07. 크롤링에서 시각화까지

“전쟁과 평화” 페이지 분석에 도전해 보자!!

- 지금까지의 파이썬 코딩과 크롤링 지식으로 위 페이지의 데이터를 요청하고, 파싱한 후 원하는 정보를 추출해 보자. (urllib.request로 요청, bs4로 파싱)

- **도전1**
  1. 고유명사(등장 인물)를 검색하여 출력하기
  2. the prince 와 the Empress 중에서 어느 이름이 몇 번 더 많이 등장하는지 분석하기

“전쟁과 평화” 페이지 분석에 도전해 보자!!

- 지금까지의 파이썬 코딩과 크롤링 지식으로 위 페이지의 데이터를 요청하고, 파싱한 후 원하는 정보를 추출해 보자. (urllib.request로 요청, bs4로 파싱)

- **도전2**
  1. 대사만 탐색하여 출력하고, 대사 건수 출력하기
  2. 첫 번째 대사에서 단어 빈도수 계산하기
  3. 2번 이상 나타난 단어들에 대한 빈도수를 막대 그래프로 시각화하여 비교하기
  4. 단어와 빈도수의 쌍들을 CSV 파일로 저장하기

파싱 및 정제의 중요성에 대하여

- 미켈란젤로가 다비드 상과 같은 걸작을 만들어 놓고

"돌에서 다비드처럼 보이지 않는 부분을 깎아내기만 했다"

- 복잡한 웹 페이지에서 필요한 정보를 얻어낸다는 것,

복잡한 HTML 문서를 분석해서
필요하지 않은 정보를 버리고 원하는 정보만 추출하면 된다!!

파싱 및 정제의 중요성에 대하여

- **bs4의 find( )와 findAll()**
  - HTML 페이지에서 원하는 태그를 쉽게 탐색 할 수 있도록 한다.

- **bs4에서 정의하고 있는 두 함수**
  - findAll(tag, attributes, recursive, text, limit, keywords)
  - find(tag, attributes, recursive, text, keywords)

```
bsObj.find(
(name=None, attrs={}, recursive=True, text=None, **kwargs)
Return only the first child of this Tag matching the given
criteria.
```

```
bsObj.findAll(
(name=None, attrs={}, recursive=True, text=None, limit=None, **kwargs)
Extracts a list of Tag objects that match the given
criteria.  You can specify the name of the Tag and any
attributes you want the Tag to have.
```

*** 주로 tag, attributes 만 쓰지만, 각 매개변수의 용도에 대해 간단히 알아본다.

find( )와 findAll() : tag 매개변수의 사용

- 태그 이름으로 된 문자열이나, 태그 이름 문자열들로 이루어진 리스트를 넘긴다. 다음 코드는 문서의 모든 헤더 태그를 찾아서 리스트를 반환한다.

- findAll(**"h1"**) #문서 속의 모든 h1태그 추출해서 리스트 반환

- findAll(**{"h1", "h2", "h3", "h4", "h5", "h6"}**)
  #문서 속의 모든 h1 ~ h6 태그들 추출

find( )와 findAll() : attributes 매개변수

- 속성의 이름과 값으로 이루어진 파이썬 딕셔너리를 받아서 그와 일치하는 태그를 찾아서 리스트를 반환한다.

  findAll("**span**", {"**class**" : "**green**"})

- 다음과 같이 여러 속성 값에 해당하는 태그들을 찾을 수도 있다. 아래 코드는 class 속성의 값이 green 또는 red인 span 태그를 모두 반환한다.

  findAll("**span**", {"**class**" : {"**green**", "**red**"}})

find( )와 findAll() : recursive 매개변수

- 문서에서 얼마나 깊이 찾아 들어가고 싶은지 지정한다.

- recursive 속성 값이 true 이면
  - findAll 함수는 일치하는 태그를 찾아 자식, 자식의 자식, 그 다음까지 재귀적으로 검색한다.

- recursive 속성 값이 false 이면
  - 문서에서 일치하는 최상위 태그만 검색한다.

