[평생] 멀티미디어 일반형 양식

[원고] 빅데이터 수집 시스템 개발

13회차 : 정규식 활용과 문자열

내용전문가	백현숫	교수설계(한기대)	홍길동
협력업체	업체명	교수설계(협력업체)	홍길동

NCS 분류 정보	20-01-02-09	정보통신 - 정보기술 - 정보기술개발 - 빅데이터플랫폼구축
능력단위 정보	03	빅데이터 수집시스템 개발
능력단위 요소 정보	2001020903_17v1.1	빅데이터 수집시스템 설계하기

업무	작성자	버전	작성일	특이사항
원고 작성	백현숙	v.1.0	2019/09/11	2회차 원고
한기대 피드백		v.1.1		
원고 보완		v.2.0		
자문 진행		v.2.1		
원고 보완				

◈ 학습 목차

- 학습 목차를 작성해 주세요. (NCS 비적용 과정일 경우에는 NCS 및 능력단위 정보를 기입하지 않아도 됩니다.)

NCS 분류 정보	20-01-02-09	정보통신 - 정보기술 - 정보기술개발 - 빅데이터플랫폼구축
능력단위 정보	능력단위코드 기입	빅데이터 수집시스템 개발

모듈명	회차명	능력단위 요소		수행준거	
1. 모듈명 기입	01. 회차명 기입	1	능력단위요소명 기입	1.1	수행준거 기입
	02. 회차명 기입	1	능력단위요소명 기입	1.2	수행준거 기입
	03. 회차명 기입	1	능력단위요소명 기입	1.3 1.4	수행준거 기입 수행준거 기입
	04. 회차명 기입	1	능력단위요소명 기입	1.4	수행준거 기입
2. 모듈명 기입	05. 회차명 기입	2	능력단위요소명 기입	2.1 2.2	수행준거 기입 수행준거 기입

과정명		모듈명		회차명	

◈ 학습열기

정규식은 특정 패턴을 만족하는 데이터를 검출하거나, 입력된 데이터가 정상적 형태인 데이터인지 확인하기 위해 사용합니다. 정규식은 파이썬에만 존재하는것이 아니고, 다른 언어들에서도 많이 사용합니다. 리눅스나 유닉스의 스크립트 프로그램에서도 정규식을 사용하고 자바스크립트 같은 언어들도 정규식을 사용합니다. 웹사이트 가입시 보통 전화번호나 이메일, 사업자 번호 등을 입력하는데 사이트를 이용하는 사람들 중 컴퓨터에 익숙하지 않은 사람들 또는 제대로 테스트 안될 거라 생각하고 대충 입력하거나 잘못된 데이터로 사이트를 무력화 시키려는 경우 등 여러가지 이유로 잘못된 값이 들어 오는 것을 막기위해 사용합니다. 그리고 크롤링한 자료들로 부터 특정 패턴에 맞는 값들을 추출하고자 할 때, 로그파일에서 특정 패턴을 유지하는 값들을 찾아서 분석하고자 할때 많이 사용합니다. 정규식은 많이 사용하는데 비해서 의외로 만들기가 쉽지는 않습니다. 이 장에서는 정규식을 만들고, 정규식을 이용하여 데이터 검증을 하거나 데이터를 추출하는 방법등을 배우도록 하겠습니다

내레이션

학습자가 학습 화면에 구성한 내용을 보고 있다고 가정하고, 학습 설명으로 함께 들어야 할 음성 내용을 자세하게 기입해 주세요. → 이건 나래이션해요? 저도 이런건 나래이션하면 좋겠어요..ㅜㅜ

과정명		모듈명		회차명	

◈ 학습내용

- 정규식 활용과 문자열

◈ 학습목표

- 입력된 데이터값을 검증할 수 있는 정규식을 만들 수 있다
- 정규식을 이용하여 원하는 형식의 데이터를 추출할 수 있다

내레이션

(기입하지 않습니다.)

간지 부분

◈ 정규식 활용

정규식을 활용하는 함수들

내레이션

(기입하지 않습니다.)

