Web Request

Http Request Module

- urllib
  - » 설치

```
built in module
```

  - » 사용 사례

```
from urllib import request
d = request.urlopen('http://www.daum.net')
d.read().decode('utf-8')
```

- requests
  - » 설치

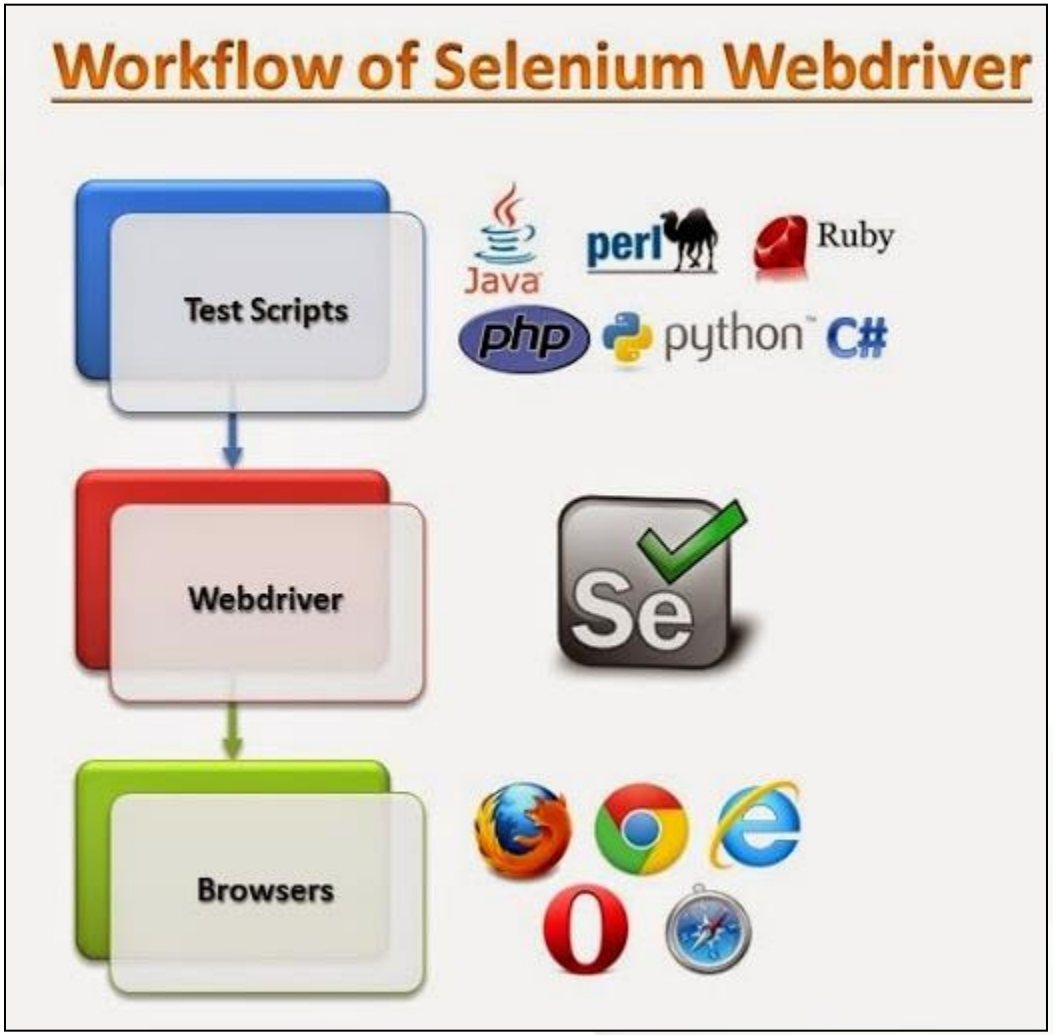
```
conda install requests (or pip install requests)
```

