Fast campus

파이썬을 활용한 실전 웹크롤링

1주차 - 크롤링 이론과 실전 기초

2018.07.21

신준호
jjjjooonno@gmail.com

Fast campus

목차 - 이론

1. 웹 크롤링 / 웹 스크래핑에 대한 기초 개념 및 워딩 이해

2. 파이썬 코드 오류시 대처법

Fast campus

목차 - 실습

1. 파이썬의 반복문, 제어문

2. 파이썬의 기본 외장 함수 활용하기

3. 네이버 요일별 웹툰 페이지 크롤링

웹 크롤링 / 웹 스크래핑에 대한 기초개념 및 용어 이해의 목적

추후 강의 진행시 사용할 용어를 학습하고,
웹 크롤링, 웹 스크래핑의
기본적인 개념과 흐름을 학습합니다.

웹 크롤링 / 웹 스크래핑에 대한 기초 개념 및 용어 이해

컴퓨터 소프트웨어 기술을 활용하여 **웹 상의 정보**를
자동적으로 수집하여 **정형의 데이터 형식**으로 추출하는 행위

웹 크롤링 / 웹 스크래핑에 대한 기초 개념 및 용어 이해

컴퓨터 소프트웨어 기술을 활용하여

웹 상의 정보를

자동적으로 수집하여

정형의 데이터 형식으로 추출하는 행위

Fast campus

웹 크롤링 / 웹 스크래핑에 대한 기초 개념 및 용어 이해

실제 코드를 작성하는 언어로 파이썬을 선택!

파이썬 개발환경은 모두 통일하여 진행

- **개발환경 가이드 참고!**

Fast campus

웹 크롤링 / 웹 스크래핑에 대한 기초 개념 및 용어 이해

컴퓨터 소프트웨어 기술을 활용하여

웹 상의 정보를

2주차에 학습할 내용인 HTML/CSS/JS

웹이라는 문서의 내용을 표현하는 언어

웹페이지에 보이는 모든것들은 위의 세가지 언어로 표현

웹 크롤링 / 웹 스크래핑에 대한 기초 개념 및 용어 이해

Amazon Web Service

서버를 대여하여 항상 정해진 시간에 자동적으로 크롤링 프로그램을 작동시킬 수 있습니다.

자동적으로 수집하여

정형의 데이터 형식으로 추출하는 행위

웹 크롤링 / 웹 스크래핑에 대한 기초 개념 및 용어 이해

크롤링한 데이터는 csv / xlsx와 같은 포맷의 파일로 추출하거나,
혹은 데이터베이스에 저장합니다.

정형의 데이터 형식으로 추출하는 행위

파이썬 코드 오류 시 대처법 학습의 목적

과제 진행, 및 실습 시 발생하는 오류는 조교와 강사의 도움으로 해결이 가능합니다. 하지만 모든 예외와 에러를 수업시간에 해결하는 것은 불가능 하기 때문에 오류에 대한 대처법을 학습하고, 추후 강의가 끝난 후에도 오류해결을 스스로 할 수 있는 방법을 학습합니다.

