Python 응용

2016년도 2학기 2학년 방과후학교

CSV 파일 다루기

CSV(영어: comma-separated values)는 몇 가지 필드를 쉼표(,)로 구분한 텍스트 데이터 및 텍스트 파일이다. 확장자는 .csv이며 MIME 형식은 text/csv이다. comma-separated variables라고도 한다.

CVS 실습 1 - 지하철 승하차 인원 (CVS 파일)

"사용월","호선명","지하철역","승차인원","하차인원","작업일자","비고"
"USE_MON","LINE_NUM","SUB_STA_NM","RIDE_PASGR_NUM","ALIGHT_PASGR_NUM","WORK_DT","COMMT"
"201509","중앙선","용문","105047","103005","20151008",""
"201509","중앙선","원덕","13473","12468","20151008",""
"201509","중앙선","양평","123365","123574","20151008",""
"201509","중앙선","오빈","10600","10507","20151008",""
"201509","중앙선","아신","29787","29992","20151008",""
"201509","중앙선","국수","29653","28371","20151008",""
"201509","중앙선","신원","11023","8967","20151008",""
"201509","중앙선","양수","67176","67081","20151008",""
"201509","중앙선","운길산","42754","39118","20151008",""
"201509","중앙선","팔당","32997","32931","20151008",""
"201509","중앙선","도심","79871","66969","20151008",""
"201509","중앙선","덕소","195700","197400","20151008",""
"201509","중앙선","양정","27008","23336","20151008",""
...

CSV 파일을 잘 보자!!
Header 두 줄 + 데이터 n줄
데이터 타입은 모두 문자열

CVS 실습 1 - 지하철 승하차 인원 (코드)

```
import csv
from collections import Counter

# subway.csv 파일을 읽어 승하차 인원이 많은 지하철역 TOP 5를 뽑아보자
in_counter = Counter()      # dictionary
out_counter = Counter()

with open('subway.csv', encoding='utf-8') as f:
    f.readline()     # header 읽기
    f.readline()

    reader = csv.reader(f)

    # "사용월", "호선명", "지하철역", "승차인원", "하차인원", "작업일자", "비고"
    for line in reader:
        # 승차인원, 하차인원 뽑기
        in_num = line[3].strip()
        out_num = line[4].strip()

        # key = 호선명 + 지하철역
        key = line[1].strip() + ' ' + line[2].strip()

        in_counter[key] = int(in_num)
        out_counter[key] = int(out_num)
```

```
print('< 승차인원 TOP 5 >')
for data in in_counter.most_common(5):
    print(data[0], data[1])

print('< 하차인원 TOP 5 >')
for data in out_counter.most_common(5):
    print(data[0], data[1])
```

CVS 실습 2 - 도서관 좌석, 자료수 (CVS 파일)

도서관명,시도명,시군구명,도서관유형,운영시작시각,운영종료시각,열람좌석수,자료수(도서),자료수(연속간행물),자료수(비도서),대출가능권수,대출가능일수