◈ 정규표현식

- 정규 표현식(Regular Expression)은 특정한 규칙을 가진 문자열의 패턴을 표현하는 데 사용하는 표현식(Expression)으로 텍스트에서 특정 문자열을 검색하거나 특정문자열을 치환할 때 사용 됩니다.
- 사용자가 우편번호나 전화번호, 이메일 주소를 맞는 형식으로 입력했는지 확인하거나 크롤링해온 자료에서 이메일이나 전화번호등을 추출하거나, 로그파일에서 특정 에러메시지가 들어간 라인들을 찾을 때 정규 표현식을 사용합니다.
- 정규 표현식은 간단히 정규식, Regex 으로 부르기도 합니다
- 정규표현식은 파이썬 뿐만 아니라, 다른 언어(자바스크립트, 리눅스 스크립트등)들에서 많이 사용되고 있습니다.

내레이션

◈ 정규표현식 사용방법

- 파이썬에서는 정규식을 지원하기 위해 re 라는 모듈을 사용합니다.
- 검색하고자 하는 내용의 패턴을 만듭니다. 패턴에는 이스케이프 문자를 사용하여 기술하는 경우가 있으므로 이스케이프 본래의 기능을 무력화 시키기 위해 패턴 문자열 앞에 r을 붙입니다.
- re 모듈에 패턴을 부여하고 compile 시킨 후 문장에 적용하거나, compile 시키지 않고 바로 적용 가능합니다.
- match, search, filndall 등 정규표현식 함수를 이용해 원하는 데이터를 검증하거나 추출합니다.

```
import re

pattern = r'비'
text = "하늘에 비가 오고 있습니다.  어제도 비가 왔고 오늘도 비가 오고 있습니다"
regex = re.compile(pattern) #패턴을 컴파일 시킨다
result = regex.findall(text)  #matiching 이 이루어진 모든 문자열의 리스트를 반환합니다
print( result )
```

내레이션

◈ 우편번호 형식 맞추기 정규식 예제

- 우리나라의 우편번호는 6자리에서 5자리로 체계가 바뀌었습니다. 정수 값을 입력을 받아서 이 데이터가 우편번호 형식에 맞는지 확인해보는 예제를 만들어 보겠습니다.
- 우편번호 패턴 방식 : \d{5}$ <- 정수 5개만 가능하다

- 파일명 : exam13_1.py

```
import re

zipcode = input("우편번호를 입력하세요")
pattern = r ' ₩d{5}$'
regex = re.compile(pattern)
result = regex.match(zipcode)
if result != None:
    print("형식이 일치합니다.")
else:
    print("잘못된 형식입니다.")
```

정수 5자리만 입력이 가능합니다. 5자리의 정수를 입력하면 “형식이 일치합니다.”로 출력되고 그밖에
문자를 넣거나, 자릿수가 부족하거나 넘치면 “잘못된 형식입니다”라고 출력됩니다

내레이션

◈ 정규식 함수

match(pattern, string)
문자열의 시작부분부터 매칭이 되는지 검색한다.
매칭후 매칭에 대한 정보를 저장한 객체를 반환한다

search(pattern, string)
문자열에 패턴과 매칭되는 곳이 있는지 검색한다
매칭후 매칭에 대한 정보를 저장한 객체를 반환한다

findall(pattern, string)
정규식과 매치되는 모든 문자열을 리스트로 반환한다

finditer(pattern, string)
정규식과 매치되는 모든 문자열을 iterator 객체(매칭객체)로 반환한다

sub(pattern, replace, string, count=0, flag=0)
정규식과 매치되는 모든 문자열을 대체문자열로 교체하고, 결과를 str 타입으로 반환한다.

내레이션

◈ match 함수

- 형식 : match(pattern, string, flag=0)

- 문자열의 시작 부분부터 패턴과 매칭이 되는지 검색하는 함수입니다.

- 검색 결과는 matchObject 인스턴트로 반환됩니다. 만일 일치하는 패턴이 없을 경우 None을 반환합니다.

