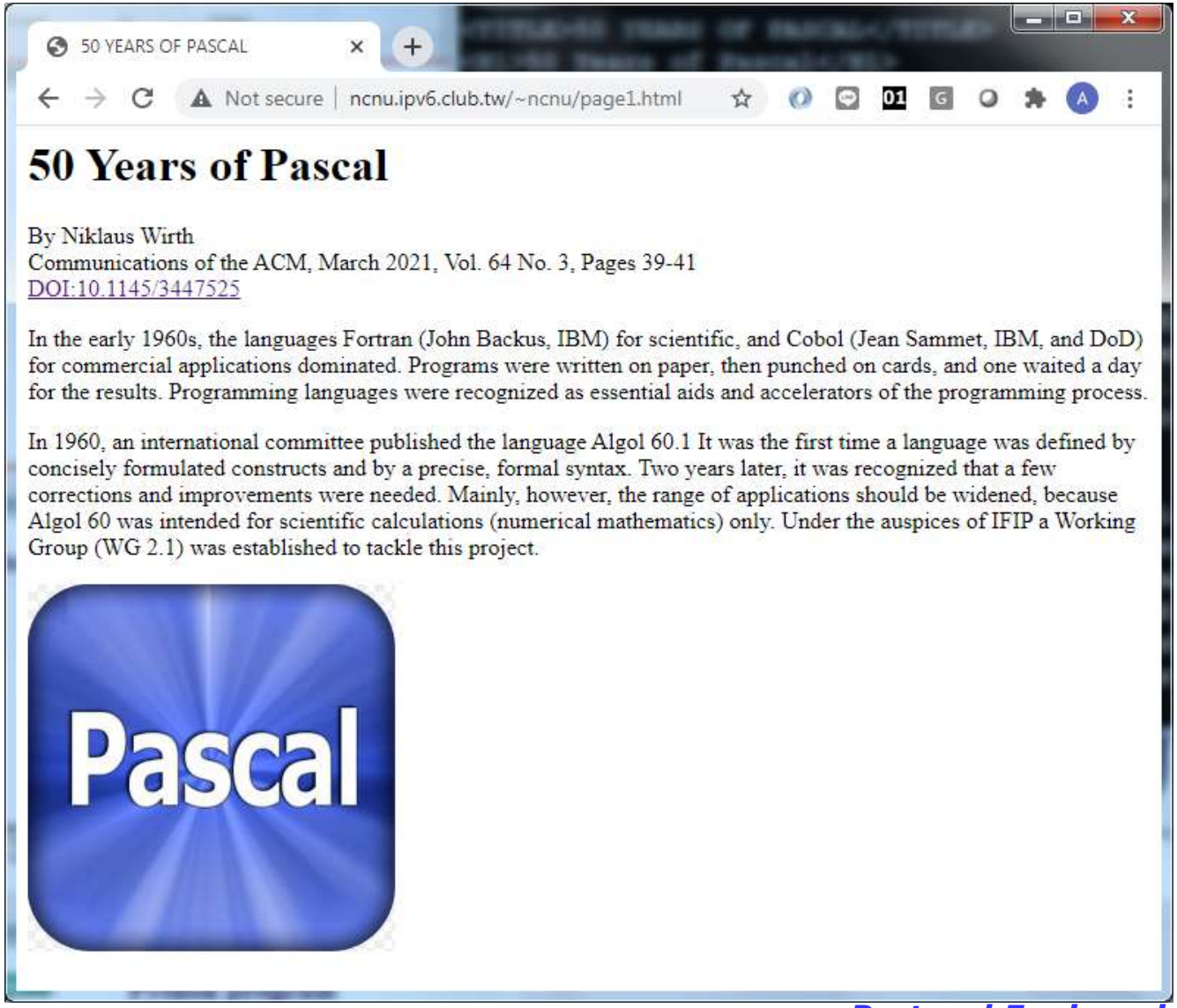
Retrieve a Remote File

Knowing HTML

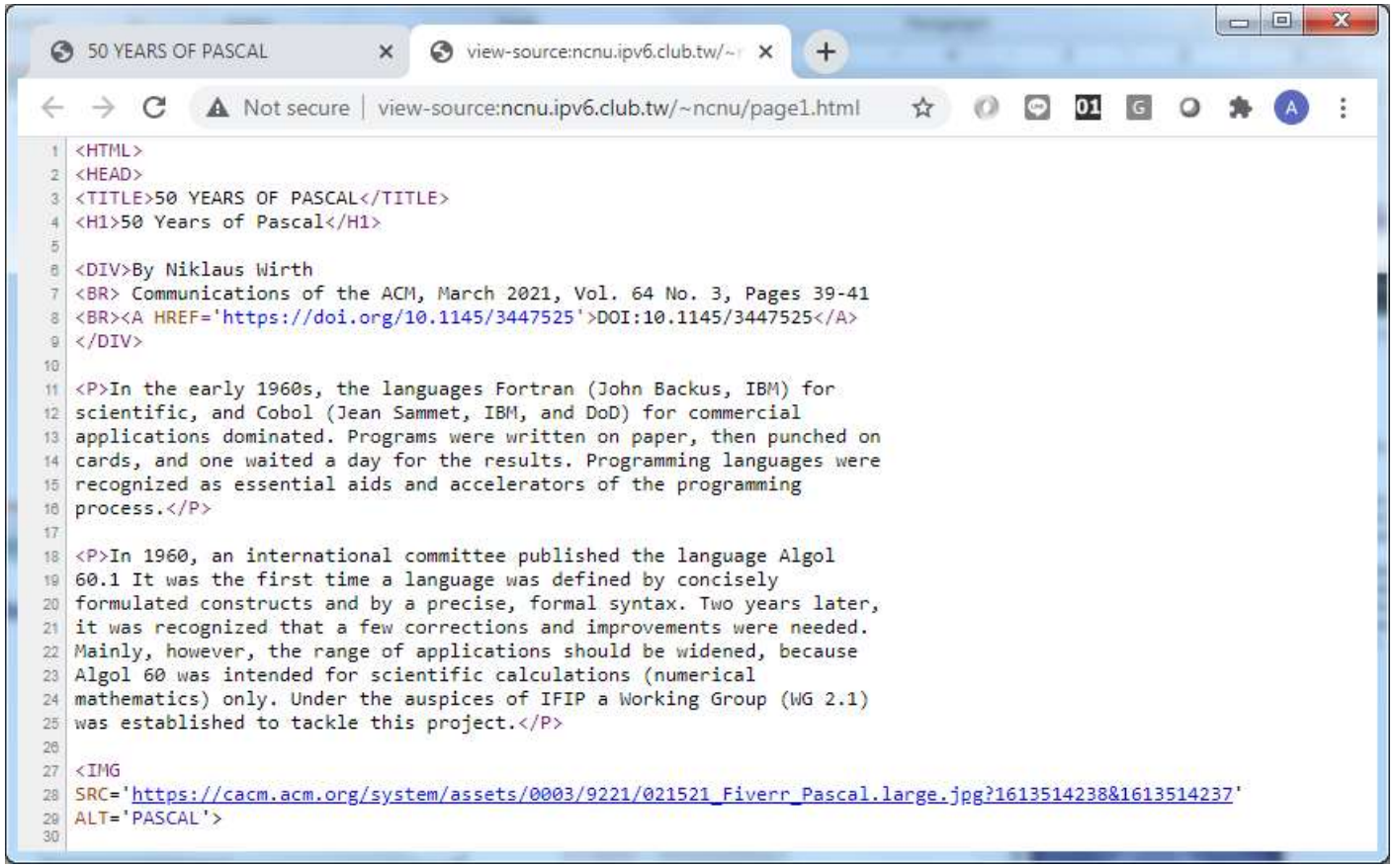
Parsing an HTML File

Result from Google Form

Webpages you see in a browser …



is actually an HTML file interpreted and formatted by software



Exercise:

1. Visit a webpage, e.g., http://poet.ncnu.org/page1.html
2. View its HTML source (by Ctrl-U)

You can get the HTML file by a Python program

Remark

```
# Retrieve HTML string from a URL
```

module function

```
from urllib.request import urlopen
```

protocol host filename

```
url = "http://poet.ncnu.org/poem02.txt" # Universal Resource Locator

data = urlopen(url).read()
```

Python sees a remote file similarly as a local file. You can "read" it into a variable.

```
html = data.decode()
```

Decode from bytes to str.