한천하누리작은도서관,전라남도,화순군,작은도서관,15:00,20:00,8,1000,0,0,3,14

상대한보 작은도서관,경상남도,진주시,작은도서관,9:00,18:00,20,1000,0,0,0,0

연세 작은도서관,경상남도,진주시,작은도서관,12:00,19:00,18,1000,0,0,0,0

초전 해모로 루비채 2단지 북카페,경상남도,진주시,작은도서관,12:00,17:00,50,1010,0,0,3,7

대곡면 새마을문고,경상남도,진주시,작은도서관,9:00,13:00,16,1100,0,0,0,0

잠정햇살마을도서관,전라남도,화순군,작은도서관,15:00,22:00,39,1150,0,0,3,14

행복소리 작은도서관,경상남도,진주시,작은도서관,9:00,18:00,12,1150,0,0,0,0

해망마을도서관,전라남도,화순군,작은도서관,14:00,17:00,30,1200,0,0,3,14

마중물 작은도서관,전라남도,화순군,작은도서관,13:00,19:00,12,1200,0,0,3,14

한우리 독서문화원,경상남도,진주시,작은도서관,10:00,13:00,24,1200,0,0,0,0

꿈이자라는문고,충청북도,영동군,작은도서관,9:00,18:00,10,1200,0,0,2,7

궁촌작은도서관,충청북도,영동군,작은도서관,9:00,18:00,6,1200,0,0,2,7

능주면민도서관,전라남도,화순군,작은도서관,9:00,22:00,36,1209,0,0,3,14

사북공공도서관,강원도,정선군,공공도서관,9:00,18:00,213,62064,,,6,14

…

CSV 파일을 잘 보자!!
Header 한 줄 + 데이터 n줄
중간에 없는 데이터도 있음

CVS 실습 2 - 도서관 좌석, 자료수 (코드)

```
import csv
from collections import Counter

seat_counter = Counter()
data_counter = Counter()

with open('library.csv', encoding='utf-8') as f:
    f.readline()   # header 읽기
    reader = csv.reader(f)

    # 도서관명,시도명,시군구명,도서관유형,운영시작시각,운영종료시각,열람좌석수,자료수(도서),자료수(연속간행물),자료수(비도서)
    for line in reader:
        # key : 시도명 시군구명 도서관명
        key = line[1].strip() + '  ' + line[2].strip() + '  ' + line[0].strip()

        # 열람좌석수
        seat_num = line[6].strip()

        # 자료수
        t1, t2, t3 = line[7].strip(), line[8].strip(), line[9].strip()
        if t1 == '  ' :  t1 = 0
        if t2 == '  ' :  t2 = 0
        if t3 == '  ' :  t3 = 0
        data_num = int(t1) + int(t2) + int(t3)

        seat_counter[key] = int(seat_num)
        data_counter[key] = data_num
```

```
print('< 열람좌석수 TOP 5 >')
for data in seat_counter.most_common(5):
    # print(data)
    print(data[0], data[1])

print()

print('< 자료수 TOP 5 >')
for data in data_counter.most_common(5):
    # print(data)
    print(data[0], data[1])
```

Open API 다루기

공개 API(Open Application Programming Interface, Open API, 오픈 API)는 누구나 사용할 수 있도록 공개된 API를 말한다.
구글맵이 대표적인 예이다. 지도 서비스 및 다양한 서비스들에서 시도되고 있으며 누구나 접근하여 사용할 수 있다는 장점이 있다. 메타블로그들도 공개 API를 사용하여 만드는 것들이다.

```
# 오픈 API
# 공공데이터 포털 : http://data.go.kr
# 통계청 : http://kosis.kr/openapi
# 서울시 : http://data.seoul.go.kr
# 카카오 : https://developers.kakao.com/docs
# 네이버 : https://developers.naver.com/main
# 다음 : http://developers.daum.net/
# skplanet : https://developers.skplanetx.com/
```

OpenAPI 실습 1 - 다음 image 검색 (코드1)

```
from urllib.request import urlopen, quote
import json
from pprint import pprint
import html

apiKey = “발급받는 App Key“   # key 생성 시 웹 브라우져를 선택하기
q = "뽀로로"
output = "json"

url = "https://apis.daum.net/search/image?apikey={apiKey}&q={q}&output={output}"₩
        .format(apiKey=apiKey, q=quote(q), output=output)        # quote : 한글 인코딩 처리 (가 -> %EA%B0%80)
f = urlopen(url)
body = f.read()

s = body.decode("utf-8") # bytes -> str
s = html.unescape(s)        # html 문자 unescape 처리 (&lt;b&gt; -> <b>)

j = json.loads(s) # str -> dict
pprint(j)
```

OpenAPI 실습 1 - 다음 image 검색 (코드2)

```
{'channel': {'description': 'Daum Open API search result',
             'generator': 'Daum Open API',
             'item': [{'cp': '16CIYSC5zGTVsMKcxM',
                       'height': '198',
                       'image': 'http://i2.media.daumcdn.net/svc/image/0.jpg',
                       'link': 'http://v.media.daum.net/v/20070101181106359',
                       'pubDate': '20070101181106',
                       'thumbnail': 'https://search4.kakaocdn.net/CA5DmYg7mZt',
                       'title': '토종 펭귄 '<b>뽀로로</b>' 날다',
                       'width': '579'},
              …
```

