

**Fastcampus**

**Computer Science SCHOOL**

**Network Basic (2)**

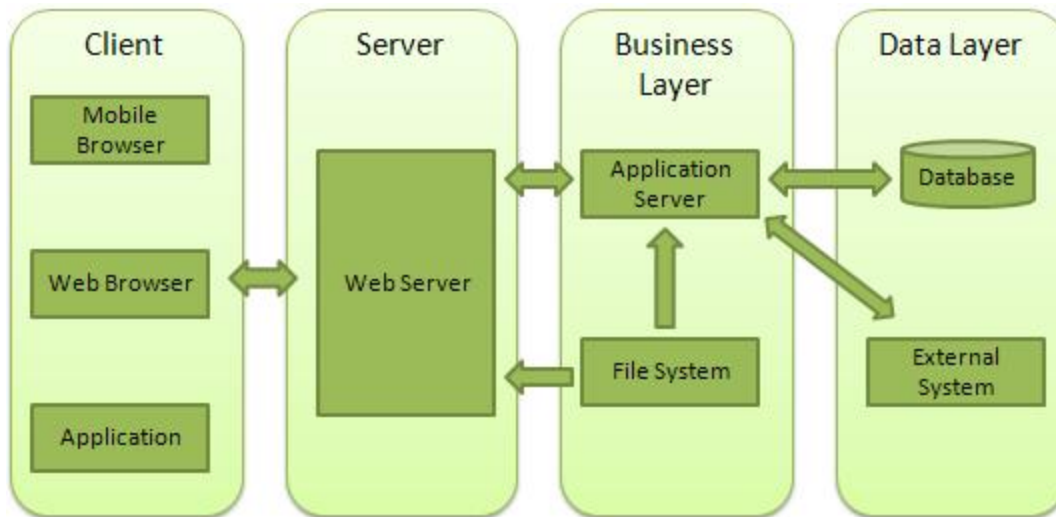
2017.2.15

**A/S**

python urlopen pycharm https problem

# Web Programming

# Web architecture

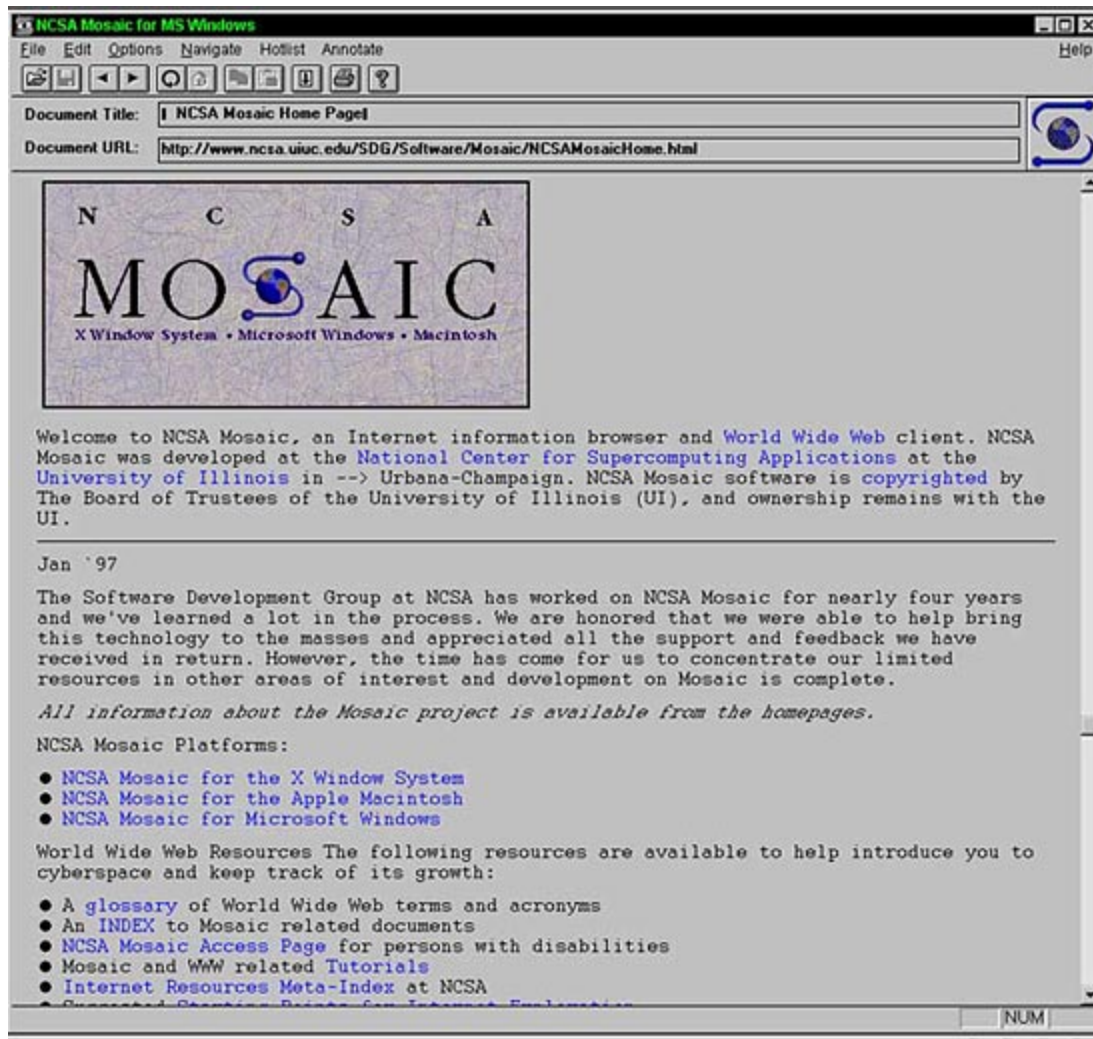


## 웹 개발 패턴의 변화

- 1991 ~ 1999: Sir Timothy John "Tim" Berners-Lee가 하이퍼텍스트 기반의 프로젝트를 제안한 이후 정적인 컨텐츠들을 중심으로 한 웹 기술이 발달
- 1999 ~ 2009: Linux, Apache, Mysql, Php 중심의 동적인 서버, 정적인 클라이언트 모델이 지속됨
- 2010 ~ 현재: JavaScript!! (Dynamic Web Client)