파이썬 코드 오류 시 대처법

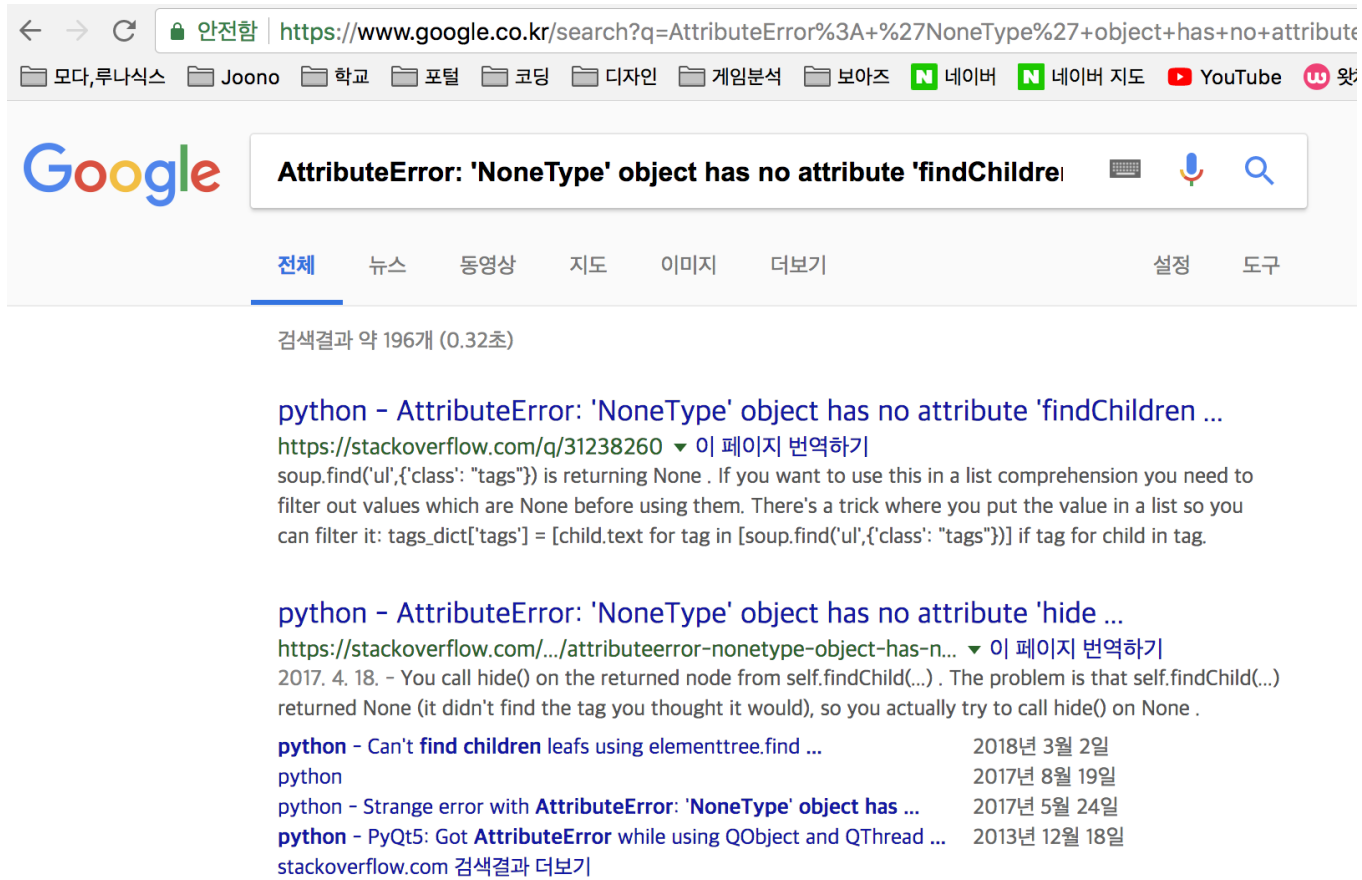
```
Traceback (most recent call last):
  File "/Users/joono/GoogleDrive/SKKU/BOAZ/패스트캠퍼스/fast_campus/코드
/test_case/exam2.py", line 51, in <module>
    rel_words_chil = rel_words_parent.findChildren()
AttributeError: 'NoneType' object has no attribute 'findChildren'
```

파이썬에서 오류가 발생되면
오류가 발생한 코드와
어떠한 에러가 발생했는지를 보여줍니다.

이때, 위의 빨간 부분을 구글에 검색해 봅니다.

