실습 2-3-1-1. 단어의 검색 결과 출력하기

- 다음 어학사전(http://dic.daum.net/index.do?dic=all)에 (http://dic.daum.net/index.do?dic=all)에) ‘curiosity’ 단어를 검색하였을 때 출력 되는 화면에서 가장 상단의 결과를 출력한다
- 영어 단어의 의미(본 예제에서는 '1. 호기심 2. 큐리오시티')를 출력한다

In [ ]:

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

In [ ]:

```
# 검색하고 싶은 단어 입력하기
word = 'curiosity'

# 불러오려는 url 입력하기
url = 'http://dic.daum.net/search.do?q=' + word
# 디폴트 url에 string 타입의 word 변수를 합쳐서 url 변수 생성

# urlopen 함수를 통해 web 변수를 생성
web = urlopen(url)

# BeautifulSoup으로 web 페이지상의 HTML 구조를 파싱
web_page = BeautifulSoup(web, 'html.parser')

# find 함수를 통해 찾고자 하는 단어와 단어의 뜻이 있는 HTML 태그를 찾음
box = web_page.find('div', {'class': 'search_cleanword'}).find('span', {'class': 'txt_emph1'})
# 'curiosity' 텍스트가 있는 곳
box2 = web_page.find('ul', {'class': 'list_search'})
                                    # 단어의 뜻이 있는 곳
```

In [4]:

```
print(box.get_text())

print(box2.get_text())
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-4-b182a7dfbc2f> in <module>()
----> 1 print(box.get_text())
      2 print(box2.get_text())

NameError: name 'box' is not defined
```

실습 2-3-1-2. 여러 단어의 검색 결과 출력하기

- 다음 어학사전(http://dic.daum.net/index.do?dic=all)에 (http://dic.daum.net/index.do?dic=all)에) ‘curiosity’ , ‘killed’, ‘the’, ‘cat’ 4개 단어를 검색하였을 때 출력 되는 화면에서 가장 상단의 결과를 출력한다
- 각 영어 단어의 의미를 출력한다

In [4]:

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

In [5]:

```
# 검색하고 싶은 단어 입력하기
words = ['curiosity', 'killed', 'the', 'cat']                    # 단어를 여러 개 검색할 것이므로
 복수의 단어를 리스트에 저장한다

# for문을 통해 각각의 단어와 그 뜻을 출력한다
for word in words:
# 단어의 리스트에서 각각의 단어를 꺼냄
        url = 'http://dic.daum.net/search.do?q=' + word          # 불러오려는 url 입력하기
        web = urlopen(url)
# urlopen 함수를 통해 web 변수를 생성
        web_page = BeautifulSoup(web, 'html.parser')             # BeautifulSoup으로 web 페이지상
의 HTML 구조를 파싱
        # find 함수를 통해 찾고자 하는 단어와 단어의 뜻이 있는 HTML 태그를 찾음
        box = web_page.find('div', {'class': 'search_cleanword'}).find('span',{'class': 'txt_emp
h1'})   # 단어가 있는 곳
        box2 = web_page.findAll('ul', {'class': 'list_search'})[0]
                                                 # 단어의 뜻이 있는 곳
        # get_text 함수로 텍스트를 추출하고 이를 출력한다
        print(box.get_text())
        print(box2.get_text())
```

```
curiosity

1.호기심
2.큐리오시티

killed

1.죽음
2.사망
3.살해되다
4.목숨을 잃다

the

1.그
2.그럴수록
3.더욱더

cat

1.고양이
2.고양이과 동물
```

실습 2-3-1-3. 여러 단어의 검색 결과 저장하기

- 다음 어학사전(http://dic.daum.net/index.do?dic=all)에 (http://dic.daum.net/index.do?dic=all)에) ‘curiosity’ , ‘killed’, ‘the’, ‘cat’ 4개 단어를 검색하였을 때 출력 되는 화면에서 가장 상단의 결과를 텍스트 파일에 저장한다
- 각 영어 단어의 의미를 저장한다

In [ ]:

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

In [ ]:

```
# 검색하고 싶은 단어 입력하기
words = ['curiosity', 'killed', 'the', 'cat']                                    # 단어를 여러 개
 검색할 것이므로 복수의 단어를 리스트에 저장한다

# open 함수를 통해 'result-1-1-3.txt' 파일을 열고 이를 f로 지정한다
with open('result-1-1-3.txt', 'w', encoding = 'utf-8') as f:    # 쓰기 형식('w')로 지정하고 인코
딩은 'utf-8'로 설정한다
        for word in words:
                # for문을 통해 각각의 단어와 그 뜻을 저장한다
                url = 'http://dic.daum.net/search.do?q=' + word                  # 불러오려는 url
 입력하기
                web = urlopen(url)
                # urlopen 함수를 통해 web 변수를 생성
                web_page = BeautifulSoup(web, 'html.parser')                     # BeautifulSoup
으로 web 페이지상의 HTML 구조를 파싱
                # find 함수를 통해 찾고자 하는 단어와 단어의 뜻이 있는 HTML 태그를 찾음
                box = web_page.find('div', {'class': 'search_cleanword'}).find('span',{'class':
'txt_emph1'})   # 단어가 있는 곳
                box2 = web_page.find('ul', {'class': 'list_search'})
                                                        # 단어의 뜻이 있는 곳
                # get_text 함수로 텍스트를 추출하고 이를 저장한다
                # write 함수로 파일에 내용을 쓴다
                f.write(box.get_text() + '₩n')
                f.write(box2.get_text() + '₩n')
        f.close()
                        # 파일을 닫는다
```

실습 2-3-1-4. 여러 단어를 불러와 검색 결과 저장하기

- 다음 어학사전(http://dic.daum.net/index.do?dic=all)에 (http://dic.daum.net/index.do?dic=all)에) 텍스트 파일(data.txt)에 있는 각 단어를 검색하였을 때 출력 되는 화면에서 가장 상단의 결과를 텍스트 파일에 저장한다
- 각 영어 단어의 의미를 저장한다

In [ ]:

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

In [ ]:

```
# open 함수를 통해 'data-1-1-4.txt' 파일을 열고 그 내용을 가져온다
with open('data-1-1-4.txt', 'r', encoding = 'utf-8') as f:                       # 읽기 형식('r')
로 지정하고 인코딩은 'utf-8'으로 설정한다
        words = f.readlines()
                # readlines 함수를 통해 전체 텍스트 파일의 내용을 리스트로 가져온다 (줄로 구분)
        f.close()
                                # 파일을 닫는다
```

In [ ]:

```
# open 함수를 통해 'result-1-1-4.txt' 파일을 열고 이를 f로 지정한다
with open('result-1-1-4.txt', 'w', encoding = 'utf-8') as f:            # 쓰기 형식('w')로 지정하고 인코딩은 'utf-8'로 설정한다
        for word in words:
                        # for문을 통해 각각의 단어와 그 뜻을 저장한다
                url = 'http://dic.daum.net/search.do?q=' + word                                 # 불러오려는 url 입력하기
                web = urlopen(url)
                        # urlopen 함수를 통해 web 변수를 생성
                web_page = BeautifulSoup(web, 'html.parser')                                    # BeautifulSoup으로 web 페이지상의 HTML 구조를 파싱
                box = web_page.find('div', {'class': 'search_cleanword'}).find('span',{'class': 'txt_emph1'})           # 단어가 있는 곳
                box2 = web_page.find('ul', {'class': 'list_search'})
                                                                                # 단어의 뜻이 있는 곳
                # get_text 함수로 텍스트를 추출하고 이를 저장한다
                # write 함수로 파일에 내용을 쓴다
                f.write(box.get_text() + '₩n')
                f.write(box2.get_text() + '₩n')
        f.close()
                                # 파일을 닫는다
```