# Web Browser

# Mosaic(1993)

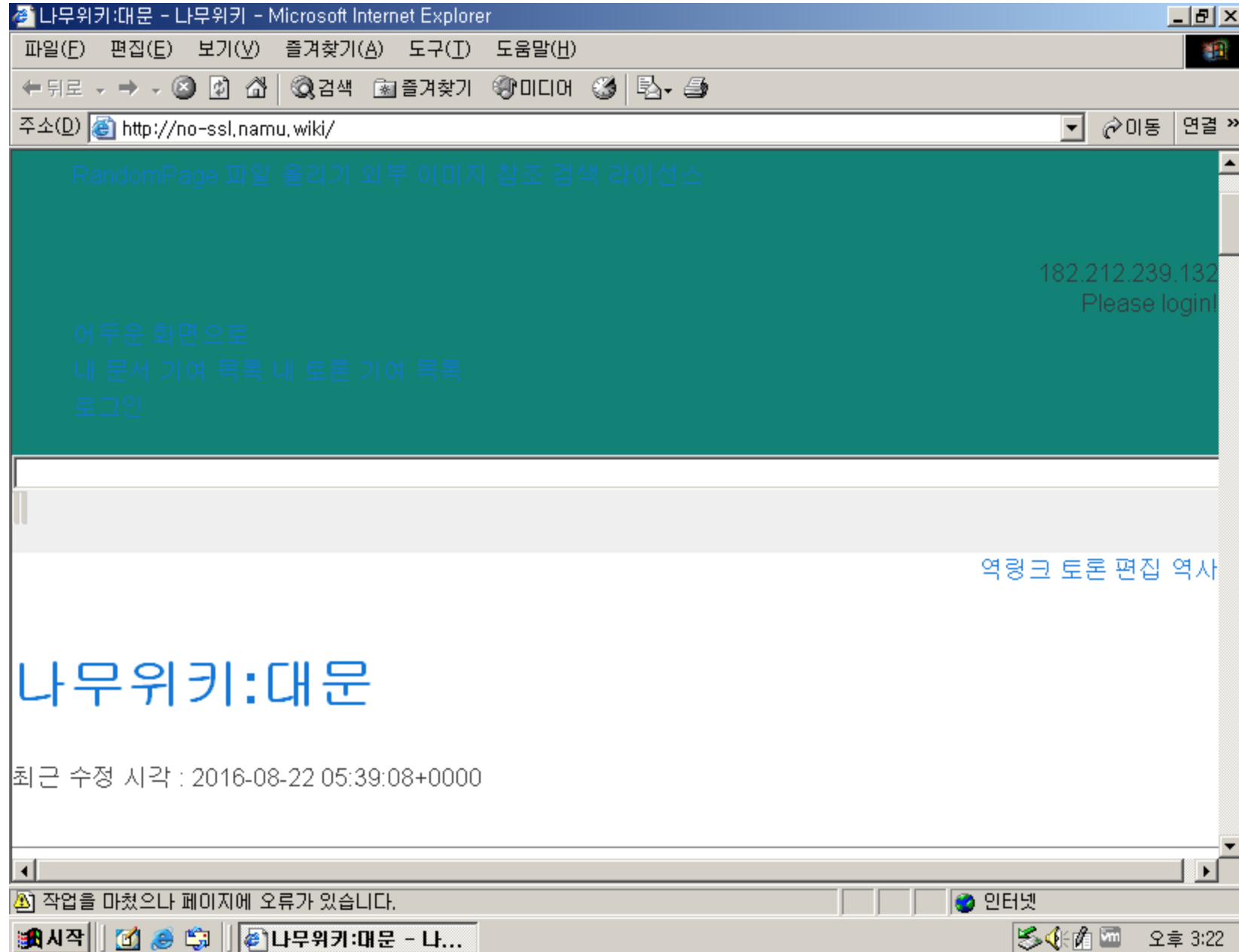