  - » 사용 사례

```
import requests
r = requests.get('http://www.naver.com')
r.status_code
r.text
```

selenium

- 브라우저 기반 테스트 자동화 도구
- 애플리케이션 코드를 통해 링크 클릭, 버튼 클릭 등 브라우저에서의 사용자 동작을 자동화 할 수 있는 도구
- 로컬 컴퓨터의 웹 브라우저를 제어하기 위해 selenium 클라이언트 사용
- selenium 클라이언트는 WebDriver 라는 공통 인터페이스와 이를 구현한 브라우저별 드라이버로 구성



selenium

- 파이썬 클라이언트 모듈 설치

```
conda install selenium ( or pip install selenium )
```

- 브라우저별 드라이버 설치
  - » Firefox

    https://github.com/mozilla/geckodriver/releases

  - » Chrome

    https://sites.google.com/a/chromium.org/chromedriver/downloads

  - » Edge

    https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/

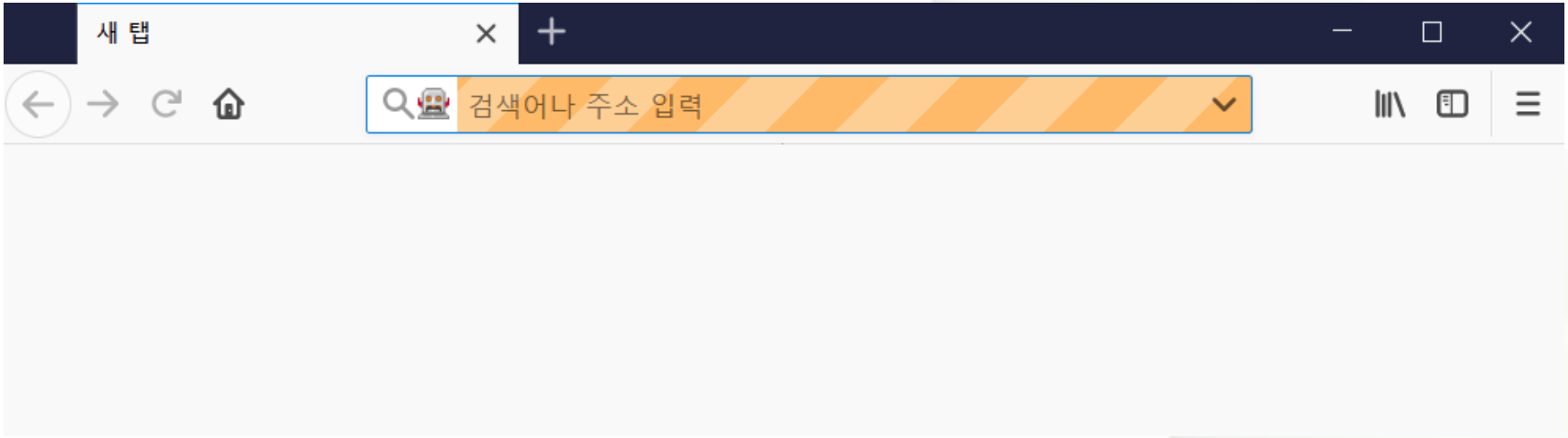
Quick Start

- 드라이버 모듈 임포트

```
from selenium import webdriver
```

- 브라우저 연결 및 실행

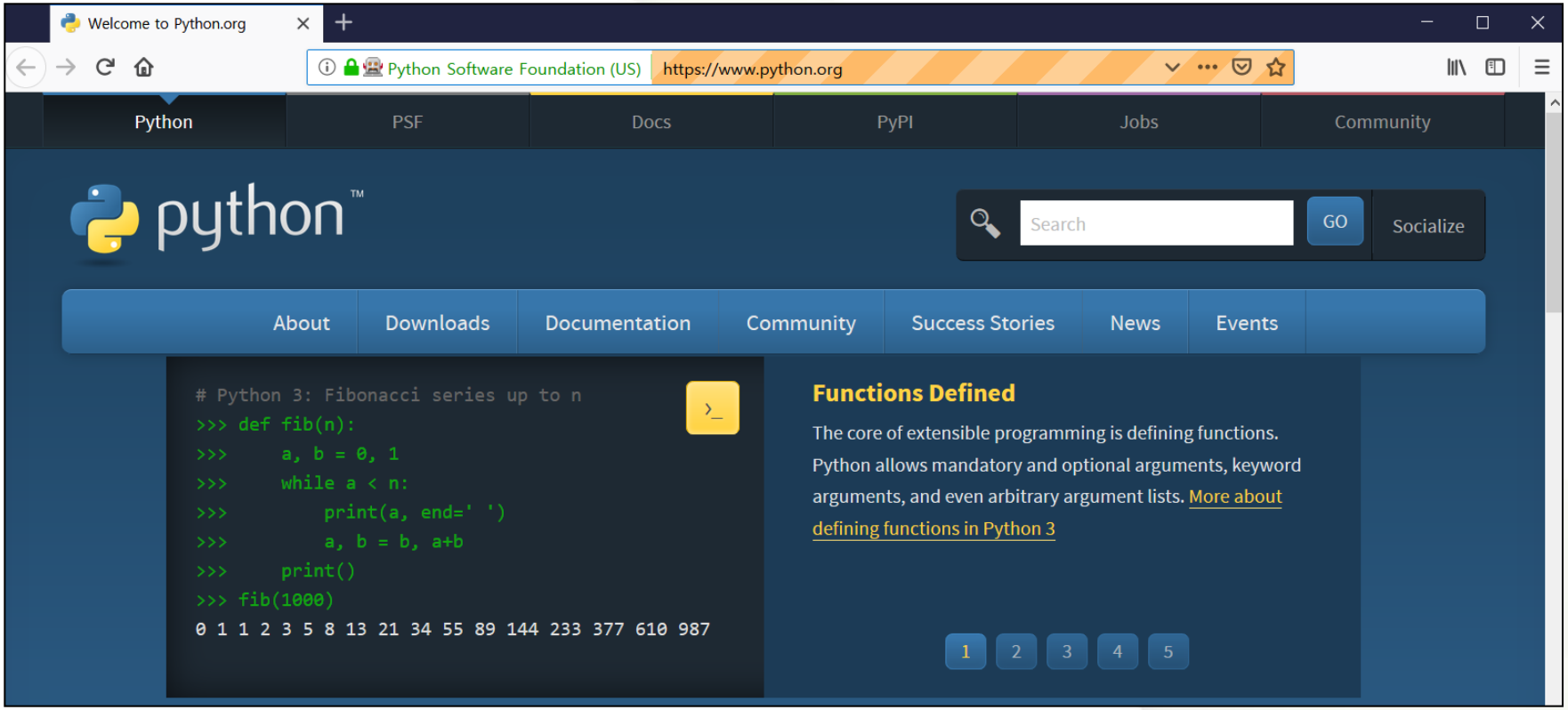
```
browser = \
webdriver.Firefox(executable_path=r"path-for-geckodriver-executable")
#browser = \
webdriver.Chrome(executable_path=r "path-for-chromedriver-executable")
```