- **기본적으로 recursive 는 true 로 동작!!!**

find( )와 findAll() : text 매개변수

- 태그의 속성이 아니라, 함수에서 지정하는 문자열과 일치하는 것을 텍스트 컨텐츠에서 찾아서 반환한다. 즉, 태그에 의해 둘러싸인 특정 문자열을 검색한다.

```
wordList = bsObj.findAll(text = "best")
print(len(wordList))
```

- **best**
- best of best
- bestbest

find( )와 findAll() : text 매개변수

- 태그의 속성이 아니라, 함수에서 지정하는 문자열과 일치하는 것을 텍스트 컨텐츠에서 찾아서 반환한다. 즉, 태그에 의해 둘러싸인 특정 문자열을 검색한다.

```
from bs4 import BeautifulSoup

htmlObj = """<p>I am a girl</p><p>best of best</p>
                <font color="blue">best</font>best<br/>"""

bsObj = BeautifulSoup(htmlObj, "lxml")
findList = bsObj.findAll(text = "best")
print(findList)
print(len(findList), "개")
```

```
['best', 'best']
2 개
```

find( )와 findAll() : limit 매개변수

- findAll에서만 사용한다.
- find는 findAll을 호출할 때 limit을 1로 지정한 것과 같다.
- 이 매개변수는 처음 몇 개의 항목만 찾고 싶을 때 사용

```
from bs4 import BeautifulSoup

htmlObj = """<p>I am a girl</p>
            <font color="blue">best</font>
            <font color="blue">best</font>best<br/>"""

bsObj = BeautifulSoup(htmlObj, "lxml")

findList = bsObj.findAll(text = "best", limit = 2)

print(findList)
```

['best', 'best']

find( )와 findAll() : keyword 매개변수

- 특정 속성이 포함된 태그를 찾을 때 사용한다.
- 다음 코드는 "속성의 값이 text 인 태그들을 찾아라" 라는 뜻

```
from bs4 import BeautifulSoup
htmlObj = """<font color="blue">jslee</font>
                <font color="blue">egkim</font>
                <font color="red">shpark</font>"""

bsObj = BeautifulSoup(htmlObj, "lxml")

findList = bsObj.findAll(color = "blue")
print(findList)

findList = bsObj.findAll("", {"color":"blue"})
print(findList)
```

```
[<font color="blue">jslee</font>, <font color="blue">egkim</font>]
[<font color="blue">jslee</font>, <font color="blue">egkim</font>]
```

처음의 과제로 돌아가 보자

고유명사를 찾아서 출력하고, 어떤 인물이 더 많이 나타나는지 분석한다

“전쟁과 평화” 페이지 분석에 도전해 보자!!

- 지금까지의 파이썬 코딩과 크롤링 지식으로 위 페이지의 데이터를 요청하고, 파싱한 후 원하는 정보를 추출해 보자. (urllib.request로 요청, bs4로 파싱)

  ① 고유명사(등장 인물)를 검색하여 출력하기

  ② the prince 와 the Empress 중에서 어느 이름이 몇 번 더 많이 등장하는가 분석하기

http://www.pythonscraping.com/pages/warandpeace.html

크롤링 할 웹 사이트 분석이 먼저다!!

- 수집하고 싶은 데이터
  - 이름, 통계 자료, 텍스트 블록일 수 있다
  - 20 단계나 되는 HTML 문서 속에 있을 수 있다
  - 태그가 없이 깊숙한 곳에 묻혀 있을 수도 있다

  - **무조건 크롤러를 코딩 하는 것이 대수는 아니다!!!**
  - 사이트 관리자가 조금만 수정해도 크롤러가 동작하지 않을 수 있다.

  - **따라서 사이트 분석이 최선이다!!!**

War and Peace

Chapter 1

"Well, Prince, so Genoa and Lucca are now just family estates of the Buona
warn you, if you don't tell me that this means war, if you still try to defend th
horrors perpetrated by that Antichrist- I really believe he is Antichrist- I will h
more to do with you and you are no longer my friend, no longer my 'faithful
call yourself! But how do you do? I see I have frightened you- sit down and
news."