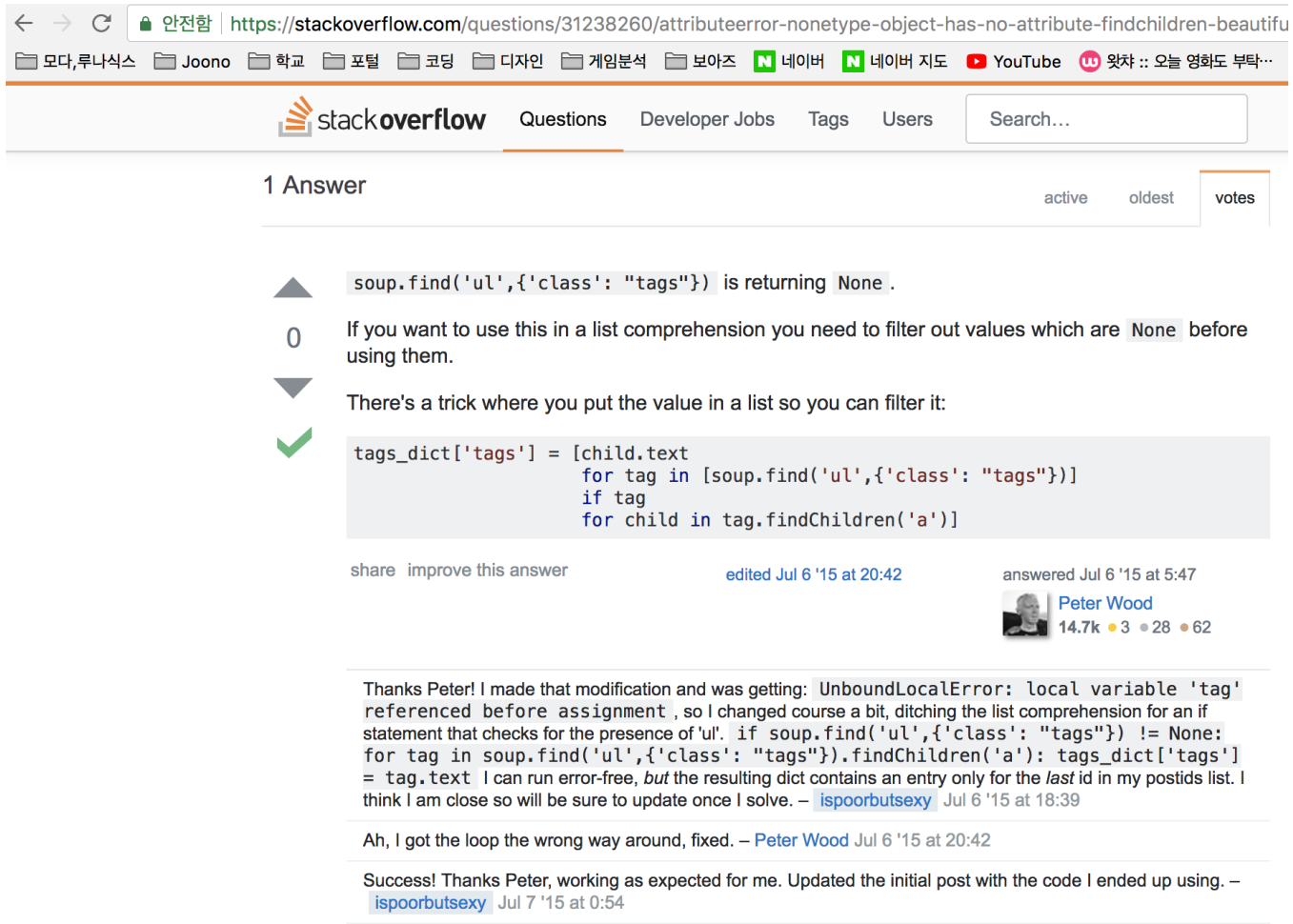
Fast campus

파이썬 코드 오류 시 대처법



Fast campus

파이썬 코드 오류 시 대처법



안전함 | https://stackoverflow.com/questions/31238260/attributeerror-nonetype-object-has-no-attribute-findchildren-beautifu

모다,루나식스 | Joono | 학교 | 포털 | 코딩 | 디자인 | 게임분석 | 보아즈 | 네이버 | 네이버 지도 | YouTube | 왓챠 :: 오늘 영화도 부탁…

stackoverflow Questions Developer Jobs Tags Users Search…

1 Answer — active oldest votes

0

`soup.find('ul',{'class': "tags"})` is returning `None`.

If you want to use this in a list comprehension you need to filter out values which are `None` before using them.