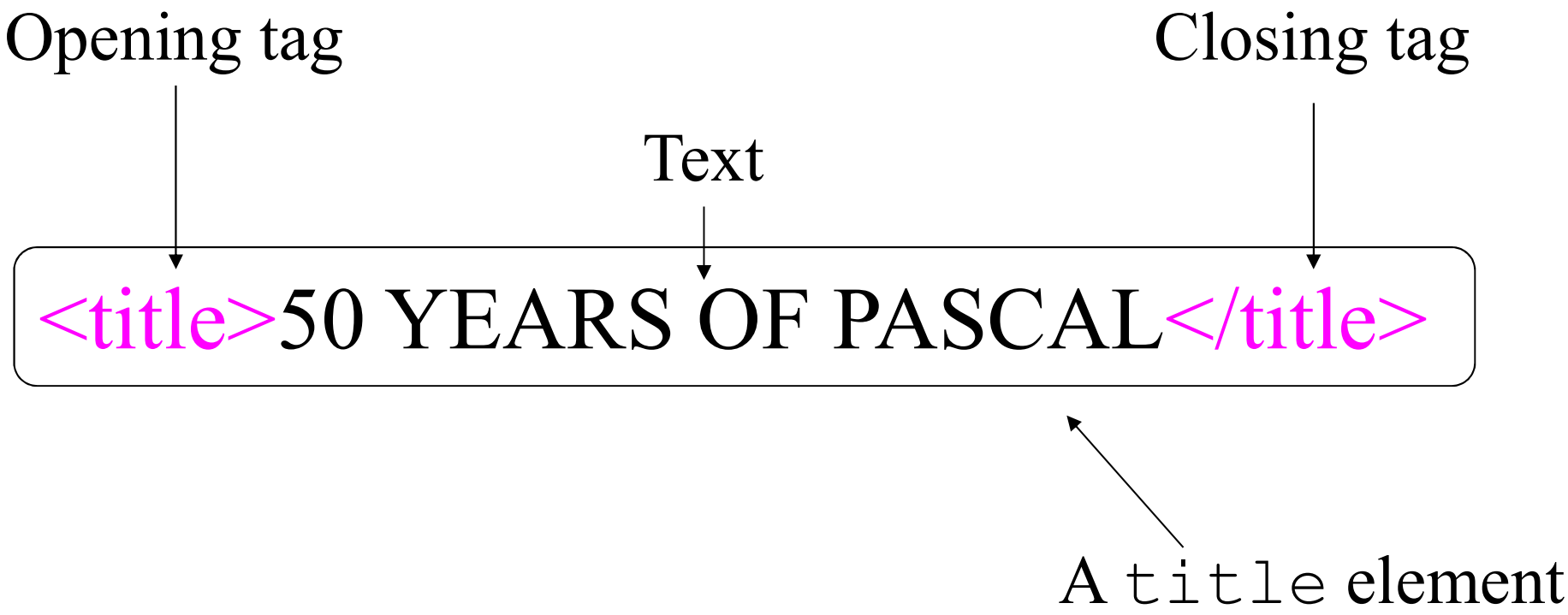
```
print(html)
```

A Glance at an HTML File

```
<HTML>
<HEAD>
<TITLE>50 YEARS OF PASCAL</TITLE>
</HEAD>

<BODY>
<H1>50 Years of Pascal</H1>

<DIV>By Niklaus Wirth
<BR> Communications of the ACM, March 2021, Vol. 64 No. 3, Pages 39-41
<BR><A HREF='https://doi.org/10.1145/3447525'>DOI:10.1145/3447525</A>
</DIV>

<P>In the early 1960s …
```

HTML Tags

- HTML tags normally come in pairs like <title> and </title>.