# Netscape(1994)





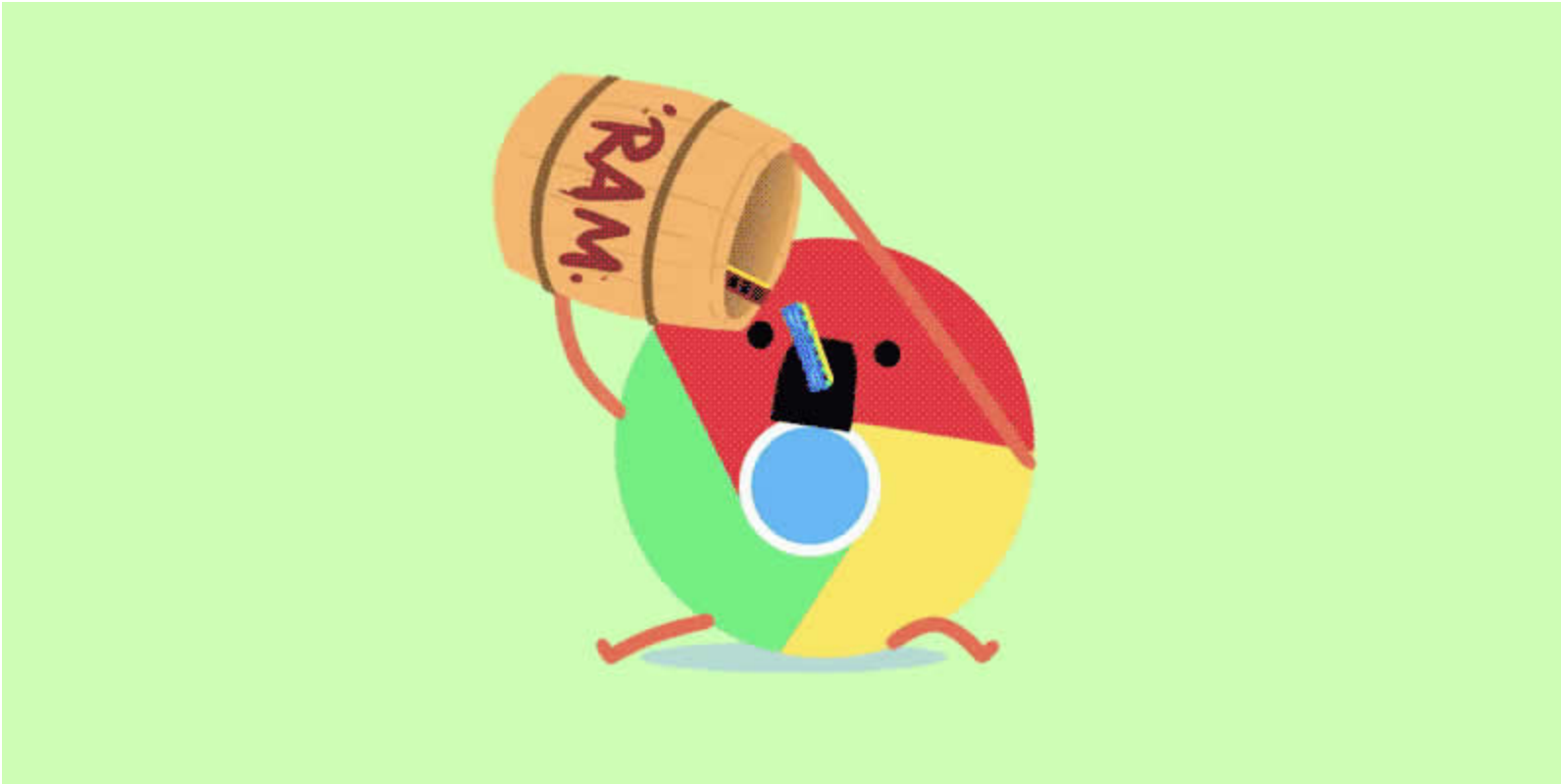
# Internet Explorer(1995)