It was in July, 1805, and the speaker was the well-known Anna Pavlovna Sch
honor and favorite of the Empress Marya Fedorovna. With these words she
Vasili Kuragin, a man of high rank and importance, who was the first to arriv
reception. Anna Pavlovna had had a cough for some days. She was, as sh
from la grippe; grippe being then a new word in St. Petersburg, used only by

All her invitations without exception, written in French, and delivered by a scarlet-liveried footman that morning, ran as follows:

"If you have nothing better to do, Count [or Prince], and if the prospect of spending an evening with a poor invalid is not too terrible, I shall be very charmed to see you tonight between 7 and 10- Annette Scherer."

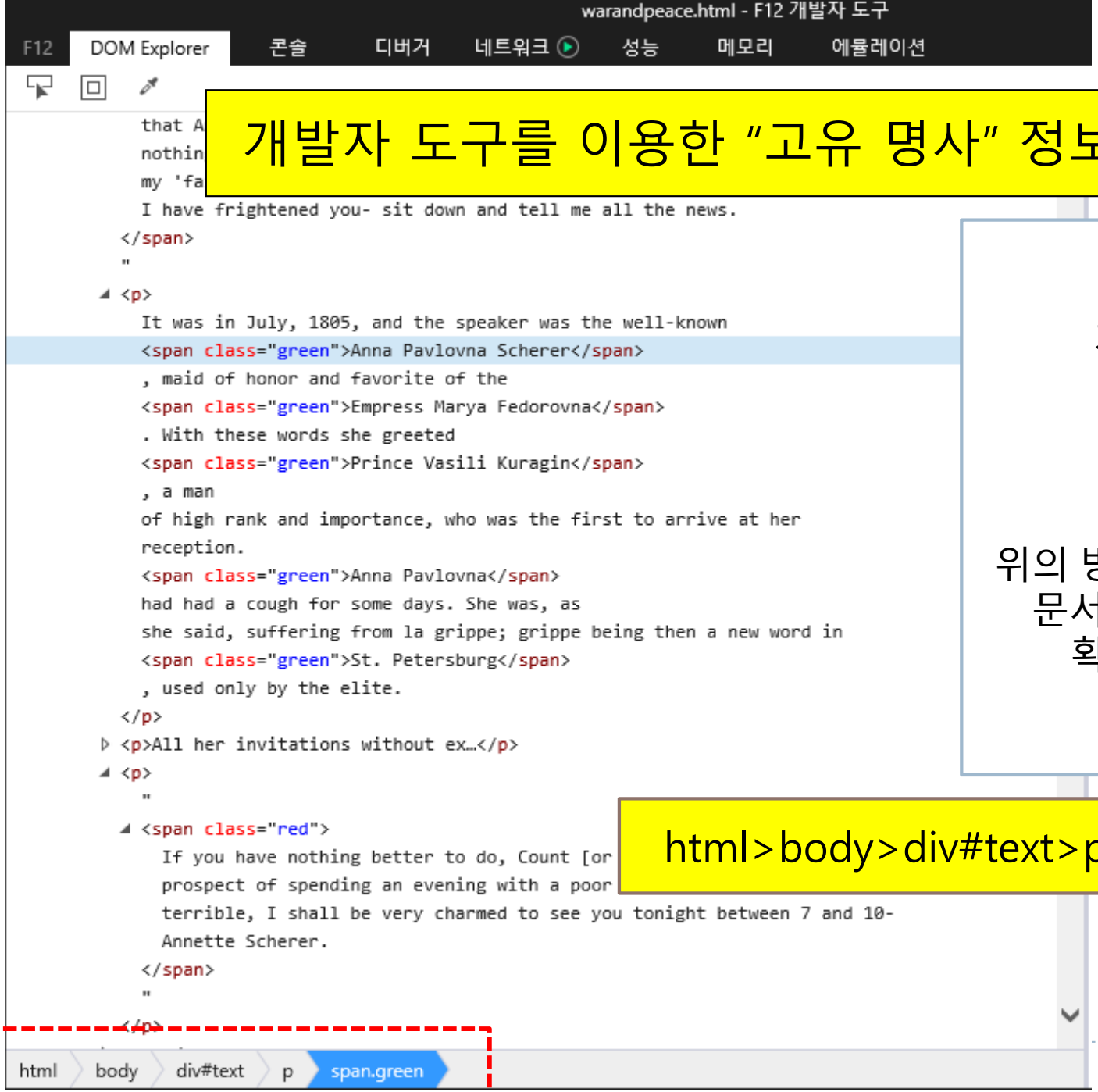
"Heavens! what a virulent attack!" replied the prince, not in the least disconcerted by this reception. He had just entered, wearing an embroidered court uniform, knee breeches, and shoes, and had stars on his breast and a serene expression on his flat face. He spoke in that refined French in which our grandfathers not only spoke but thought, and with the gentle, patronizing intonation natural to a man of importance who had grown old in society and at court. He went up to Anna Pavlovna, kissed her hand, presenting to her his bald, scented, and shining head, and complacently seated himself on the sofa.