There's a trick where you put the value in a list so you can filter it:

```
tags_dict['tags'] = [child.text
                     for tag in [soup.find('ul',{'class': "tags"})]
                     if tag
                     for child in tag.findChildren('a')]
```

share improve this answer — edited Jul 6 '15 at 20:42 — answered Jul 6 '15 at 5:47 Peter Wood 14.7k ● 3 ● 28 ● 62

Thanks Peter! I made that modification and was getting: `UnboundLocalError: local variable 'tag' referenced before assignment`, so I changed course a bit, ditching the list comprehension for an if statement that checks for the presence of 'ul'. `if soup.find('ul',{'class': "tags"}) != None: for tag in soup.find('ul',{'class': "tags"}).findChildren('a'): tags_dict['tags'] = tag.text` I can run error-free, *but* the resulting dict contains an entry only for the *last* id in my postids list. I think I am close so will be sure to update once I solve. – ispoorbutsexy Jul 6 '15 at 18:39

Ah, I got the loop the wrong way around, fixed. – Peter Wood Jul 6 '15 at 20:42

Success! Thanks Peter, working as expected for me. Updated the initial post with the code I ended up using. – ispoorbutsexy Jul 7 '15 at 0:54

파이썬 코드 오류 시 대처법

대부분 오류를 검색하면 제일 상단에 Stackoverflow 사이트의 질문과 답변이 나오게 됩니다.

사실 나의 코드와 질문자의 코드가 완벽하게 같지는 않지만 적어도 오류가 나는 원인을 답변자가 상세히 설명하기 때문에 웬만한 오류는 구글링으로 해결이 가능합니다.

파이썬 제어문 반복문 학습의 목적

웹 크롤링 코드를 짤 경우, 반복적인 동작 수행과
조건에 따른 다른 동작 수행을 위한
for, while, if, else 문을 사용하는 경우가 많습니다.

이러한 코드 작성시 원활한 코드 작성을 위한
정확한 문장 사용을 위한 학습을 진행합니다.

반복문

반복문이란 원하는 코드를 원하는 만큼 자동적으로 반복하여 실행하게 하는 문장입니다. Python에서는 for문과 while문을 사용합니다.

for문은 정해진 크기의 배열의 길이만큼 반복하거나,
혹은 내가 정해주는 횟수만큼 반복하는 문장입니다.

그러나 while문은 정해진 길이가 없이 어떠한 조건을 만족할 때까지 무한히 반복이 가능한 문장입니다.

제어문

제어문이란 조건에 따라 각기 다른 코드를 실행하고자 할 때 사용합니다.

조건은 여러개로 설정이 가능합니다.

또한 반복문 안에 조건문을 넣어서 조건에 맞는 경우
다음 반복으로 강제로 이동시키는 (continue)와,

조건에 맞는 경우 그대로 코드를 진행시키는
명시만 해주는 (pass) 를 사용 가능합니다.

반복문 실습 코드

```
list_ = [‘a',1,3,5,7]
# 단순한 숫자 범위에서의 for문
for number in range(0,5):
    print(number)
# 리스트 안의 내용을 하나씩 사용하는 for문
for item in list_:
    print(item)
# enumerate를 사용하여 index와 내용물 둘 다를 사용하는 for문
for index, item in enumerate(list_):
    print(str(index+1)+'th item is '+str(item))

# n이 10보다 작은 경우에만 반복하고 n이 10 이상이 되는 순간 멈추게 되는 while문
n = 0
while n < 10:
    print(n)
    n = n + 1
```

제어문 실습 코드

```
# 7이라는 숫자를 n이라는 변수에 할당해 놓습니다.
n = 7
# n이 5보다 큰지 작거나 같은지 비교하는 조건문입니다.
if n > 5:
        print('n is bigger than 5')
else:
        print('n is smaller than or equal to 5')
# for문과 if문을 함께 쓴 코드입니다. 들여쓰기로 구분하는 부분을 유심히 봐주시기 바랍니다.
for number in range(0,10):
        if (number >= 0) and (number < 4):
                print(number,'  ','0 <= number < 4')
        elif (number >= 4) and (number < 7):
                print(number,'  ','4 <= number < 7')
        else:
                print(number,'  ','7 <= number')
# for문과 if문을 함께 써서, 조건을 만족한다면 바로 다음 루프로 넘어가게 한 코드입니다.
for number in range(10,20):
        if number > 15:
                continue
        print(number,'  ','number <= 15')
```

파이썬 외장함수 학습의 목적

파이썬에는 다양한 외장함수가 존재합니다.
이러한 내장함수는 디렉토리(폴더) 및 파일명 검색, 시스템 경로,
시간과 관련한 함수, 집합(리스트 혹은 튜플)에 대한 함수,
문자열에 대한 정규표현식 등이 있습니다.