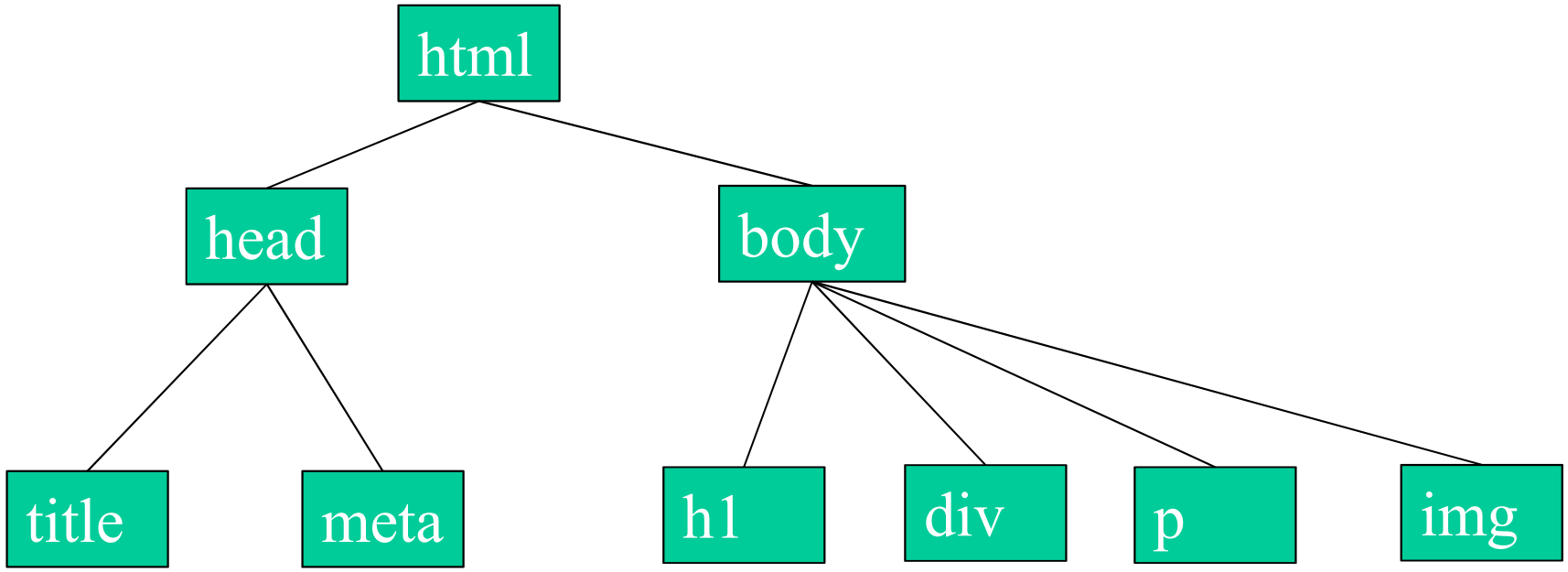
Tags

- title - page title
- h1 - level 1 heading
- h2 - level 2 heading
- a - defines a hyperlink
  - attributes: href
- img - image
  - attributes: src, alt, width, height
- ol - ordered list
- ul - unordered list
- li - list item
- div - defines a division so that it can be easily styled using the class or id attribute in CSS.
- span - defines an inline container to be easily styled.

Example: page3.html

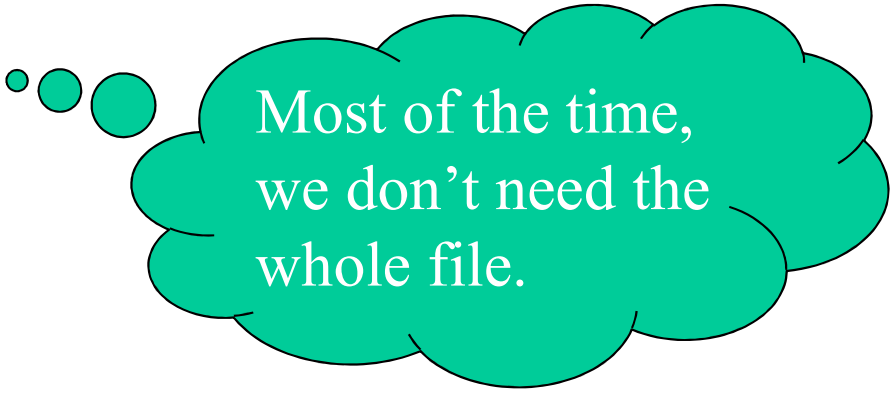
```
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Page 3</title>
   <style> .blue { color:blue; }
      .red { color:magenta; }
      h2 { color:green; }
   </style>
   </head>
   <body>
      <h1>Good morning</h1>
      <ol>
         <li>Monday</li>
         <li>Tuesday</li>
         <li>Wednesday</li>
      </ol>
      <h2>A few links</h2>
      <ul>
      <li> <a href='http://cnn.com/'>CNN</a></li>
      <li><a href='http://bbc.com/'>BBC</a></li>
      </ul>
<img src='http://poet.ncnu.org/run_girls.png' alt="Run! Girls" width='80%'>

This is a sentence to demonstrate the span tag and the div tag.
A <span class="red">span</span> is an <span class="red">inline</span> container, while a <div class="blue">div</div> is an independent <div class="blue">division</div>.

</body></html>
```

Structure of an HTML File



Structure of an HTML File

```
<HTML>
<HEAD>
<TITLE>50 YEARS OF PASCAL</TITLE>
</HEAD>

<BODY>
<H1>50 Years of Pascal</H1>

<DIV>By Niklaus Wirth
<BR> Communications of the ACM, March 2021, Vol. 64 No. 3, Pages 39-41
<BR><A HREF='https://doi.org/10.1145/3447525'>DOI:10.1145/3447525</A>
</DIV>

<P>In the early 1960s …
```



Parse an HTML file with BeautifulSoup

```
from urllib.request import urlopen
from bs4 import BeautifulSoup

url = "http://poet.ncnu.org/page1.html"
html = urlopen(url).read().decode()
bsObj = BeautifulSoup(html, "html.parser")
print(bsObj.h1)
```

```
<h1>50 Years of Pascal</h1>
```

Exercise: Modify this code to display the title of this webpage.