크롤링 대상 페이지에 대한 표면적 분석

제목인 듯 큰 텍스트
빨간색 텍스트
초록색 텍스트

빨간색들은 단락으로 구성
초록색이 간간히 나타남

빨간색 단락들은 대사
초록색 텍스트들은 고유명사



개발자 도구를 이용한 "고유 명사" 정보 위치 분석

개발자 도구
(F12)
소스 보기
요소 검사

위의 방법들을 이용하여
문서 객체의 구조를
확인할 수 있다.

html>body>div#text>p>span.green

대상 페이지 요청하기 : urllib.request

```
from urllib.request   import   urlopen


url  =
"http://www.pythonscraping.com/pages/warandpeace.html"


try :
    htmlObj  =  urlopen(url)
except   :
    print("제대로 요청이 안됩니다")
else :
    print(htmlObj.read())     #제대로 요청이 되었나 확인
```

요청해 온 페이지 파싱 : bs4

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

url =
"http://www.pythonscraping.com/pages/warandpeace.html"

try :
    htmlObj = urlopen(url)

except  :
    print("제대로 요청이 안됩니다")

else :
    bsObj = BeautifulSoup(htmlObj, "lxml")
    print(bsObj)
```

파싱 결과 출력 및 탐색 대상 위치 분석

```
<html>
<head>
<style>
.green{
            col
}
.red{
            col
}
#text{
            wi
}
</style>
</head>
```

```
<body>
<h1>War and Peace</h1>
<h2>Chapter 1</h2>
<div id="text">
"<span class="red">Well, Prince, ..... 중간 생략 .....
"<span class="red">I am your faithful slave and to you alone I can confess that my
children are the bane of my life. It is the cross I have to bear. That is how I explain it to myself. It can't be helped!</span>"
<p></p>
He said no more, but expressed his resignation to cruel fate by a gesture. <span class="green">Anna Pavlovna</span> meditated.
</div>
</body>
</html>
```

고유 명사들은 span 태그들 중 class 속성이 green 이다

고유 명사(등장 인물) 탐색을 위한 코드

- findAll( )을 사용하여, 태그에 들어 있는 텍스트만 추출해서 고유 명사로 이루어진 파이썬 리스트를 만들 수 있다.

```
nameList = bsObj.findAll( "span", {"class" : "green"} )
print(nameList)
```

등장 인물 탐색을 위한 코드

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

url = "http://www.pythonscraping.com/pages/warandpeace.html"

try :
    htmlObj = urlopen(url)

except  :
    print("제대로 요청이 안됩니다")

else :
    bsObj = BeautifulSoup(htmlObj, "lxml")
    print(bsObj)

nameList = bsObj.findAll( "span", {"class" : "green"} )
print(nameList)
```

태그와 함께 추출된 등장 인물 정보

<결과>

[Anna Pavlovna Scherer, Empress Marya Fedorovna, Prince Vasili Kuragin, Anna Pavlovna, St. Petersburg, the prince, Anna Pavlovna, Anna Pavlovna, the prince,
… 중간 생략 … class="green">Baron Funke, The prince, Anna Pavlovna, the Empress, The prince, Anatole, the prince, The prince, Anna Pavlovna, Anna Pavlovna, Anna Pavlovna]