결과 구조를 잘 확인하여
원하는 데이터를 가져오기!!

```
items = j['channel']['item']
cnt = 1
for item in items:
    print("%d. title : %s, image : %s" % (cnt, item['title'], item['image']))
    cnt += 1
```

OpenAPI 실습 2 - melon 실시간 차트 (코드1)

```
from urllib.request import urlopen, Request
import json
from pprint import pprint

apiKey = “발급받은 app 키"
page = "1"
count = "10"

url = "http://apis.skplanetx.com/melon/charts/realtime?version=1&page={page}&count={count}"₩
          .format(page=page, count=count)

req = Request(url)
req.add_header('appKey', apiKey)          # header에 appKey를 보냄

f = urlopen(req)
body = f.read()        # body
s = body.decode("utf-8")
j = json.loads(s)

pprint(j)
```

OpenAPI 실습 2 - melon 실시간 차트 (코드2)

```
{'melon': {'count': 10,
           'menuId': 54020101,
           'page': 1,
           'rankDay': '20161030',
           'rankHour': '12',
           'songs': {'song': [{'albumId': 10008861,
                               'albumName': 'TWICEcoaster : LANE 1',
                               'artists': {'artist': [{'artistId': 905701,
                                                       'artistName': 'TWICE '
                                                                     '(트와이스)'}]},
                               'currentRank': 1,
                               'pastRank': 1,
                               'playTime': 213,
                               'songId': 30054344,
                               'songName': 'TT'},
```

```
melon = j['melon']
songs = melon['songs']['song']
for song in songs:
    print('%s. %s - ' % (song['currentRank'], song['songName']), end='')
    artists = song['artists']['artist']
    for artist in artists:
        print('%s ' % artist['artistName'], end='')
    print()
```

Crawling 하기

웹 페이지를 그대로 가져와서 거기서 데이터를 추출해 내는 행위.
크롤링(Crawling) 혹은 스크래이핑(Scraping) 이라고도 한다.
Python이 이 분야의 선두주자로서, 컴퓨터 프로그래밍이 익숙하지 않은
비전공자들인 인문학이나 통계 분야의 종사자들이 쓰기 쉽도록 라이브러리 들이
발달하면서 급격히 발전하고 있다. 대표적인 파이썬 라이브러리의 예로
beautifulsoup와 Scrapy 크롤링 프레임워크 등이 있다.

Crawling 실습 1 - KBO 팀순위 (BeautifulSoup)

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

data = urlopen('http://www.koreabaseball.com/TeamRank/TeamRank.aspx')
soup = BeautifulSoup(data, "html.parser")

# 팀 순위
table = soup.find('div', {'id': 'cphContainer_cphContents_udpRecord'}).find('table')
print(table)

ths = table.find_all('th')
print(ths)
for th in ths:
    print(th.text, end='₩t')
print()
print('-' * 70)

trs = table.find_all('tr')
for tr in trs:
    tds = tr.find_all('td')
    for td in tds:
        print(td.text, end='₩t')
    print()
```

Crawling 실습 2 - 네이버 웹툰 (BeaultifulSoup)

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

# <ol id="realTimeRankFavorite" class="asideBoxRank">
# <li class="rank01">
# <a href="/webtoon/detail.nhn?titleId=626907&no=107" title="복학왕-105화 결혼식 1화">복학왕-105화 결혼식 1화</a>
# </li>
html = urlopen('http://comic.naver.com/index.nhn')
soup = BeautifulSoup(html, "html.parser")

ol = soup.find('ol', {'id': 'realTimeRankFavorite'})
items = ol.find_all('a')

rank = 1
for item in items:
    print("%d. %s" % (rank, item.text))
    rank += 1
```

Crawling Framework - Scrapy

1. Python 2.7 설치하기

2. Scrapy 설치하기

2-1) Windows

1) python 2.7 install
2) Path 환경변수 추가 (c:₩python27, c:₩python27₩Scripts)
3) install pywin32 (http://sourceforge.net/projects/pywin32)
4) pip install --upgrade pip
5) pip install wheel
6) http://www.lfd.uci.edu/~gohlke/pythonlibs/#lxml 에서 아래 whl 다운로드
7) pip install lxml-3.6.4-cp27-cp27m-win32.whl
8) pip install Scrapy

2-2) Linux, OS X

> python -m pip install --upgrade pip
> sudo pip install Scrapy

An open source and collaborative framework
for extracting the data you need from websites.
In a fast, simple, yet extensible way.