Structure of a BeautifulSoup object

- html→<html><head>…</head><body>…</body></html>
  - head→<head><title>50 YEARS OF PASCAL</title></head>
    - ☞ title→<title>50 YEARS OF PASCAL</title>
  - body→<body><h1>50 Years of Pascal</h1> By Niklaus Wirth … </body>
    - ☞ h1→<h1>50 Years of Pascal</h1>
    - ☞ div→ By Niklaus Wirth …

You can conveniently access the h1 tag by
`bsObj.h1`
or by any of the following
`bsObj.html.body.h1`
`bsObj.body.h1`
`bsObj.html.h1`

bsObj.tagname

- `bsObj.tagname` which we learned previously only returns the first tag found in this webpage.
- Consider http://poet.ncnu.org/page2.html, what will `bsObj.div` return?

```
<html>
<head>
<title>This is a page</title>
</head>

<body>
<h1>First h1</h1>
<div>Division 1</div>

<h1>Second h1</h1>
<h2> Header 2</h2>
<div>Division 2</div>
</body>
</html>
```

Exercise: Write a Python program to show bsObj.findAll("div").
Exercise: Modify the previous exercise to show only the second tag.

1. findAll()

- `findAll(tag, attributes, text, limit, keywords)`

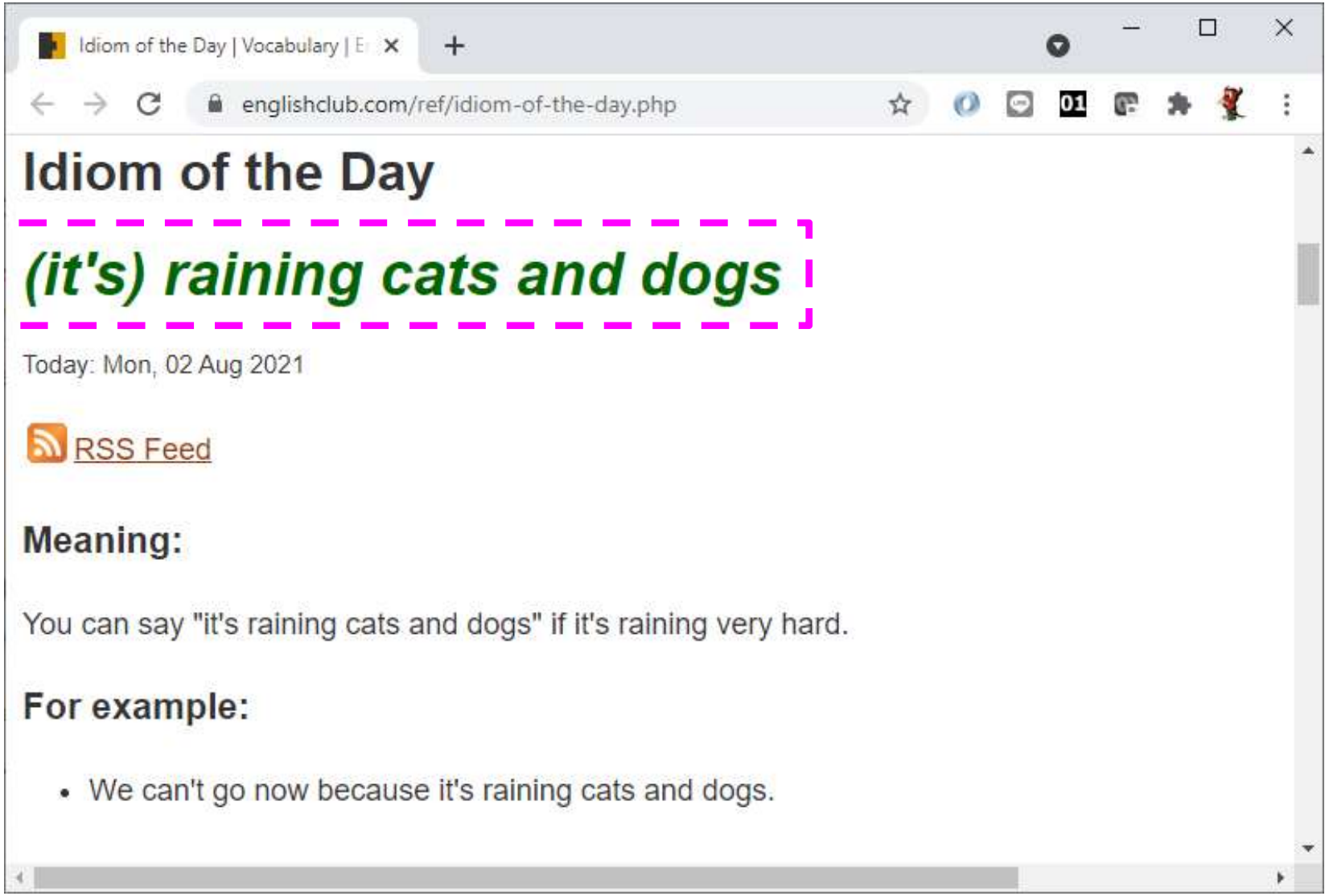
- All header tags
  - .findAll(["h1", "h2" , "h3", "h4", "h5", "h6"])
- With attributes class="green" or class="red"
  - .findAll("span", {"class": ["green", "red"]})
- Text content of the tags
  - nameList = bsObj.findAll(text="the prince")
  - print(len(nameList))