태그와 함께 추출된 등장 인물 정보 - 정리

for name in nameList :
print(name)

```
<span class="green">Anna Pavlovna Scherer</span>
<span class="green">Empress Marya Fedorovna</span>
<span class="green">Prince Vasili Kuragin</span>
<span class="green">Anna Pavlovna</span>

… 중간 생략 …

<span class="green">The prince</span>
<span class="green">Anna Pavlovna</span>
<span class="green">Anna Pavlovna</span>
```

nameList의 각 항목에서 텍스트만 추출

- 앞의 리스트의 각 항목은 다음의 꼴이다.

```
<span class="green">Anna Pavlovna Scherer</span>
```

- 여기에서 텍스트만 추출하고 싶다. 어떻게 할까?

```
for name in nameList :
print(name.get_text())
```

nameList[0]
nameList[1]
nameList[2]
:
:
:

Anna
Pavlovna Scherer
Empress Marya
Fedorovna
Prince Vasili Kuragin
… 중간 생략 …
the prince
The prince
Anna
Pavlovna
Anna Pavlovna

nameList의 각 항목에서 텍스트만 추출

- 다음 출력의 결과를 비교해 보자.

```
print(nameList[0])

print(nameList[0].getText())

print(nameList[0].get_text())
```

Anna Pavlovna Scherer

Anna
Pavlovna Scherer

Anna
Pavlovna Scherer

* 리스트 요소에 getText() 함수를 적용할 수 있게 된 것을 볼 때 어떤 사실을 깨닫게 되는가?

다음의 표현은 모두 같은 의미이다!!!

- nameList = bsObj.findAll("span", {"class":"green"})

- nameList = bsObj.findAll("span", **attrs** = {"class":"green"})
- nameList = bsObj.findAll("", {"class":"green"})
- nameList = bsObj.findAll(**class_**="green")

  # class 를 바로 사용하는 이것은 비표준적 방법으로 syntax error 발생

둘 중에서 어떤 인물이 더 많이 등장했나?

"the prince" 와 "the Empress" 중
어느 이름이 더 많이 나왔는지
분석해 봅시다.

```
princeText = bsObj.findAll(text = "the prince")

n = len(princeText)
print("the prince  " + str(n) + "회 나타남")
print("----------------------")

for  i  in princeText :
        print(i)

print("\n\n")
```

```
the prince  7회 나타남
----------------------
the prince
the prince
the prince
the prince
the prince
the prince
the prince


the Empress  3회 나타남
----------------------
the Empress
the Empress
the Empress
```