Quick Start

- 페이지 이동

```
browser.get("http://python.org")
```



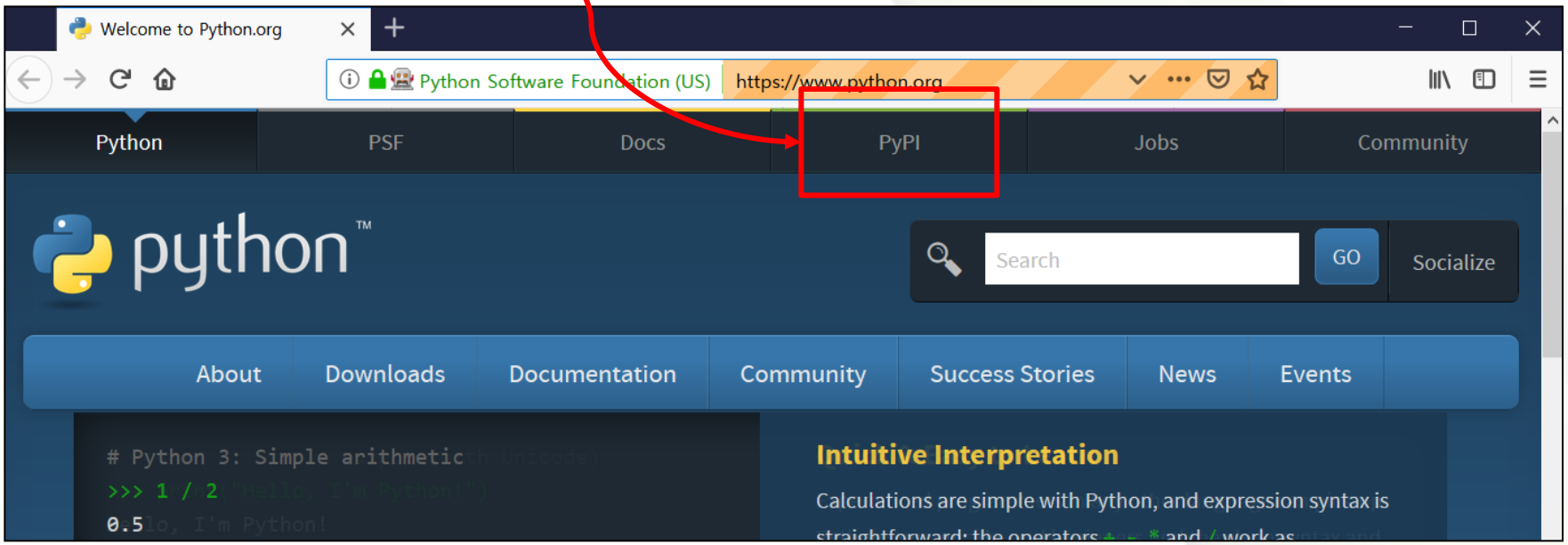
Quick Start

- PyPI 메뉴 요소 검색

```
menus = browser.find_elements_by_css_selector('#top ul.menu li’)

pypi = None
for m in menus:
    if m.text == "PyPI":
        pypi = m
    print(m.text)
```

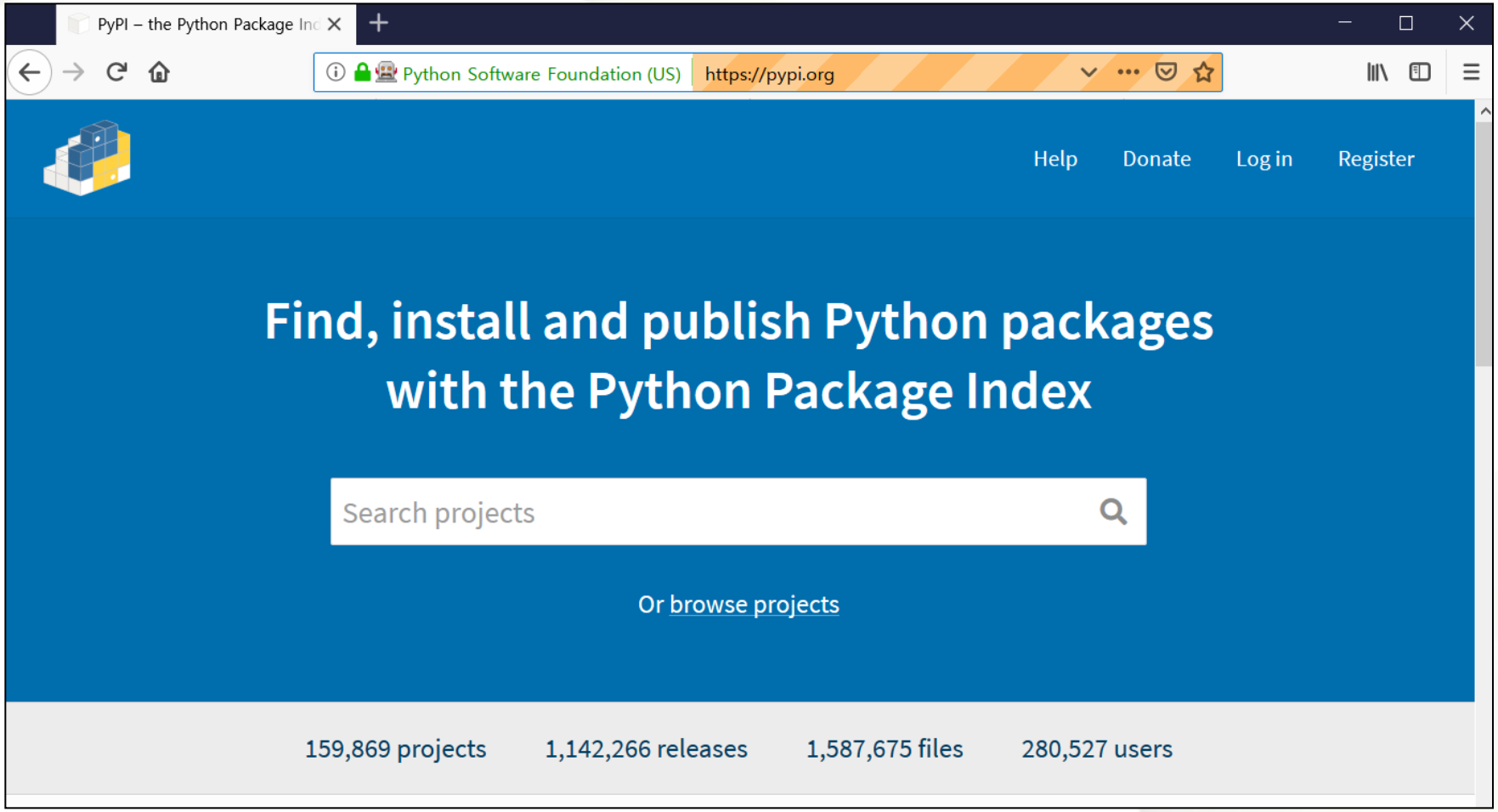
```
Python
PSF
Docs
PyPI
Jobs
Community
```