The output is "7".

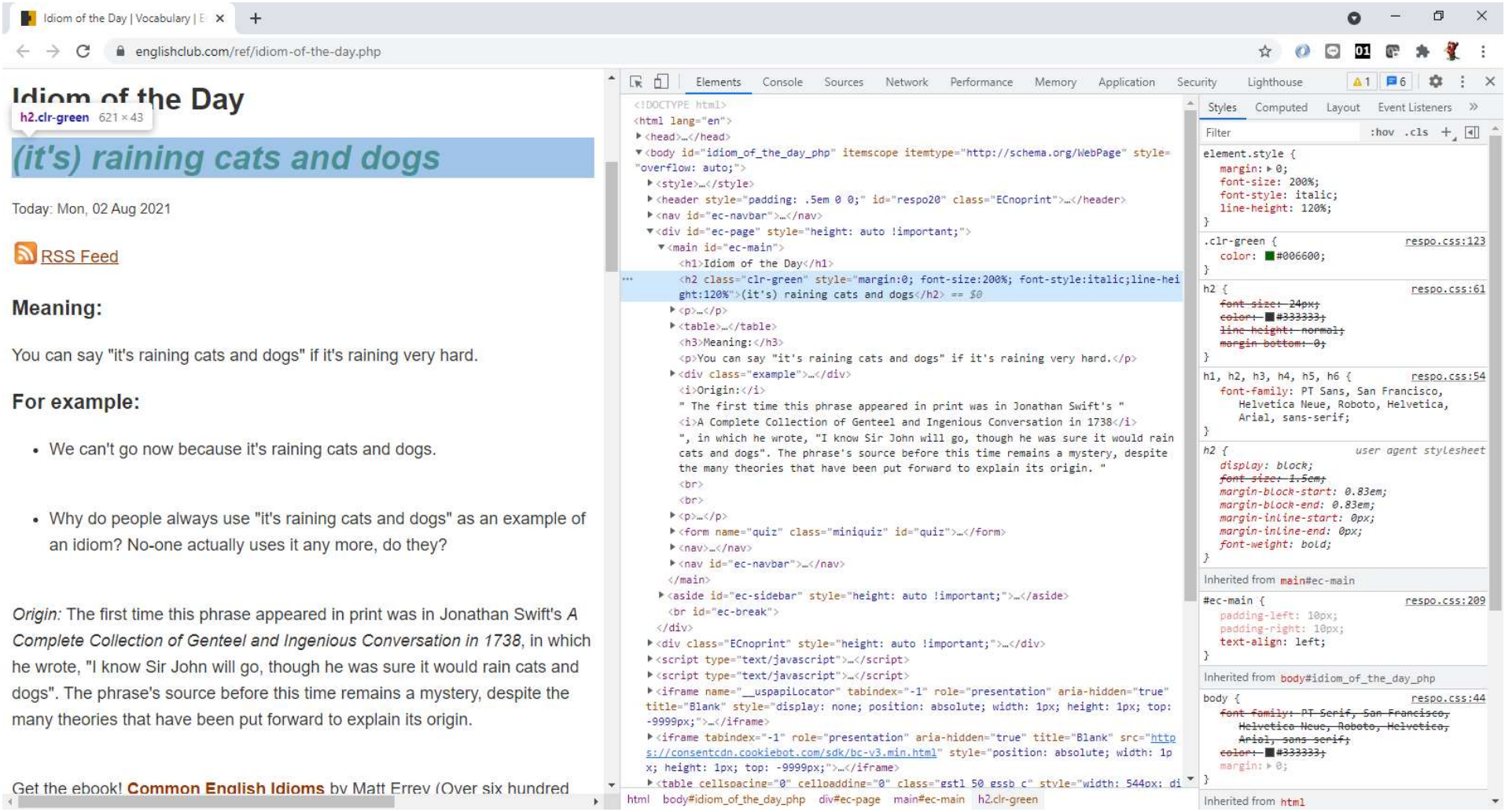
- Limit of the returned list, when you are only interested in the first x items.
  - aList = bsObj.findAll("a", limit=x)
- Keywords:
  - bsObj.findAll(id='link2')