# FireFox(2004)



# Chrome(2008)



**웹 개발의 현재**

**JavaScript**

# Client-side

- HTML/CSS, javaScript
- jQuery, AJAX
- Front-end Web Framework
  - AngularJS 2
  - React.js
  - Vue.js
- CSS Framework
  - Bootstrap
  - Foundation

# Server-side

- Depends on Language
  - PHP: Laravel
  - JavaScript: Node.js(Express.js)
  - Java: Spring
  - C++, C#: [ASP.net](#)
  - Python: Django, Flask
  - Golang: itself
  - Ruby: Ruby on Rails

# Database

- RDBMS
  - MySQL
  - PostgreSQL
  - MariaDB
- noSQL
  - MongoDB
  - CouchDB
  - Redis

**etc**

- celery (for Distributed Task Queue)
- github, Bitbucket, gitlab (for SCM)
- travis CI or jenkins (for Continuous Integration)
- slack, trello



# URI, URL, URN

## URI

- Uniform Resource Information
- `https://www.example.com/post/how-to-make-url`

## URL

- Uniform Resource Locator
- `https://www.example.com/post/`

## URN

- Uniform Resource Name
- `www.example.com/post/how-to-make-url`

# REST API

RE presentational S tate T ransfer

A pplication P rogramming I nterface

Resource - URI

Verb - HTTP method

Representations - 표현

# **CRUD**

**Create**

**Read**

**Update**

**Delete**

## REST API 설계시 주의할 점

- 버전관리 <https://api.foo.com/v1/bar>
- 명사형 사용 <https://foo.com/showid/> --> <https://foo.com/user/>
- 반응형 <https://foo.com/m/user/>, <https://m.foo.com/user/> (x)
- 언어코드 <https://foo.com/kr/>, <https://kr.foo.com/> (x)
- 응답상태 코드 (200, 400, 500)

# HTTP Response code

[wikipedia](#)

200, 201 - Success

400, 404 - Not found

500 - Server error

**BUT IT'S NOT  
"RESTful" IF YOU...**



**ENOUGH!**

# Simple Server Framework: Flask

```
$ pip install flask
```

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    return 'hello world!'