2

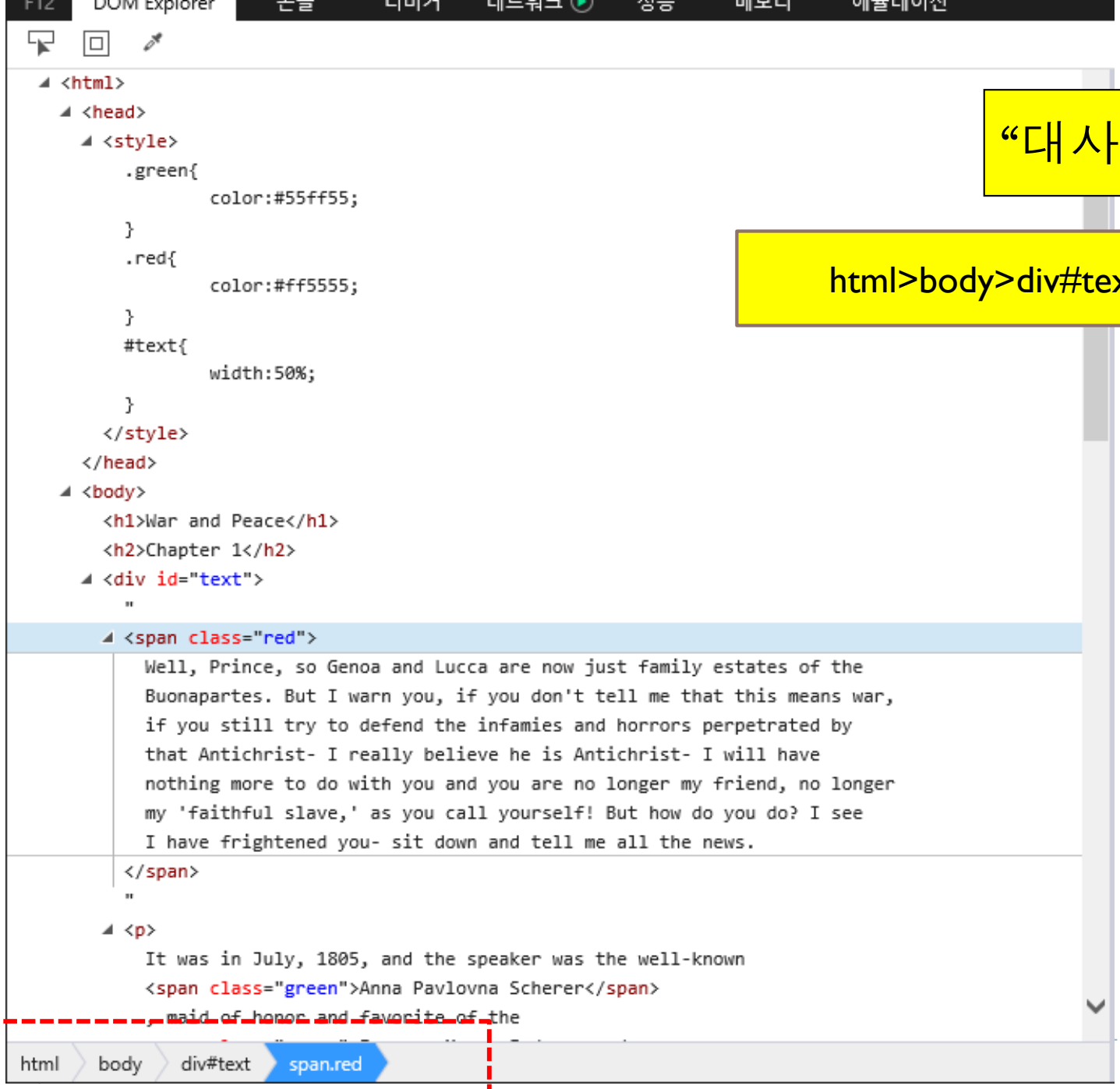
처음의 과제로 돌아가 보자

대사를 찾아서 출력하고, 대사 건수를 출력한다
첫 번째 대사에서 2번 이상 나타나는 단어들의 빈도수를 분석한다
단어와 빈도수의 관계를 막대 그래프로 나타내어 비교한다
단어와 빈도수의 쌍들을 CSV 파일로 저장한다

“전쟁과 평화” 페이지 분석에 도전해 보자!!

- 지금까지의 파이썬 코딩과 크롤링 지식으로 위 페이지의 데이터를 요청하고, 파싱한 후 원하는 정보를 추출해 보자. (urllib.request로 요청, bs4로 파싱)

  ① 대사만 탐색하여 출력하고, 대사 건수 출력하기
  ② 첫 번째 대사에서 단어 빈도수 계산하기
  ③ 2번 이상 나타난 단어들에 대한 빈도수를 막대 그래프로 시각화하여 비교하기
  ④ 단어와 빈도수의 쌍들을 CSV 파일로 저장하기



"대사" 정보 위치

html>body>div#text>span.red

두 번째 문제 해결을 위한 시작!!

1. 페이지 요청 및 파싱

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

url = "http://www.pythonscraping.com/pages/warandpeace.html"

try :
    htmlObj = urlopen(url)
except :
    print("요청이 제대로 되지 않았습니다")
else :
    bsObj = BeautifulSoup(htmlObj, "lxml")

print(bsObj.prettify())
```

대사만 추출하여 출력해 봅시다

```
diaList = bsObj.findAll("span", {"class" : "red"})

for dia in diaList :
        print(dia.get_text() + "\n\n")
```

- 몇 개의 대사가 존재하는가?