- MatchObject 주요 멤버함수
  - group() : 매칭된 문자열을 반환. 0, 1, 2 파라미터를 주면 해당 문자열을 반환.
  - 패턴이 ()등을 이용해 그룹으로 되어 있을 경우 자동으로 분리
  - start() : 매칭된 문자열 시작 위치.
  - end() : 매칭된 문자열 종료 위치.
  - span() : 매칭된 문자열 시작과 종료위치를 튜플로 반환.

:

내레이션

◈ match 함수 예제

```
import re

text1 = "I like star"
text2 = "star is beautiful"

pattern = "star"
print (re.match( pattern, text1))
print (re.match( pattern, text2))
matchObj = re.match( pattern, text2)
print(matchObj.group() )
print(matchObj.start() )
print(matchObj.end() )
print(matchObj.span() )
```

match 함수는 첫부분에 star가 와야 한다. 이 문장에서 패턴을 못 찾아냄

그룹함수를 통해 단어 추출, 첫번째 패턴의 시작위치밀 단어의 종료위치, 단어위치값을 튜플로 나타냄

```
None
<re.Match object; span=(0, 4), match='star'>
star
0
4
(0, 4)
```

◈search 함수

- 형식 : search(pattern, string, flag=0)
 - 문자열에 패턴과 매칭 되는 부분이 있는지 검색하는 함수입니다.
- match() 함수와의 차이점은 match 함수가 문자열의 시작부분이 일치해야 하는데 반해 search 함수는 문자열의 시작부터가 아니라 중간부터 있더라도 검색을 진행합니다.
- 일치하는 패턴이 여러개 있더라도 맨 처음에 검색된 부분을 반환합니다.

MatchObject 주요 멤버함수

- group() : 매칭된 문자열을 반환, 0, 1, 2 파라미터를 주면 해당 문자열을 반환.
- 패턴이 ()등을 이용해 그룹으로 되어 있을 경우 자동으로 분리
- start() : 매칭된 문자열 시작 위치.
- end() : 매칭된 문자열 종료 위치.
- span() : 매칭된 문자열 시작과 종료위치를 튜플로 반환.

:

내레이션

◈ search 함수 예제

- 파일명 : exam13_3.py

```
import re

text1 = "I like star, red star, yellow star"
text2 = "star is beautiful"

pattern = "star"
print (re.search( pattern, text1))
print (re.search( pattern, text2))

matchObj = re.search( pattern, text1)
print(matchObj.group() )
print(matchObj.start() )
print(matchObj.end() )
print(matchObj.span() )

matchObj = re.search( pattern, text2)
print(matchObj.group() )
print(matchObj.start() )
print(matchObj.end() )
print(matchObj.span() )
```

match 함수는 첫부분에 star가 와야 한다.
search함수는 이 문장에서 패턴을 찾아낸다

그룹함수를 통해 단어 추출, 첫번째 패턴의 시작위치밀 단어의 종료위치, 단어위치값을 튜플로

```
<re.Match object; span=(7, 11), match='star'>
<re.Match object; span=(0, 4), match='star'>
star
7
11
(7, 11)
star
0
4
(0, 4)
```

내레이션

◈findall 함수

- 형식 : findall(pattern, string, flag=0)
- 문자열에 패턴과 매칭 되는 부분에 대하여 string 리스트로 반환합니다.
- 반환값은 일치하는 문자열들의 리스트입니다.
- 특정한 패턴과 일치하는 문자열만 추출할때 사용합니다.
- 예)전화번호만 추출하거나 이메일만 추출하고자 할때