if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

# Simple Server Framework: Flask

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index(name=None):
    return render_template('index.html', name=name)

@app.route('/about')
def index(name=None):
    return render_template('about.html', name=name)

if __name__ == '__main__':
    app.run(host='0.0.0.0')
```



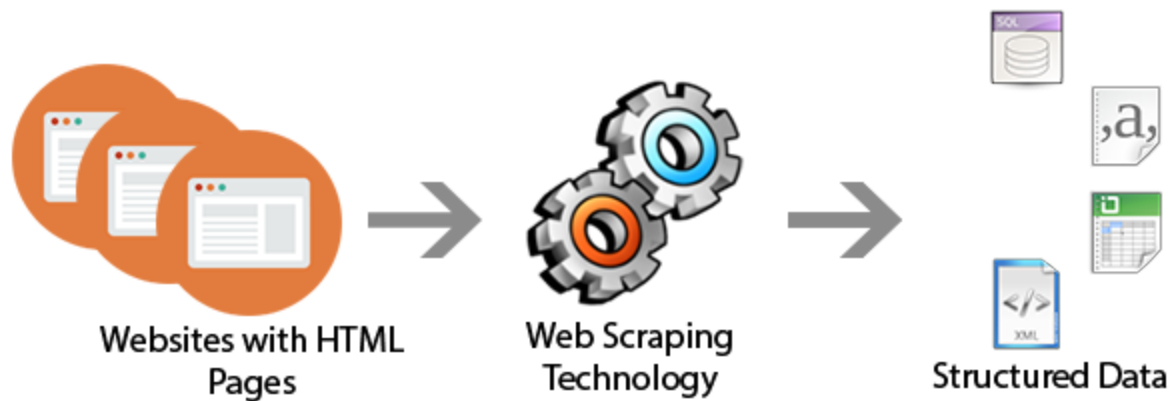
# Simple Server Framework: Flask

```
/
  simpleserver.py
  /templates
    index.html
    about.html
```

# Web Crawling with Python

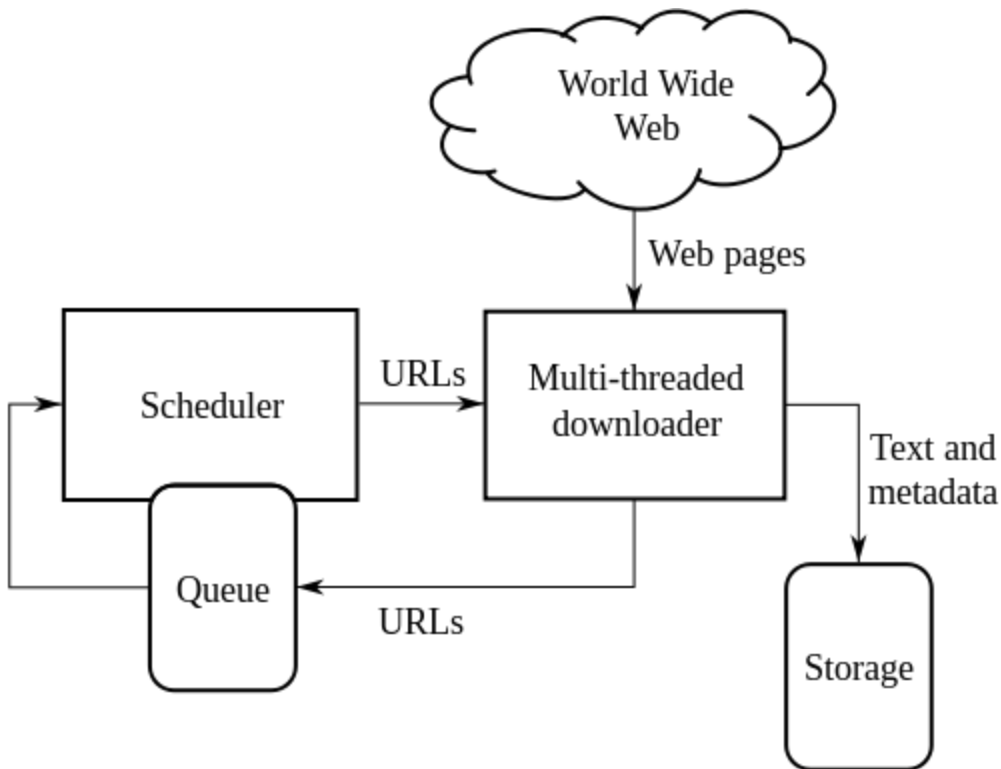
# Scraping vs Crawling vs Parsing

Scraping: 데이터를 수집하는 행위



# Scraping vs Crawling vs Parsing

Crawling: 조직적 자동화된 방법으로 월드 와이드 웹을 탐색하는 것



# Scraping vs Crawling vs Parsing

Parsing: 문장 혹은 문서를 구성 성분으로 분해하고 위계관계를 분석하여 문장의 구조를 결정하는 것



# Beautiful Soup

# Web Scraping with BeautifulSoup