실습 2-3-2-1. 비트코인 관련 데이터 출력하기

- 코인마켓캡(https://coinmarketcap.com/ (https://coinmarketcap.com/)) 메인 페이지에서 가장 상단에 있는 비트코인의 화폐 이름(Name)과 시가총액(Market Cap), 개당 가격(Price), 그리고 24시간 볼륨(Volume)을 출력해 본다
- 한 줄에 하나씩 출력되도록 한다

In [ ]:

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

In [ ]:

```
# 불러오려는 url 입력하기 - 코인마켓캡 닷컴
url = 'https://coinmarketcap.com/'

# urlopen 함수를 통해 web 변수를 생성
web = urlopen(url)

# BeautifulSoup으로 web 페이지상의 HTML 구조를 파싱
web_page = BeautifulSoup(web, 'html.parser')

# 비트코인과 관련된 데이터가 있는 곳을 찾아 bit 변수에 저장한다
table = web_page.find('table', {'id': 'currencies'})    # 1위부터 100위까지의 cryptocurrency 데이터가 있는 테이블을 찾는다
bit = table.find('tr', {'id': 'id-bitcoin'})                    # 비트코인 데이터가 행(첫째 행)의 데이터를 찾는다
```

In [ ]:

```
# 비트코인 관련된 데이터를 출력한다
coinName = bit.find('td', {'class': 'no-wrap currency-name'})
print('Coin Name: ', coinName.get_text().strip())
        # 코인의 이름('Bitcoin')을 출력한다
marketCap = bit.find('td', {'class': 'no-wrap market-cap text-right'})
print('Market Cap: ', marketCap.get_text().strip())
        # 코인의 시가총액(Market cap)을 출력한다
price = bit.find('a', {'class': 'price'})

print('Price: ', price.get_text().strip())
                # 코인의 가격(price)를 출력한다
volume = bit.find('a', {'class': 'volume'})
print('Volume: ', volume.get_text().strip())
        # 코인의 24시간 볼륨(volume)을 출력한다
```

실습 2-3-2-2. 상위 100개 코인 관련 데이터 저장하기

- 코인마켓캡(https://coinmarketcap.com/ (https://coinmarketcap.com/)) 메인 페이지에 있는 상위 100개 코인의 화폐 이름(Name)과 시가총액(Market Cap), 개당 가격(Price), 그리고 24시간 볼륨(Volume)을 텍스트 파일로 저장한다
- 한 줄에 코인 하나의 데이터를 저장하고, 각 행에서는 탭으로 구분한다

In [ ]:

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

In [ ]:

```
# 불러오려는 url 입력하기 - 코인마켓캡 닷컴
url = 'https://coinmarketcap.com/'

# urlopen 함수를 통해 web 변수를 생성
web = urlopen(url)

# BeautifulSoup으로 web 페이지상의 HTML 구조를 파싱
web_page = BeautifulSoup(web, 'html.parser')

# 코인과 관련된 데이터가 있는 곳을 찾는다
table = web_page.find('table', {'id': 'currencies'})
        # 1위부터 100위까지의 cryptocurrency 데이터가 있는 테이블을 찾는다
coinNames = table.findAll('td', {'class': 'no-wrap currency-name'})
# 코인의 이름(name)이 있는 곳들을 찾아 리스트로 저장한다
marketCaps = table.findAll('td', {'class': 'no-wrap market-cap text-right'})    # 코인의 시가총액(market cap)이 있는 곳들을 찾아 리스트로 저장한다
prices = table.findAll('a', {'class': 'price'})
                # 코인의 가격(price)이 있는 곳들을 찾아 리스트로 저장한다
volumes = table.findAll('a', {'class': 'volume'})
                # 코인의 24시간 볼륨(volume)이 있는 곳들을 찾아 리스트로 저장한다
```