You may also use
`aList = bsObj.findAll("a")`
and extract the first 50 items by
`aList[0:50]`
But …

Idiom of the Day

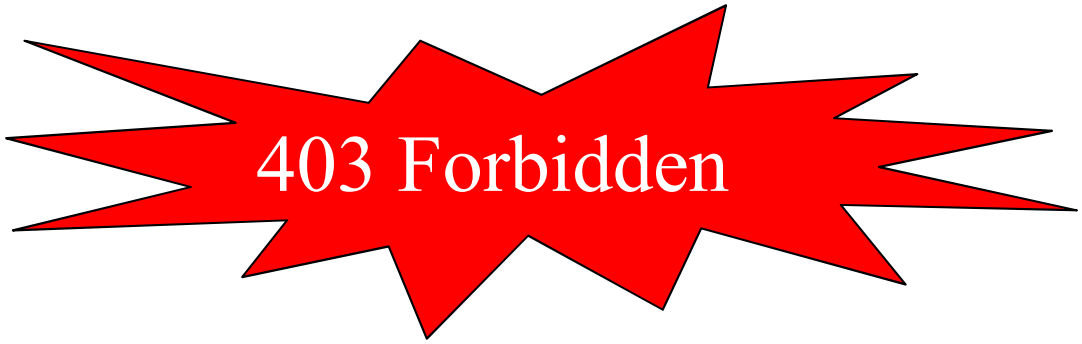


Google Chrome Developer Tools <F12>



My Project Is Defied!

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
```




```
url = 'https://www.englishclub.com/ref/idiom-of-the-day.php'
html = urlopen(url).read().decode()
bsObj = BeautifulSoup(html, "html.parser")
idiom = bsObj.find("h2", {"class": "clr-green"}).get_text()
print( idiom )
```

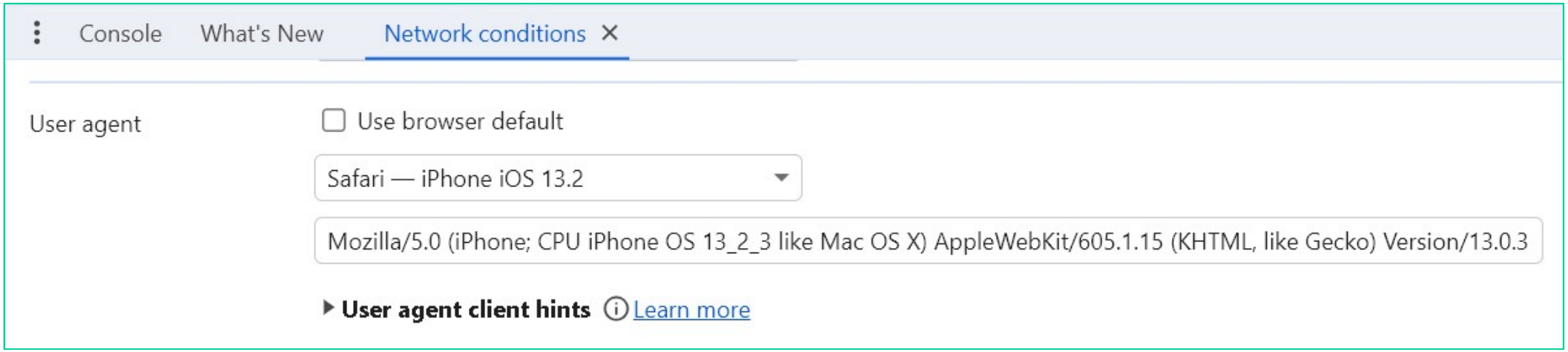
Tell the server I am not a crawler!