```
$ pip install beautifulsoup4
$ pip install beautifulsoup4
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.5.1-py3-none-any.whl (83kB)
    100% |██████████████████████████████████████████████████████████████████████████████| 92kB 153kB/s
Installing collected packages: beautifulsoup4
Successfully installed beautifulsoup4-4.5.1
```

# Web Scraping with BeautifulSoup

```
$ pip list
DEPRECATION: The default format will switch to
the [list] section) to disable this warning.
beautifulsoup4 (4.5.1)
pip (9.0.1)
setuptools (20.10.1)
urllib3 (1.19.1)
```



# Web Scraping with BeautifulSoup

```
>>> import urllib
>>> from bs4 import BeautifulSoup
>>> html = """
... <html><head><title>The Dormouse's story</title></head>
... <body>
... <p class="title"><b>The Dormouse's story</b></p>
...
... <p class="story">Once upon a time there were three little sisters; and their names were
... <a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
... <a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
... <a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
... and they lived at the bottom of a well.</p>
...
... <p class="story">...</p>
... """
>>> soup = BeautifulSoup(html, 'html.parser')
>>> print(soup.prettify())
```

# Web Scraping with BeautifulSoup

```
import urllib
from bs4 import BeautifulSoup
html = """

uglified html code

"""
soup = BeautifulSoup(html, "html.parser")
print(soup.prettify())
```

# Web Scraping with BeautifulSoup

```
curl https://www.rottentomatoes.com
```

```
import urllib.request
from bs4 import BeautifulSoup

url = "https://www.rottentomatoes.com"
html = urllib.request.urlopen(url)
source = html.read()
html.close()

soup = BeautifulSoup(source, "html.parser")
print(soup)

table = soup.find(id="Top-Box-Office")
print(table)
```

# Web Scraping with BeautifulSoup

```
all_tr = table.find_all("tr")

for tr in all_tr:
    all_td = tr.find_all("td")
    score = all_td[0].find("span", attrs={"class": "tMeter"})
    movie_name = all_td[1].a.text
    amount = all_td[2].a.text
    print(score, movie_name, amount)
```

# Web Scraping with BeautifulSoup

```
>>> import urllib.request
>>> from bs4 import BeautifulSoup
>>> url = "https://www.rottentomatoes.com"
>>> html = urllib.request.urlopen(url)
>>> source = html.read()
>>> html.close()
>>> soup = BeautifulSoup(source, "html.parser")
>>> table = soup.find(id="Top-Box-Office")
>>> all_tr = table.find_all("tr")
>>> for tr in all_tr:
...     all_td = tr.find_all("td")
...     score = all_td[0].find("span", attrs={"class": "tMeterScore"}).text
...     movie_name = all_td[1].a.text
...     amount = all_td[2].a.text
...     print(score, movie_name, amount)
```

...	
69% Sing	\$41.5M
95% Fences	\$10.2M
40% Why Him?	\$10.1M
16% Assassin's Creed	\$8.1M
12% Collateral Beauty	\$4.1M
73% Fantastic Beasts and Where to Find Them	\$4.0M

**So, Let's Scrap Naver**