전화번호 체크 패턴 `r"\d{3}-\d{4}-\d{4}“`

이메일 체크 패턴 `r"\b[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}\b"`

내레이션

◈ findall 함수 예제

- 파일명 : exam13_4.py

```
import re
#전화번호만 추출하기
text = """
    phone : 010-0000-0000 email:test1@nate.com
    phone : 010-1111-1111 email:test2@naver.com
    phone : 010-2222-2222 email:test3@gmail.com
    """
print()
print("--- 전화번호 추출하기 ---")
phonepattern = r"\d{3}-\d{4}-\d{4}"

matchObj = re.findall( phonepattern, text)

for item in matchObj:
    print( item)
print("--- 이메일 추출하기 ---")
emailpattern = r"\b[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}\b"

matchObj = re.findall( emailpattern, text)
for item in matchObj:
    print( item)
print()
```

text에 기술된 전화번호와 이메일로 부터 원하는 이메일과 전화번호를 추출하는 예제입니다.
findall 함수는 원하는 패턴의 데이터를 str의 list 형태로 반환합니다.

```
--- 전화번호 추출하기 ---
010-0000-0000
010-1111-1111
010-2222-2222
--- 이메일 추출하기 ---
test1@nate.com
test2@naver.com
test3@gmail.com
```

\b가 앞에 있을때는 뒤에 오는 패턴으로 시작하는이라는 의미이고, \b가 뒤쪽에 있을때는 앞의 패턴으로 끝나라는 의미입니다.
이메일의 경우 test123@hanmail.net 형태인데 앞쪽에는 영문자와 숫자가 하나이상 올 수 있고 @와 도메인 그리고 .(도트) 뒤에 영문자가 옵니다.
2!4자리정도

내레이션

◈finditer 함수

- 형식 : finditer(pattern, string, flag=0)
- 문자열에 패턴과 매칭 되는 부분에 대하여 매칭객체를 리스트로 반환합니다.
- 반환값은 문자열이 아니라 매칭 객체라는 부분에서 find 함수와 차이가 납니다
- 특정한 패턴과 일치하는 문자열만 추출할때 사용합니다.
- 예)전화번호만 추출하거나 이메일만 추출하고자 할때

MatchObject 주요 멤버함수

- group() : 매칭된 문자열을 반환, 0, 1, 2 파라미터를 주면 해당 문자열을 반환.
- 패턴이 ()등을 이용해 그룹으로 되어 있을 경우 자동으로 분리
- start() : 매칭된 문자열 시작 위치.
- end() : 매칭된 문자열 종료 위치.
- span() : 매칭된 문자열 시작과 종료위치를 튜플로 반환.

내레이션

:

◈ findall 함수 예제

- 파일명 : exam13_5.py

```
import re
#전화번호만 추출하기
text = """
    phone : 010-0000-0000 email:test1@nate.com
    phone : 010-1111-1111 email:test2@naver.com
    phone : 010-2222-2222 email:test3@gmail.com
    """
print()
print("--- 전화번호 추출하기 ---")
phonepattern = r"\d{3}-\d{4}-\d{4}"
matchObj = re.finditer( phonepattern, text)
for item in matchObj:
    print( item.group() )
    print( item.span() )
print()
```

```
--- 전화번호 추출하기 ---
010-0000-0000
(13, 26)
010-1111-1111
(60, 73)
010-2222-2222
(108, 121)
```

finditer 함수는 패턴과 일치할경우 MatchObject 형태의 객체 반복자를 반환합니다. 이터레이터 형태이므로 for 구문을 통해 일치한 대상에 대한 정보들을 확인할 수 있습니다.

내레이션

◈sub 함수

- **형식 : sub(pattern, replace, string, count=0, flag=0)**
  - replace는 문자열이 될 수도 있고, 함수가 될 수도 있다
  - count는 최대 몇 번까지 교체할 것인가를 설정하는 인자이다. 이 값이 0 이면, 모두 교채, 0보다 크면 지정된 횟수만큼만 교체된다

- 문자열에 패턴과 매칭 되는 부분에 대하여 매칭객체를 리스트로 반환합니다.
- 반환값은 문자열이 아니라 매칭 객체라는 부분에서 find 함수와 차이가 납니다
- 특정한 패턴과 일치하는 문자열만 추출할때 사용합니다.
- 예)전화번호만 추출하거나 이메일만 추출하고자 할때

MatchObject 주요 멤버함수
- group() : 매칭된 문자열을 반환, 0, 1, 2 파라미터를 주면 해당 문자열을 반환.
- 패턴이 ()등을 이용해 그룹으로 되어 있을 경우 자동으로 분리
- start() : 매칭된 문자열 시작 위치.
- end() : 매칭된 문자열 종료 위치.
- span() : 매칭된 문자열 시작과 종료위치를 튜플로 반환.