```
from urllib.request import urlopen, Request
from bs4 import BeautifulSoup
url = 'https://www.englishclub.com/ref/idiom-of-the-day.php'
req = Request(url, headers={ 'User-Agent': 'Mozilla/5.0
(Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/35.0.1916.47    Safari/537.36' } )
html = urlopen(req).read().decode()
bsObj = BeautifulSoup(html, "html.parser")
idiom = bsObj.find("h2", {"class": "clr-green"}).get_text()
print(idiom)
```

Chrome User-Agent Switcher

■ <F12> Developer tools

- Three dots beside the close button - More tools - Network conditions
  - ☞Choose the “User agent” you like.



Gradually Refine the Constraints to Filter Out What You Want

- bsObj.findAll("h2")
  - [<h2 class="clr-green" style="margin:0; font-size:200%; font-style:italic;line-height:120%">(it's) raining cats and dogs</h2>, <h2>Quick Quiz</h2>, <h2>b</h2>]
- bsObj.find("h2", {"class": "clr-green"})
  - <h2 class="clr-green" style="margin:0; font-size:200%; font-style:italic;line-height:120%">(it's) raining cats and dogs</h2>
- bsObj.find("h2", {"class": "clr-green"}).get_text()
  - (it's) raining cats and dogs

魔鏡歌詞網

- 白露 (王梓鈺)
  - https://mojim.com/twy217658x35x1.htm
- 小滿 (王梓鈺)
  - https://mojim.com/twy217658x36x1.htm
  - https://www.youtube.com/watch?v=RGtSdbjxVKU
- 小滿 (劉德華)
  - https://www.youtube.com/watch?v=SEmJfO7hhfs

小滿 (王梓鈺)

```
url = 'https://mojim.com/twy217658x36x1.htm'
from urllib.request import urlopen
from bs4 import BeautifulSoup
html = urlopen(url).read().decode()
bsObj = BeautifulSoup(html)
a = bsObj.find("div", {"id": "fsZ"} )
a.get_text().replace('更多更詳盡歌詞 在 ※ Mojim.com\u3000魔鏡歌詞網', '')
```

You should let your program remove this automatically, instead of doing that manually by yourself.

換行？

- 檢視第4行得到的 html 變數，在原本的 HTML 文件中，是用
來進行換行。
- BeautifulSoup物件在get_text()時把這些 HTML的 tag 都移除了，所以「換行」的效果，字型顏色的變化，也都被移除了。

```
html1 = html.replace('<br />', '\n')
bsObj1 = BeautifulSoup(html1)
a1 = bsObj1.find("div", {"id": "fsZ"} )
print( a1.get_text() )
```

Homework: 慶典 (2024)

- https://mojim.com/慶+方文山.html?t4

- urlopen() does not accept a URL containing Chinese text.
- You may invoke quote() to prepare the URL for you.
  - from urllib.parse import quote
  - url = 'https://mojim.com/' + quote(keyword) + '.html?t4'

- Write a program to list the URLs of those 16 lyrics which contains "慶+方文山".
  - 左邊的 URL 是「專輯介紹」，右邊的才是歌詞。

attrs

- url = 'https://mojim.com/%E6%85%B6%2B%E6%96%B9%E6%96%87%E5%B1%B1.html?t4'
- from urllib.request import urlopen
- html = urlopen(url).read().decode()
- from bs4 import BeautifulSoup
- bsObj = BeautifulSoup(html)
- table = bsObj.find("table", { "class" : "iB" })
- links = table.findAll('a')
- links[0]
  - 4.離開的那一些
- links[16:18]
  - [1.詩水蛇山神廟,
  - 蛇山神廟作詞方文山作曲周杰倫編曲 Kenn C北方的馬蹄瀰漫著雪白的過去這整遍銀白色的大地凝 ... 定神秘地愛你廣場熱鬧慶典很華麗你繞我跳圓舞曲我確定跟你的默契＊@扭腰擺手精準的比例我們完美跳到底我]
- print(links[0].attrs["href"], links[0].attrs["title"])

A Few Words

- Read the Terms of Service and Privacy Policies of a website before scraping it (this might not be possible in many situations though).
- If it's not clear from looking at the website, contact the webmaster and ask if and what you're allowed to harvest.
- Be gentle on smaller websites.
  - Run your scraper in off-peak hours
  - Space out your requests.
    - ☞ For example, pause 5 seconds between two requests.
- If the website has a public API that provides the data you're looking for, use it and avoid scraping all together.
  - 立法院開放資料平台 https://data.ly.gov.tw/getds.action?id=9
  - 高雄市政府資料開放 https://data.kcg.gov.tw/

There are some webpages …

- Requires login via username/password.

- Reject web crawlers via verification code.

- Display secret messages after a special action
  - page4.html

Summary

- Many online services are provided via the web. With the skill of web scraping, you can write a program to quickly retrieving a large amount of data.
  - You may even submit many data automatically.
- Many web servers do not welcome web crawlers.
  - Don't send many requests in a short period of time.
  - Let your crawler tell the server "I am not a crawler. I am a Chrome/Firefox/Safari browser."
- Some webpages heavily utilize JavaScript to prevent their data. In today's lesson we only learn how to handle HTML webpages.
  - To further handle JavaScript, you'll need a more advanced lesson to learn a new tool - Selenium.