Quick Start

- PyPI 메뉴 클릭

```
pypi.click()  # 클릭
```



Quick Start

- 브라우저 종료

```
import time

time.sleep(5) # 5초 대기
browser.quit() # 브라우저 종료
```

Beautiful Soup (HTML Parser)

Beautiful Soup

- HTML 또는 XML로부터 데이터를 읽을 때 사용하는 파이썬 라이브러리

- document

```
https://www.crummy.com/software/BeautifulSoup/bs4/doc/
```

- 설치

```
pip install beautifulsoup4
```

or

```
pip install bs4
```

- 사용

```
from bs4 import BeautifulSoup

soup = BeautifulSoup(html_doc, 'html.parser')
```

BeautifulSoup Parser

- BeautifulSoup이 html 문서를 분석할 때 사용하는 도구
  - » 기본적으로 파이썬 내장 파서 지원
  - » 추가로 lxml, html5lib 등의 써드파티 파서 지원

- 써드파티 파서 설치

```
pip install lxml (or conda install lxml)
```

```
pip install html5lib (or conda install html5lib)
```

Beautiful Soup Parser 비교

파서	사용 방법
Python’s html.parser	BeautifulSoup(markup, "html.parser")
lxml’s HTML parser	BeautifulSoup(markup, "lxml")
lxml’s XML parser	BeautifulSoup(markup, "lxml-xml") BeautifulSoup(markup, "xml")
html5lib	BeautifulSoup(markup, "html5lib")

파서	장점	단점
Python’s html.parser	Decent speed Lenient (as of Python 2.7.3 and 3.2.)	Not very lenient (before Python 2.7.3 or 3.2.2)
lxml’s HTML parser	Very fast Lenient	External C dependency
lxml’s XML parser	Very fast The only currently supported XML parser	External C dependency
html5lib	Extremely lenient Parses pages like a web browser does Creates valid HTML5	Very slow External Python dependency

Quick Start

- import library

```
from bs4 import BeautifulSoup
```

- html 문서 데이터 만들기

```
html_doc = """<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>"""
```

Quick Start

- BeautifulSoup 객체 만들기

```
soup = BeautifulSoup(html_doc, 'html.parser')
```

- 문서 내용 출력

```
soup.prettify()
```

Quick Start

- 문서 탐색

```
soup.title
# <title>The Dormouse's story</title>

soup.title.name
# u'title'

soup.title.string
# u'The Dormouse's story'

soup.title.parent.name
# u'head'

soup.p
# <p class="title"><b>The Dormouse's story</b></p>

soup.p['class']
# u'title'

soup.a
# <a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>
```

Quick Start

- 문서 탐색 (계속)

```
soup.find_all('a')
# [<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
#  <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]

soup.find(id="link3")
# <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>

for link in soup.find_all('a'):
    print(link.get('href'))
# http://example.com/elsie
# http://example.com/lacie
# http://example.com/tillie
```

Quick Start

- 문서 탐색 (계속)

```
print(soup.get_text())
# The Dormouse's story
#
# The Dormouse's story
#
# Once upon a time there were three little sisters; and their names were
# Elsie,
# Lacie and
# Tillie;
# and they lived at the bottom of a well.
#
# ...
```

객체

- BeautifulSoup은 HTML 문서를 파이선 객체의 트리로 변환

- 객체 종류

객체	설명
Tag	XML 및 HTML의 tag에 해당하는 객체
BeautifulSoup	전체 문서에 해당하는 객체
NavigableString	tag에 포함된 (부분)문자열 객체
Comments	주석에 해당하는 객체
기타 XML 문서 요소	CData, ProcessingInstruction, Declaration, Doctype 등의 객체 사용 NavigableString 상속