크롤링 코드에 많이 사용하는 외장함수를 위주로 학습합니다.

os 모듈

os 모듈은 현재 디렉토리(폴더)의 경로를 얻을수도 있고, 또한 어떠한 경로(폴더)에 어떠한 파일이 있는지를 구할 수 있는 모듈입니다.

import os

current_directory = os.getcwd()
print(current_directory) --> **'현재 프로젝트의 경로명'**

file_list = os.listdir('무슨 파일들이 있는지 알고싶은 경로')
print(file_list) --> **['파일명1','파일명2',...]**

os.chdir("바꾸고 싶은 현재 경로") -- **바꾸고 싶은 경로로 현재 디렉토리가 변경됩니다.**

time 모듈

time 모듈은 코드가 실행되는 시간을 뽑거나, 혹은 코드의 실행 중 쉬는시간을 주고 싶을 때 사용합니다.
time.time() -- 현재 시간
--> 컴퓨터가 인식하는 현재 시간을 뽑아줍니다.
time.localtime(time.time())
--> 사람이 인식 할 수 있는 모양으로 현재 시간을 뽑아줍니다.
time.strftime('**출력할 형식 포맷 코드**', time.localtime(time.time()))
--> 문자열로 현재 시간을 원하는 포맷으로 뽑아줍니다.

포맷 코드 예시
%a - Mon과 같은 요일 줄임말 / %b Jan과 같은 월 줄임말 / %m 숫자 달
%H 24시간 단위 시간 / %I 12시간 단위 시간 / %j 365 기준 일수 / %d 31 기준 일수
%Y 연도출력 / %Z 시간 기준 출력(대한민국 등)

collections 모듈

collections는 집합(리스트, 딕셔너리 등등)에 좀더 다양한 기능을 추가하기 위한 모듈입니다.

크롤링을 할때는 collections중에서도 Counter를 많이 쓰게 됩니다.

Counter는 집합에서의 원소 개수를 카운트하여 딕셔너리 형태로 반환해 줍니다.
from collections import Counter
list_ct = ['a','b','c','d','d','d','c','a','f']
apb_count = Counter(list_ct) --> {'a' : 2, 'b' : 1, 'c' : 2, 'd' : 3, 'f' : 1}

re 모듈

re는 regular expression의 약자로 정규표현식을 사용하는 모듈입니다.
정규표현식이란 특정한 규칙을 가진 문자열의 집합을 표현하는 데 사용하는 **형식 언어**입니다.

Python에서 문자열은 ""와 '' 사이에 넣은 것으로 구분합니다.

이러한 문자열에서 특정한 **패턴**을 만족하는 문자열만을 **추출**하거나,
찾기 위해 정규표현식을 사용합니다.

크롤링으로 우리가 원하는 정보를 찾아낼 때, 정규표현식은 좀더 다양하고, 효율적인 코드작성을 가능하게 합니다.

re 모듈

정규표현식은 아래와 같은 구조를 가지고 있습니다.

'[정규표현식]정규표현식'

따옴표안에 []를 집어넣고, [] **안과 밖**에 정규표현식을 작성하여 우리가 원하는 문자의 패턴을 인식시키는 것입니다.

다음장 부터는 예시를 보며 학습을 하겠습니다.

re 모듈

‘[가나다]’

위의 문자열은 가나다중에 한 문자라도 목표 텍스트에 들어있으면 매치하라는 패턴입니다.

위 패턴은 **가지, 나비, 다람쥐**와는 매치가 되겠지만, **라디오**와는 매치가 되지 않습니다.

또한 [a-z]는 모든 영어 소문자와 매치가 된다는 의미입니다.
대문자는 [A-Z], 숫자는 [0-9]를 사용합니다.
한글은 [ㄱ-ㅎ]으로 자음을, [가-힣]으로 모든 한글을 매칭시킬 수 있습니다.

re 모듈

패턴	문자열	매치 여부
'[가나다]'	가지	O
'[가나다]'	나비	O
'[가나다]'	라디오	X
'[가-힣]'	sam	X
'[가-힣]'	훈민정음뷁	O
'[가-힣]'	sam 오취리	X / O(뒤의 슬라이드에서 설명)

re 모듈

메타 문자란 패턴을 짜기 위해 사용하는 문자들입니다.
원래의 문자열이 아닌 특수한 용도를 가지고 있습니다.