```
for dia in diaList :
        print(dia.get_text() + "\n\n")

print("이 페이지에 존재하는 대사는 총 " + str(len(diaList)) +  " 개 입니다!!!")
```

첫 번째 대사 속의 단어 빈도수 분석하기

2. 첫 대사만 가져와서 단어 분리하기

```
findObj = bsObj.find("span", {"class" : "red"})
textObj = findObj.get_text()

#단어 추출과 관계없는 문자 제거하기
textObj = textObj.replace("\n" , "")
textObj = textObj.replace("-" ,  "")
textObj = textObj.replace(", " ,  "")

#긴 문자열을 공백을 기준으로 분리하기
splObj = textObj.split("  ")     # 리스트 반환
print(splObj)
```

첫 번째 대사 속의 단어 빈도수 분석하기

3. 단어 별로 빈도수 계산하기

```
word_freq = {}    # 단어와 빈도수를 쌍으로 저장할 딕셔너리 생성

for word in splObj :
    word_freq.setdefault(word, 0) #모든 단어의 빈도수 값을 0으로 초기화
    word_freq[word] += 1      # 각 단어의 카운트를 증가시킴

print(word_freq)
```

첫 번째 대사 속의 단어 빈도수 분석하기

4. 시각화를 위한 자료 구조 만들기

```
xList=[ ]    # x축 자료로 사용할 단어 이름들을 저장할 리스트 생성
yList=[ ]    # y축 자료로 사용할 각 단어의 빈도수들을 저장할 리스트 생성
```

첫 번째 대사 속의 단어 빈도수 분석하기

5. 딕셔너리의 키와 값들을 각 리스트에 저장하기

```
for  word  in word_freq :
        xList.append(word)
        yList.append(word_freq[word])

print(xList)
print(yList)
```

첫 번째 대사 속의 단어 빈도수 분석하기

6. 단어와 빈도수의 관계를 막대 그래프로 시각화하기

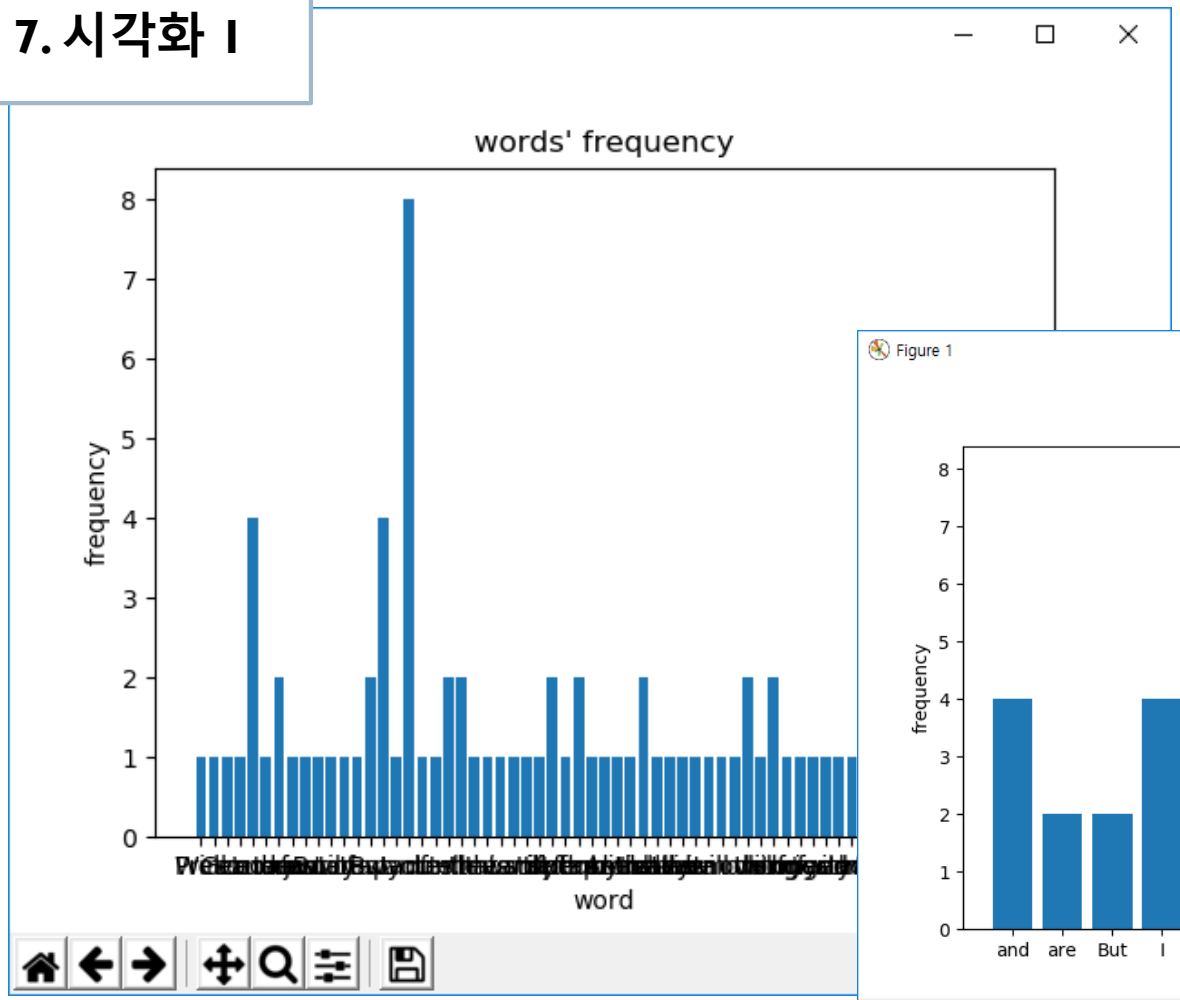
```
from matplotlib import pyplot