3. Scrapy Project 생성

〉 scrapy startproject 〈project명〉

예) scrapy startproject tutorial

```
tutorial/
    scrapy.cfg            # deploy configuration file

    tutorial/             # project's Python module, you'll import your code from here
        __init__.py

        items.py          # project items definition file

        pipelines.py      # project pipelines file

        settings.py       # project settings file

        spiders/          # a directory where you'll later put your spiders
            __init__.py
```

4. spider 디렉토리 밑에 Spider를 상속받은 Subclass 작성

```python
import scrapy


class QuotesSpider(scrapy.Spider):
    name = "quotes"

    def start_requests(self):
        urls = [
            'http://quotes.toscrape.com/page/1/',
            'http://quotes.toscrape.com/page/2/',
        ]
        for url in urls:
            yield scrapy.Request(url=url, callback=self.parse)

    def parse(self, response):
        page = response.url.split("/")[-2]
        filename = 'quotes-%s.html' % page
        with open(filename, 'wb') as f:
            f.write(response.body)
        self.log('Saved file %s' % filename)
```

5. Spider 실행

```
scrapy crawl quotes
```

```
... (omitted for brevity)
2016-09-20 14:48:00 [scrapy] INFO: Spider opened
2016-09-20 14:48:00 [scrapy] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)
2016-09-20 14:48:00 [scrapy] DEBUG: Telnet console listening on 127.0.0.1:6023
2016-09-20 14:48:00 [scrapy] DEBUG: Crawled (404) <GET http://quotes.toscrape.com/robots.txt> (referer:
2016-09-20 14:48:00 [scrapy] DEBUG: Crawled (200) <GET http://quotes.toscrape.com/page/1/> (referer: Nor
2016-09-20 14:48:01 [quotes] DEBUG: Saved file quotes-1.html
2016-09-20 14:48:01 [scrapy] DEBUG: Crawled (200) <GET http://quotes.toscrape.com/page/2/> (referer: Nor
2016-09-20 14:48:01 [quotes] DEBUG: Saved file quotes-2.html
2016-09-20 14:48:01 [scrapy] INFO: Closing spider (finished)
...
```

json 또는 csv 파일로 저장하기

scrapy crawl quotes -o quotes.json

scrapy crawl quotes -o quotes.csv

Crawling Framework - Scrapy 실습 1 (Quote - Xpath 사용)

```
#-*- coding: utf-8 -*-
import scrapy
class QuotesSpider(scrapy.Spider):
    # project내에서 unique id
    name = "quotes"
    start_urls = [
        'http://quotes.toscrape.com/page/1/',
        'http://quotes.toscrape.com/page/2/',
    ]

    # Xpath 선택자 사용
    def parse (self, response):
        quotes = response.xpath('//div[@class="quote"]')
        for quote in quotes:
            text = quote.xpath('span/text()').extract_first()
            author = quote.xpath('span/small[@class="author"]/text()').extract_first()
            tags = quote.xpath('div[@class="tags"]/a/text()').extract()
            # 데이터를 모아두었다가 한꺼번에 리턴
            yield {
                'title': text,
                'author': author,
                'tags': tags
            }
```

Crawling Framework - Scrapy 실습 1 (Quote - CSS 사용)

```
#-*- coding: utf-8 -*-
import scrapy
class QuotesSpider(scrapy.Spider):
    # project내에서 unique id
    name = "quotes"
    start_urls = [
        'http://quotes.toscrape.com/page/1/',
        'http://quotes.toscrape.com/page/2/',
    ]

    # CSS 선택자 사용
    def parse(self, response):
        quotes = response.css('div.quote')
        for quote in quotes:
            text = quote.css("span.text::text").extract_first()
            author = quote.css("small.author::text").extract_first()
            tags = quote.css("div.tags a.tag::text").extract()

            yield {
                'title': text,
                'author': author,
                'tags': tags
            }
```

Crawling Framework - Scrapy 실습 1 (Quote - 결과)

Quotes to Scrape

"The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking."