내레이션

◈ sub 함수 예제
- 파일명 : exam13_6.py

```
import re

text1 = "I like stars, red star, yellow star"

print()
pattern = "star"
result = re.sub( pattern, "moon", text1)
print(result)

result2 = re.sub( pattern, "moon", text1, count=2)
print(result2)
```

```
I like moons, red moon, yellow moon
I like moons, red moon, yellow star
```

내레이션

간지 부분

◈ 정규식 만들기

다양한 정규식 패턴 만들기

내레이션

(기입하지 않습니다.)

◈정규식 패턴 표현

패턴 : ^
설명 : 시작이 이 패턴으로 이루어져야 한다
예제 : 패턴이 ^abc 라면 문자열의 시작이 abc이어야 한다
abcde, abc, abc123 다 된다

패턴 : $
설명 : 이 패턴으로 끝나야 한다
예제 : 패턴이 abc$라면 문자열의 끝은 abc여야 한다, match 함수는 안된다.
abc, dabc, 123abc 다 된다

패턴 : [문자들]
설명 : []에 속한 문자들만 해당된다 가능한 문자열의 집합을 의미한다
예제 :패턴이 [pIP]ython 라면
python, Python 은 가능하나 , PYTHON은 안된다.
[A-Z] 첫글자가 알파벳 대문자만 가능하다
KOREA-O, korea-X, Korea-O

내레이션

◈정규식 패턴 표현

패턴 : [^문자들]
설명 : 피해야할 문자들의 집합이다. 이 문자들로 구성된 단어만 피한다
예제: 패턴이 [^abc] 라면
a, b, c, abc, acb, bca - x, d-o

패턴 : |
설명 : or 연산 둘중 하나만 일치하면 된다.
예제) 패턴이 [kK]orea라면
korea, Korea - o, Corea - x

패턴 : ?, +, *
설명 : ? 앞의 패턴이 없거나 하나이어야 한다
+ 앞의 패턴이 하나 이상 있어야 한다
* 앞의 패턴이 0개 이상이어야 함, 반복을 의미
예제 : \d? - 숫자가 없거나 하나만 있어야 한다
\d+ - 숫자가 하나 이상이어야 함
\d* - 숫자가 없거나 하나 이상이어야 함

내레이션

◈정규식 패턴 표현

패턴 : 패턴{n}
설명 : 패턴이 n번 반복해서 나타나야 한다
예제) \d{3}
  숫자가 3개 나타나야 함

패턴 : 패턴{n, m}
설명 : 패턴이 최소 n번 최대 m번으로 나타나야 한다
예제) \d{3-4}
숫자가 최소 3번에서 최대 4번 나타나야 한다

패턴 : \d(숫자), \w(문자), \s(화이트스페이스), .(newline을 제외한 모든 문자)
\b(단어의 시작이나 끝이어야 한다, r'\bain' :ain으로 시작할때, r'ain\b':ain으로 끝날때)

내레이션

◈자주 사용하는 정규식 패턴

전화번호 : r"₩d{3}-₩d{3,4}-₩d{4}“ r은 문자열안에 사용된 ₩(이스케이프) 문자의 본래 기능을 뺏기 위해서 사용된다.