‘.’
.은 모든 문자열(엔터(/n) 제외)에 매칭이 되는 메타문자입니다.
그러나 ‘[.]’은 문자열 .에 매칭을 시키는 패턴입니다.
[]안에 들어가는 문자들은 그 문자 그대로의 문자들이니 주의하셔야 합니다.

‘문자열*’
은 반복을 뜻합니다. 만약 ‘라’라는 패턴이 있다면 이것은 ‘ ’, ‘라’, ‘라라’, ‘라라라’ 등에 매칭됩니다.

‘문자열+’
+은 위의 *과 같은 반복을 뜻하지만 *와는 다르게 1번 반복부터만 패턴으로 인식합니다.

‘문자열{start,end}’
위의 패턴에서 start와 end에는 숫자가 들어갑니다. 이 숫자는 앞의 문자가 start번부터 end번까지 반복되는 것을 특정하여 찾는 표현입니다.

re 모듈

패턴	문자열	매치 여부
'... 대리'	김철수 대리	O
'... 대리'	김리 대리	X
'... 대리'	윤 인성 대리	X
'... 대리'	sam 대리	O
'.+대리'	김철수 대리	O
'.+대리'	김리 대리	O
'.+대리'	윤 인성 대리	O
'.+대리'	sam 대리	O
[가-힣]+ 대리	김철수 대리	O
[가-힣]+ 대리	김리 대리	O
[가-힣]+ 대리	윤 인성 대리	X
[가-힣]+ 대리	sam 대리	X
[가-힣]{2,3} 대리	김철수 대리	O
[가-힣]{2,3} 대리	김리 대리	O
[가-힣]{2,3} 대리	윤 인성 대리	X
[가-힣]{2,3} 대리	sam 대리	X

re 모듈

‘|’
|는 or의 의미 입니다. ‘가|나’는 가 혹은 나가 들어있으면 매칭이 된다는 의미입니다.

‘^’
^는 문자열의 맨 처음과 매칭이 된다는 의미입니다. ‘^패스트’는 ‘패스트캠퍼스’와는 매칭되지만

’나는 패스트캠퍼스 …’와는 매칭이 되지 않습니다.

‘$’
$는 ^와 반대로 문자열의 맨 끝과 매칭이 된다는 의미입니다.

‘/b’
단어 구분자 입니다. r’/b패스트/b’를 사용한다면 문장내에서 ‘… 패스트 … ’만을 매칭하고,
‘… 패스트캠퍼스 … ’와 같이 **다른 단어속에 존재한다면 매칭하지 않습니다.**

re 모듈

패턴	문자열	매치 여부(match / search)
'^http://|^https://'	http://www.naver.com	O / O
'^http://|^https://'	https://www.xdfkjer.co.kr	O / O
'^http://|^https://'	www.op.gg	X / X
'^http://|^https://'	www.slkdjf.com	X / X
'com$'	http://www.naver.com	X / O
'com$'	https://www.xdfkjer.co.kr	X / X
'com$'	www.op.gg	X / X
'com$'	www.slkdjf.com	X / O
'.*com$'	http://www.naver.com	O / O
'.*com$'	https://www.xdfkjer.co.kr	X / X
'.*com$'	www.op.gg	X / X
'.*com$'	www.slkdjf.com	O / O

re 모듈

정규표현식을 사용하려면 패턴을 변수에 저장해야 한다.

Pattern = re.compile(‘a*[가-힣]’)

위 코드는 Pattern이라는 변수에 a가 반복되고, 그 후에 한글이 나오는 패턴을 저장한다는 뜻이다.
이렇게 저장된 패턴에는 아래의 4가지 함수를 사용할 수 있다.

match(), search() 문자열에서 패턴에 맞는 것을 찾는 함수, 그러나 match는 문자열의 시작부터,
Search는 전체를 모두 살펴본다는 데에 차이가 있다.

findall(), finditer() 패턴에 맞는 문자열을 여러개 찾는다는데에는 공통점이 있지만,
findall은 결과값이 리스트로, finditer는 결과값이 매칭결과인 파이썬 변수의 형태를
반복적으로 제공해준다는 차이점이 있다.

re 모듈

앞에서 살펴본 match와 search는 모두 매칭된 상태를 return해줍니다.