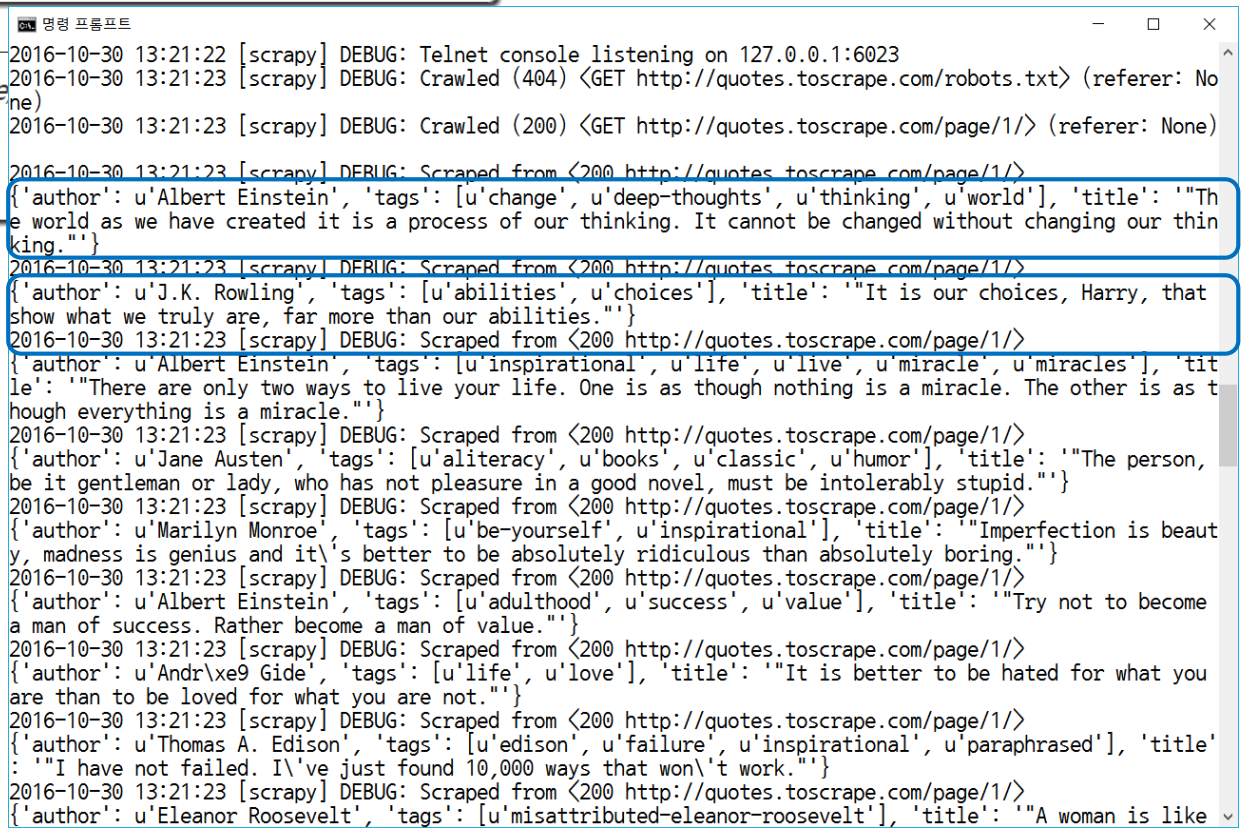
by Albert Einstein (about)

Tags: change deep-thoughts thinking world

"It is our choices, Harry, that show what we truly are

by J.K. Rowling (about)

Tags: abilities choices



Crawling Framework - Scrapy 실습 2 (Stock - item 작성)

[items.py]

```
import scrapy

class StockItem(scrapy.Item):
    # define the fields for your item here like:
    name = scrapy.Field()
    price = scrapy.Field()
    updown = scrapy.Field()
```