이메일 : r"₩b[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+₩.[a-zA-Z]{2,4}₩b“

우편번호 : r”₩d{5,6}”

내레이션

◈정규식 패턴 표현예제

파일명 : exam13_7.py

```
import re

pattern = r"^abc"
text = ["abc", "abcd", "abc15", "dabc", "", "s"]
repattern = re.compile(pattern)

for item in text:
    result = repattern.search(item)
    if result:
        print(item, "- O" )
    else:
        print(item, "- X" )
```

:

내레이션

◈정규식 패턴 표현예제

파일명 : exam13_8.py

```
import re

pattern = r"abc$"
text = ["abc", "dabcd", "asdabc", "d12abc", "", "s"]
repattern = re.compile(pattern)

for item in text:
    result = repattern.search(item)
        if result:
        print(item, "- O" )
    else:
        print(item, "- X" )
```

내레이션

◈정규식 패턴 표현예제

파일명 : exam13_9.py

```
import re

pattern = r"[p|P]ython"
text = ["python", "Python", "PYTHON"]
repattern = re.compile(pattern)

for item in text:
    result = repattern.search(item)
    if result:
        print(item, "- O" )
    else:
        print(item, "- X" )

pattern = r"[A-Z]"
text = ["python", "Python", "PYTHON", "korea", "KOREA", "Korea"]
repattern = re.compile(pattern)
for item in text:
    result = repattern.search(item)
    if result:
        print(item, "- O" )
    else:
        print(item, "- X" )
```

내레이션

:

◈정규식 패턴 표현예제

파일명 : exam13_10.py

```
import re

patterns=[r"₩d?", r"₩d+", r"₩d*"]
text = ["abc", "1abc", "12abc", "123", "aa12ab"]

for pattern in patterns:
    resultList=[]
    for item in text:
        result = re.search(pattern, item)
        if result == None:
            resultList.append(item+"-X")
        else:
            resultList.append(item+"-O")

    print(resultList)
```

:

내레이션

◈정규식 그룹화

- 정규 표현식에서 ( ) 괄호는 그룹을 의미한다.

- 표현식을 그룹화 하여 지정하는 경우에는 매칭 후 값을 출력하고자 할때 사용합니다.

- 전화번호의 경우 패턴을 지정할때 010-9000-8000 의 형태라면 정수3자리-정수4자리-정수4자리 형태라 패턴은 ₩d{3}-₩d{4}-₩d{4}가 됩니다. 이렇게 매칭을 하면 전화번호의 맨앞자리 010, 011 등과 국과 번호를 구분하려면 다시 split 과 같은 함수를 이용해 데이터를 나누어야 하는데 이럴때 정규식 그룹화를 사용합니다.

- 정규식 그룹화를 이용해 (₩d{3})-(₩d{4})-(₩d{4}) 형태로 괄호를 해주면 세개의 그룹으로 묶이게 됩니다.
매칭된 결과를 MatchObject 로 반환받는 함수들의 경우에 MatchObject 에 있는 group 라는 함수를 통해 각각의 대한 그룹 접근이 가능합니다. (₩d{3})-(₩d{4})-(₩d{4}) 의 경우 group(0)에는 전체 번호가 , group(1), group(2), group(3)를 통해 값을 얻을 수 있습니다.

- 우편번호일 경우 (₩d{3})-(₩d{2,3}) 패턴을 사용합니다.

내레이션

◈

- 파일명 : exam13_11.py

```
import re

contents = "문의사항이 있으면 010-1234-6789 으로 연락주시기 바랍니다."
pattern = r'(₩d{3})-(₩d{4})-(₩d{4})'
regex = re.compile(pattern)
result = regex.search(contents)
if result != None:
    phone1 = result.group(1)
    phone2 = result.group(2)
    phone3 = result.group(3)
    print(phone1)
    print(phone2)
    print(phone3)
else:
    print("전화번호가 없습니다.")
```

전화번호 패턴을 괄호를 이용해 그룹화 한다
3개의 구룹으로 나누어졌으므로 group(1), group(2), group(3)로 각각의 번호를 추출한다

```
010
1234
6789
```

내레이션

◈ 적용하기

사업자 번호는 앞3자리 가운데 2자리 뒤 5자리로 구성되어 있습니다. 문서에서 사업자 번호만 추출하여 그중 개인 사업자 관련 번호만 추출하고자 합니다. 사업자 번호의 각 자릿수의 의미는 다음과 같습니다.