In [ ]:

```
# open 함수를 통해 'result-1-2-2.txt' 파일을 열고 이를 f로 지정한다
with open('result-1-2-2.txt', 'w', encoding = 'utf-8') as f:
# 쓰기 형식('w')로 지정하고 인코딩은 'utf-8'로 설정한다
        # for문을 통해 시가총액 1위부터 100위까지의 코인의 정보를 저장한다
        for i in range(100):
                                        # 리스트 인덱스 접근을 위해 0부터 100까지 iterate 한다
                # 하나의 코인에 대한 정보를 row에 입력한다
                # 중간중간에 '₩t'를 입력하여 각각의 데이터를 구분한다
                row = coinNames[i].get_text().strip() + '₩t' + marketCaps[i].get_text().strip()
+ '₩t' ₩
                        + prices[i].get_text().strip() + '₩t' + volumes[i].get_text().strip() +
'₩n'
                f.write(row)
                                                # row를 파일에 쓴다
        f.close()
                                                        # 파일을 닫는다
```

실습 2-3-2-3. 특정 조건을 만족하는 코인 관련 데이터 저장하기

- 상위 100개의 코인 중 24시간 볼륨(Volume)이 $1,000,000$이상이고개당가격이50 이상인 코인들의 데이터만 저장한다
- 한 줄에 코인 하나의 데이터를 저장하고, 각 행에서는 탭으로 구분한다

In [ ]:

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

In [ ]:

```
# 불러오려는 url 입력하기 - 코인마켓캡 닷컴
url = 'https://coinmarketcap.com/'

# urlopen 함수를 통해 web 변수를 생성
web = urlopen(url)

# BeautifulSoup으로 web 페이지상의 HTML 구조를 파싱
web_page = BeautifulSoup(web, 'html.parser')

# 코인과 관련된 데이터가 있는 곳을 찾는다
table = web_page.find('table', {'id': 'currencies'})
        # 1위부터 100위까지의 cryptocurrency 데이터가 있는 테이블을 찾는다
coinNames = table.findAll('td', {'class': 'no-wrap currency-name'})
# 코인의 이름(name)이 있는 곳들을 찾아 리스트로 저장한다
marketCaps = table.findAll('td', {'class': 'no-wrap market-cap text-right'})    # 코인의 시가총
액(market cap)이 있는 곳들을 찾아 리스트로 저장한다
prices = table.findAll('a', {'class': 'price'})
                # 코인의 가격(price)이 있는 곳들을 찾아 리스트로 저장한다
volumes = table.findAll('a', {'class': 'volume'})
                # 코인의 24시간 볼륨(volume)이 있는 곳들을 찾아 리스트로 저장한다
```

In [ ]:

```
# open 함수를 통해 'result-1-2-3.txt' 파일을 열고 이를 f로 지정한다
with open('result-1-2-3.txt', 'w') as f:
                        # 쓰기 형식('w')로 지정하고 인코딩은 'utf-8'로 설정한다
# for문을 통해 시가총액 1위부터 100위까지의 코인의 데이터에 접근한다
        for i in range(100):
                price = float(prices[i].get_text().strip().replace('$','').replace(',',''))
        # 코인의 가격을 가져온다(float 타입으로)
                volume = float(volumes[i].get_text().strip().replace('$','').replace(',',''))
# 코인의 볼륨을 가져온다(float 타입으로)
                # if문을 통해 저장 조건을 준다
                if volume >= 1000000 and price >= 50:
                                        # 볼륨이 1,000,000 이상이고 가격이 50 이상일 경우:
                        # 하나의 코인에 대한 정보를 row에 입력한다
                        # 중간중간에 '₩t'를 입력하여 각각의 데이터를 구분한다
                        row = coinNames[i].get_text().strip() + '₩t' +
marketCaps[i].get_text().strip() + '₩t' ₩
                                + prices[i].get_text().strip() + '₩t' + volumes[i].get_text().st
rip() + '₩n'
                        f.write(row)
                                                                # row를 파일에 쓴다
        f.close()
                                                                        # 파일을 닫는다
```

실습 2-3-3-1. '펄프 픽션' 영화 정보 출력하기

- IMDb 사이트에서 영화 펄프 픽션 페이지(http://www.imdb.com/title/tt0110912/?ref_=nv_sr_1)에서 (http://www.imdb.com/title/tt0110912/?ref_=nv_sr_1)에서) 펄프 픽션 영화의 제목(title)과 감독(director), 그리고 출연 배우(cast)를 출력한다