Crawling Framework - Scrapy 실습 2 (Stock - Xpath 사용)

```
#-*- coding: utf-8 -*-
import scrapy
from stock.items import StockItem
from unidecode import unidecode

class StockSpider(scrapy.Spider):
    name = "stock"
    start_urls = [
        'https://www.krx.co.kr/main/main.jsp'
    ]
    # xpath 이용하기
    def parse(self, response):
        indexes = response.xpath('//div[@class="intro-index"]/ul/li')
        for index in indexes:
            spans = index.xpath('span')
            name = spans[0].xpath('text()').extract_first()
            price = spans[1].xpath('text()').extract_first()
            updown = spans[2].xpath('text()').extract_first()

            item = StockItem()
            item['name'] = name
            item['price'] = price
            item['updown'] = updown

            yield item
```

Crawling Framework - Scrapy 실습 2 (Stock - CSS 사용)

```
#-*- coding: utf-8 -*-
import scrapy
from stock.items import StockItem
from unidecode import unidecode

class StockSpider(scrapy.Spider):
    name = "stock"
    start_urls = [
        'https://www.krx.co.kr/main/main.jsp'
    ]
    # css 이용하기
    def parse(self, response):
        indexes = response.css('div.intro-index ul li')
        for index in indexes:
            name = index.css("span.index-name::text").extract_first()
            price = index.css("span.index-price::text").extract_first()
            updown = index.css("span.index-up::text").extract_first()
            if not updown:
                updown = index.css("span.index-down::text").extract_first()
            item = StockItem()
            item['name'] = unidecode(name)
            item['price'] = price
            item['updown'] = unidecode(updown)

            yield item
```

Crawling Framework - Scrapy 실습 2 (Stock - 결과)

명령 프롬프트

```
2016-10-30 13:25:03 [scrapy] DEBUG: Crawled (404) <GET https://www.krx.co.kr/robots.txt> (referer: None)
2016-10-30 13:25:03 [scrapy] DEBUG: Crawled (200) <GET https://www.krx.co.kr/main/main.jsp> (referer: None
)
2016-10-30 13:25:04 [scrapy] DEBUG: Scraped from <200 https://www.krx.co.kr/main/main.jsp>
{'name': 'KTOP 30', 'price': u'6,502.95', 'updown': 'V 12.97 (0.20)'}
2016-10-30 13:25:04 [scrapy] DEBUG: Scraped from <200 https://www.krx.co.kr/main/main.jsp>
{'name': 'KOSPI', 'price': u'2,019.42', 'updown': 'V 4.70 (0.23)'}
2016-10-30 13:25:04 [scrapy] DEBUG: Scraped from <200 https://www.krx.co.kr/main/main.jsp>
{'name': 'KOSPI 200', 'price': u'256.53', 'updown': 'V 0.13 (0.05)'}
2016-10-30 13:25:04 [scrapy] DEBUG: Scraped from <200 https://www.krx.co.kr/main/main.jsp>
{'name': 'KOSDAQ', 'price': u'640.17', 'updown': 'V 8.40 (1.30)'}
2016-10-30 13:25:04 [scrapy] DEBUG: Scraped from <200 https://www.krx.co.kr/main/main.jsp>
{'name': 'KRX_caegweonjisu', 'price': u'173.06', 'updown': 'V 0.08 (0.05)'}
2016-10-30 13:25:04 [scrapy] INFO: Closing spider (finished)
2016-10-30 13:25:04 [scrapy] INFO: Dumping Scrapy stats:
{'downloader/request_bytes': 508,
 'downloader/request_count': 2,
 'downloader/request_method_count/GET': 2,
 'downloader/response_bytes': 30265,
 'downloader/response_count': 2,
 'downloader/response_status_count/200': 1,
 'downloader/response_status_count/404': 1,
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2016, 10, 30, 4, 25, 4, 180000),
 'item_scraped_count': 5,
 'log_count/DEBUG': 8,
 'log_count/INFO': 7,
 'response_received_count': 2,
 'scheduler/dequeued': 1,
 'scheduler/dequeued/memory': 1,
```