pyplot.bar(xList, yList)
pyplot.xlabel("word")
pyplot.ylabel("frequency")
pyplot.title("words' frequency")

pyplot.show ()
```

matplotlib 라이브러리의 **pyplot** 모듈

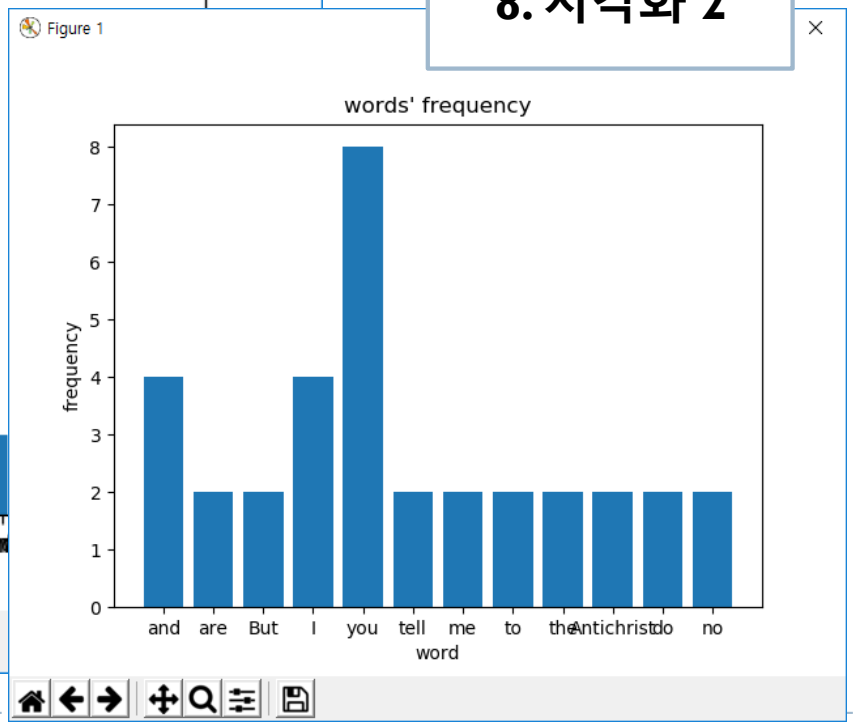
- 설치해야 쓸 수 있다.
- 다소 오래되었지만 널리 사용되고 있는 것!!!
- 간단한 막대 그래프, 선 그래프 등에 적합하다.
- 웹을 위한 복잡하고 인터랙티브한 시각화에는 적합하지 못하다!!!

첫 번째 대사 속의 단어 빈도수 분석하기

7. 시각화 1



8. 시각화 2



첫 번째 대사 속의 단어 빈도수 분석하기

9. 딕셔너리 정보를 파일로 저장하기

```
import csv

with open("wordcount.csv", "w") as myFile :
    myFile.write("words, frequency\n")
    for word in word_freq :
        myFile.write("{0}, {1}\n" .format(word, word_freq[word]))

# 소스 코드와 같은 폴더에 wordcount.csv 이름으로 저장된다
```