In [ ]:

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

In [ ]:

```
# 불러오려는 url 입력하기 (IMDb - 펄프 픽션)
url = 'http://www.imdb.com/title/tt0110912/?ref_=nv_sr_1'

# urlopen 함수를 통해 web 변수를 생성
web = urlopen(url)

# BeautifulSoup으로 web 페이지상의 HTML 구조를 파싱
web_page = BeautifulSoup(web, 'html.parser')
```

In [ ]:

```
# 영화 제목을 출력한다
titleBar = web_page.find('div', {'class': 'titleBar'})          # 타이틀(영화 제목)이 있는 바(bar)를 찾는다
title = titleBar.find('h1', {'itemprop': 'name'})                    # 타이틀바에서 영화 제목이 있는 곳을 찾는다
print('Movie: ' , title.get_text())                                                    # get_text 함수로 영화 제목을 출력한다

# 감독 이름을 출력한다
director = web_page.find('span', {'itemprop': 'director'})      # 감독 이름이 있는 곳을 찾는다
print('Director: ', director.get_text().strip())                          # get_text 함수로 감독 이름을 출력한다

# 출연 배우 이름(들)을 출력한다
table = web_page.find('table', {'class': 'cast_list'})          # 캐스팅 리스트가 있는 테이블을 찾는다
find = table.findAll('span', {'class': 'itemprop', 'itemprop': 'name'})         # 배우 이름이 있는 곳들을 찾아 리스트에 저장한다
print('Cast: ')
# for문을 통해 각 배우의 이름을 출력한다
for i in find:
        print(i.get_text())
```

실습 2-3-3-2. 여러 영화 정보 저장하기

- 영화 펄프 픽션(Pulp Fiction), 포레스트 검프(Forrest Gump), 그리고 다크 나이트(The Dark Knight)의 제목(title)과 감독 이름(director)를 텍스트 파일에 저장한다
- 영화 하나당 한 줄씩 출력되도록 한다

In [ ]:

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

In [ ]:

```
# 불러오려는 url 입력하기 - 세 개의 url이 있으므로 이를 리스트에 저장한다
urls = ['http://www.imdb.com/title/tt0110912/?ref_=nv_sr_1', ₩
                'http://www.imdb.com/title/tt0109830/?ref_=tt_rec_tt', ₩
                'http://www.imdb.com/title/tt0468569/?ref_=nv_sr_3']
```

In [ ]:

```
info = []
                        # 각 영화의 정보를 담기 위한 리스트를 생성

# 각각의 url에 접속하면서 영화 정보를 가져온다
for url in urls:
        web = urlopen(url)
                # urlopen 함수를 통해 web 변수를 생성
        web_page = BeautifulSoup(web, 'html.parser')                                    # BeautifulSoup
으로 web 페이지상의 HTML 구조를 파싱
        titleBar = web_page.find('div', {'class': 'titleBar'})          # 타이틀(영화 제목)이 있
는 바(bar)를 찾는다
        title = titleBar.find('h1', {'itemprop': 'name'})                               # 타이틀바에서
 영화 제목이 있는 곳을 찾는다
        director = web_page.find('span', {'itemprop': 'director'})      # 감독 이름이 있는 곳을
 찾는다

        # info 리스트에 각 영화의 정보를 저장한다
        info.append(title.get_text().strip() + '₩t' + director.get_text().strip())
```

In [ ]:

```
# open 함수를 통해 'result-1-3-2.txt' 파일을 열고 이를 f로 지정한다
with open('result-1-3-2.txt', 'w', encoding = 'utf-8') as f:    # 쓰기 형식('w')로 지정하고 인코
딩은 'utf-8'로 설정한다
        # info 리스트의 각 요소의 내용을 파일에 쓴다
        for i in info:
                f.write(i + '₩n')
                # 각 정보는 새로운 줄('₩n')으로 구분한다
        f.close()
                        # 파일을 닫는다
```

실습 2-3-3-3. 영화 리뷰 출력하기

- 영화 다크 나이트(The Dark Knight)의 리뷰 페이지에 들어가 첫 번째 리뷰 내용을 출력해 보자

In [ ]:

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

In [ ]:

```
# 불러오려는 url 입력하기
url = 'http://www.imdb.com/title/tt0468569/reviews?start=0'

# urlopen 함수를 통해 web 변수를 생성
web = urlopen(url)

# BeautifulSoup으로 web 페이지상의 HTML 구조를 파싱
source = BeautifulSoup(web, 'html.parser')

# 리뷰 데이터가 있는 테이블을 찾는다
table = source.find("div", {"id": "tn15content"})

# 테이블 내의 모든 paragraph('p')를 찾아 리스트에 저장한다
pars = table.findAll('p')
```

In [ ]:

```
# 각각의 paragraph의 내용을 출력한다
for par in pars:
        # 리뷰일 경우에만 출력하기 위해 if 조건문을 활용한다
    if ('This review may contain spoilers' not in par) and ('Add another review' not in par):
        print(par.get_text().replace('₩n', ' ').replace('₩r', ' '))
        break
```

실습 2-3-3-4. 영화 리뷰 저장하기

- 영화 다크 나이트(The Dark Knight)의 첫 번째 부터 10번째 페이지까지의 리뷰를 텍스트 파일에 저장한다

In [ ]:

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

In [ ]:

```
# 시작 URL을 입력한다
baseURL = 'http://www.imdb.com/title/tt0468569/reviews?start='

# urlopen 함수를 통해 web 변수를 생성('0'을 더해 첫 번째 리뷰부터 시작함을 알려준다)
web = urlopen(baseURL + '0')

# BeautifulSoup으로 web 페이지상의 HTML 구조를 파싱
source = BeautifulSoup(web, 'html.parser')

# 리뷰 데이터가 있는 테이블을 찾는다
table = source.find('div', {'id': 'tn15content'})

page_no = 10                    # 출력하고자 하는 총 페이지 개수를 입력
reviewList = []                 # 리뷰 내용을 담고자 하는 리스트를 생성
status = True                   # while문을 탈출하기 위한 조건을 설정하는 status 변수 생성
i = 0                           # 현재 크롤링한 리뷰 수를 기록하기 위한 i 변수 생성
```

In [ ]:

```
# while문을 통해 10 번째 페이지까지의 리뷰를 크롤링한다
while status:
    url = baseURL + str(i)      # 시작 URL에 i를 더해 크롤링하고자 하는 페이지를 입력한다
    web = urlopen(url)          # urlopen 함수를 통해 web 변수를 생성
    source = BeautifulSoup(web, 'html.parser')          # BeautifulSoup으로 web 페이지상의 HTML
 구조를 파싱
    table = source.find('div', {'id': 'tn15content'})   # 리뷰 데이터가 있는 테이블을 찾는다

    pars = table.findAll('p')                           # 테이블 내의 모든 paragraph('p')를 찾아
 리스트에 저장한다

    # 각각의 paragraph의 내용을 저장한다
    for par in pars:
        text = par.get_text().replace('₩n', ' ').replace('₩r', ' ')
        # 리뷰일 경우에만 저장하기 위해 if 조건문을 활용한다
        if ('This review may contain spoilers' not in text) and ('Add another review' not in te
xt):
            reviewList.append(text)
    i += 10                                             # 하나의 페이지를 크롤링할때마다 i를 10씩 증
가시킨다
    if i >= page_no * 10:                               # 만약 현재 크롤링한 페이지 개수가 10을 넘어
가면:
        status = False                                  # while문을 탈출한다
```

In [ ]:

```
# open 함수를 통해 'result-1-3-4.txt' 파일을 열고 이를 f로 지정한다
with open('result-1-3-4.txt', 'w', encoding = 'utf-8') as f:        # 쓰기 형식('w')로 지정하고
 인코딩은 'utf-8'로 설정한다
    for review in reviewList:                                       # for문을 통해 reviewList의
 각각의 리뷰를 파일에 쓴다
        f.write(review + '₩n')
    f.close()                                                       # 파일을 닫는다
```