주요 속성 및 메서드

- 하위 요소 탐색

속성 및 메서드	설명
tag name	태그 이름으로 문서 요소 탐색 사례 : soup.head, soup.title, soup.body.b, …
.contents	모든 자식 요소를 리스트 형식으로 반환
.children	모든 자식 요소를 순회 가능한 객체로 반환
.descendants	모든 후손 요소를 리스트 형식으로 반환
.string	유일한 NavigableString 형식의 자식 또는 후손 요소 반환
.strings	모든 NavigableString 형식의 요소 반환
.stripped_strings	.strings의 반환 값에서 앞뒤의 \n, \t, space 등 제거

주요 속성 및 메서드

- 상위 요소 탐색

속성 및 메서드	설명
.parent	부모 요소 반환
.parents	조상(Ancestor) 요소를 리스트 형식으로 반환

- 동일 수준 요소 탐색

속성 및 메서드	설명
.next_sibling	같은 부모를 가진 요소 중 한 개의 다음 요소 반환
.previous_sibling	같은 부모를 가진 요소 중 한 개의 이전 요소 반환
.next_siblings	같은 부모를 가진 요소 중 모든 다음 요소 반환 (반복자)
.previous_siblings	같은 부모를 가진 요소 중 모든 이전 요소 반환 (반복자)

주요 속성 및 메서드

- 탐색 요소간 이동

속성 및 메서드	설명
.next_element	파서의 탐색 순서상 한 개의 다음 요소 반환
.previous_element	파서의 탐색 순서상 한 개의 이전 요소 반환
.next_elements	파서의 탐색 순서상 모든 다음 요소 반환 (반복자 형식)
.previous_elements	파서의 탐색 순서상 모든 이전 요소 반환 (반복자 형식)

- 요소 검색

속성 및 메서드	설명
find_all()	필터링 조건을 만족하는 모든 하위 요소 반환 필터링 조건은 문자열, 정규표현식, 리스트, 함수 등 사용
태그이름()	find_all() 함수의 변형 soup.find_all('a') → soup('a')

주요 속성 및 메서드

- 요소 검색 (계속)

속성 및 메서드	설명
find()	find_all()의 결과 중 첫 번째 요소 반환
find_parents()	필터링 조건을 만족하는 모든 상위 요소 반환
find_parent()	find_parent()의 결과 중 첫 번째 요소 반환
find_previous_siblings()	같은 부모를 갖는 이전 요소 목록 반환
find_previous_sibling()	find_previous_siblings()의 결과 중 첫 번째 요소
find_next_siblings()	같은 부모를 갖는 다음 요소 목록 반환
find_next_sibling()	find_next_siblings()의 결과 중 첫 번째 요소

주요 속성 및 메서드

- 요소 검색 (계속)

속성 및 메서드	설명
find_all_next()	파서의 탐색 순서상 모든 다음 요소 반환
find_next()	find_all_next()의 결과 중 첫 번째 요소 반환
find_all_previous()	파서의 탐색 순서상 모든 이전 요소 반환
find_previous()	find_all_previous()의 결과 중 첫 번째 요소 반환
select()	CSS 선택자에 의해 선택되는 모든 요소 반환

주요 속성 및 메서드

- 문서 변경 (객체 트리 변경)

속성 및 메서드	설명
태그 이름 및 속성 변경	tag.name = ‘태그이름’ tag[‘class’] = ‘속성값’ del tag[‘class’]
태그 내용 변경	tag.string = “New Text String”
new_tag()	새 태그 생성
append()	자식 요소 목록의 끝에 새 요소 추가
insert()	자식 요소 목록의 지정된 위치에 새 요소 추가
insert_before()	첫 번째 previous_sibling으로 추가
insert_after()	첫 번째 next_sibling으로 추가

주요 속성 및 메서드

- 문서 변경 (객체 트리 변경)

속성 및 메서드	설명
clear()	모든 하위 요소 제거
extract()	요소 제거 (제거된 요소는 반환)
decompose()	요소 제거 (제거된 요소는 반환되지 않음)
replace_with()	지정된 요소로 현재 요소 대체
wrap()	새 부모 요소를 추가 (생성된 요소 반환)
unwrap()	부모 요소 제거

주요 속성 및 메서드

- 문서 내용 출력

속성 및 메서드	설명
prettify()	가독성 높은 형식으로 구성된 유니코드 문자열 반환
str()	형식을 포함하지 않는 단순 문자열 반환
get_text()	모든 하위 요소의 문자열 내용을 반환