이 return된 매칭을 살펴보기 위한 함수는 아래와 같다.

group()
매칭된 문자열을 보여준다.

start() /end()
매칭된 문자열의 **위치를 인덱스로 알려준다.**

span()
시작과 끝의 인덱스를 **튜플(시작, 끝)**로 알려준다.

re 모듈

우리가 찾을 패턴이 여러개일 수 있다. 가령 이름과 핸드폰번호를 찾는 아래의 패턴이 있다고 해보자.

‘[가-힣]+\s[0-9]{3}[-][0-9]{4}[-][0-9]{4}’

우리는 이 패턴을 사용해서 **이름과 전화번호를 따로** 뽑고 싶을 수 있다.

‘**(**[가-힣]+**)**\s**(**[0-9]{3}[-][0-9]{4}[-][0-9]{4}**)**’

그럴때는 위와 같이 ()를 사용하여 그룹을 나눠주면

매칭에 **.group(1)을 사용하면 이름**, **.group(2)를 사용하면 전화번호**가 나오게 된다.

정규표현식 테스트

https://regex101.com/#pcre

위 사이트에서 우리는 정규표현식을 **바로바로 확인하면서 표현식을 구**성할 수 있습니다.

re 모듈 QUIZ

앞에서 살펴본 example.txt 파일에서 발표 제목을 가져오는 정규표현식을 작성해 봅니다.

Hint

S22.
양방향 고온
수전해-연료
전지 발전 기술
(우상국/KIER)

G5.
바이오
세라믹스
(이진형/KICET)

문자열 관련 기본 내장 함수

Python은 문자열을 다루기에 가장 좋은 언어로 알려져있습니다.

그중 가장 큰 이유는 **문자열끼리 더하기가 되고,**
숫자와 곱하기가 된다는 점입니다.

Ex) ‘신’ + ‘준호’ = ‘신준호’ / ‘신’ * 3 = ‘신신신’

또한 **문자열에 index**가 들어가 있어, 문자열을 자르기가 매우 쉽습니다.

Ex) ‘패스트캠퍼스’[0:3] = ‘패스트’

문자열 관련 기본 내장 함수

문자열.strip()

양쪽의 공백을 없에주는 함수입니다.

문자열.replace(a,b)

문자열 안의 a를 b로 치환해 주는 함수입니다.

문자열.split(a)

문자열을 a를 기준으로 쪼개서 리스트에 담아 반환해주는 함수입니다.

문자열 관련 기본 내장 함수

문자열에 변수를 삽입하는 포매팅도 매우 편리합니다.

우리가 사용하는 python 3.7에서는 포매팅을 아래와 같이 사용합니다.

Name = ‘신준호’
Country = ‘대한민국’

f’{Name} 은 {Country} 사람입니다.’

이 때 만약 넣는 변수가 숫자 혹은 리스트, 딕셔너리라면 그 변수에 사용되는 함수까지도
{} 안에 들어갈 수 있습니다.

Ex)
Name = ['신준호','조준희']

f'{Name[0]} 강사와 {Name[1]} 조교'

네이버 요일별 웹툰 페이지 크롤링 실습의 목적

실제 크롤링 코드를 살펴보고 어떤 프로세스로 웹페이지에서 크롤링 코드로 나아갈 수 있는지 알아봅니다.

또한 파이썬 내장함수, 제어, 반복문이 어떻게 사용되는지 간략하게 살펴볼 수 있습니다.

문자열 관련 기본 내장 함수

```
import requests
from bs4 import BeautifulSoup
import re


days = ['mon','tue','wed','thu','fri','sat','sun']
day = days[0]

url = 'http://comic.naver.com/webtoon/weekdayList.nhn?week=wed'+day+'&view=list&order=ViewCount'

response = requests.get(url)

html = response.text

soup = BeautifulSoup(html,'html.parser')
title_pattern = re.compile('^/webtoon/list.nhn\?titleId=')
titles = soup.find_all('a',attrs = {'href' : title_pattern})
title_text = []
for title in titles:
   title_text.append(title.text.strip())

while '전체보기' in title_text: title_text.remove('전체보기')
while '' in title_text: title_text.remove('')
while 'NEW' in title_text: title_text.remove('NEW')

print(title_text)
```

감사합니다.