사업자번호의 의미 : 000-00-00000
앞 3자리 - 관할세무서 번호
가운데 2자리 - 사업자의 성격을 나타냄(개인사업자 : 90~99)
마지막 5자리 - 사업자용4자리+검증용1자리
다음 데이터들로 부터 사업자 번호들을 추출하고 그 중에 개인 면세 사업자에 해당하는 사업자 번호만 출력하기 바랍니다

```
contents = """
    우리커피숍 100-90-12345
    영풍문고 101-91-12121
    영미청과 102-92-23451
    황금코인 103-89-13579
    우리문구 104-91-24689
    옆집회사 105-82-12345
"""
```

내레이션

(기입하지 않습니다.)

◈ 적용하기- 풀이

```
import re

contents = """
    우리커피숍 100-90-12345
    영풍문고 101-91-12121
    영미청과 102-92-23451
    황금코인 103-89-13579
    우리문구 104-91-24689
    옆집회사 105-82-12345
"""
pattern = r'(₩d{3})-(₩d{2})-(₩d{5})'
regex = re.compile(pattern)
result = regex.finditer(contents)
print()
for item in result:
    if int(item.group(2))>=90 and int(item.group(2))<=99:
        print( item.group() )

print()
```

```
100-90-12345
101-91-12121
102-92-23451
104-91-24689
```

데이터를 분리해야 하므로 그룹을 이용해서 패턴을 지정함
r'(\d{3})-(\d{2})-(\d{5})'

모든 요소를 분리해야 하므로 finditer 함수를 이용해서 분리하였음
반환값인 MatchObject 객체의 group함수를 이용해서 데이터를 추출함
0번에는 사업자 번호가 전체가 들어있고 1번에는 앞 3자리 2번에는
가운데 2자리, 3에는 나머지 값이 있다.
group(2)를 추출하여 90~99 사이의 값인지 확인해야 하므로 str값을 int
형으로 전환하여 비교하여 그 사이에 존재하면 출력함

내레이션

(기입하지 않습니다.)

◈ 문제풀기

문제 1) 다음 정규식과 관련하여 수행 결과를 예측하시오

```
import re

str = "The rain in Spain"
x = re.search("ain$",  str)
if x != None:
    print("O")
else:
    print("X")
```

정답) O

해설) $는 /로 끝내는 이라는 의미이므로 The rain 단어에서 ain으로 끝나는 단어가 매치하므로 결과는 O입니다.

난이도) 3 (1:아주어려움, 2:어려움 3: 보통 4: 쉬움 5: 아주 쉬움)

관련페이지번호) 12

내레이션

(기입하지 않습니다.)

번호	문제	정답	난이도	해설	관련학습보기
2	**다음 정규식과 관련하여 수행 결과를 예측하시오** **str = "The rain in Spain"** **x = re.match("ain$", str)** **print(x)** **if x != None:** ** print("O")** **else:** ** print("X")**	X	2	match 함수는 단어의 시작부분만 매칭 판단을 하기때문에 이 정규식에 대해서 매칭되는 단어를 검색하지 못합니다	14
3	**str = "The rain in Spain"** **x1 = re.findall("ain$", str)** **x2 = re.findall("ain", str)** **print(len(x1), len(x2))** **위 코드의 수행 결과는 ?l** **(1) 1,1 (2)2,2, (3)1,2 (4) 2,1**	3	1	findall 함수는 패턴과 일치하는 모든 문자셋을 찾아낸다. 이때 패턴에 $가 있을 경우 문장의 맨끝을 마무리 하는 경우에만 해당된다. ain 패턴은 두개의 던어 rain과 Spain을 매칭했다고 찾아내지만 ain$는 문장 맨 끝의 Spain만 일치한다고 찾아낸다 그래서 1, 2 이다	14
4	**다음처럼 그룹을 지정했을 경우 패턴이 매칭 되었을 경우 group(0)에 들어간 값은 ? 패턴 (\d{3})-(\d{4})-(\d{4})** **문자열 : 내전화 010-1111-2222** **① 010-1111-2222** **② 010** **③ 1111** **④ 2222**	1	4	정규화 그룹을 사용할 경우에 그룹의 0번에는 전체 데이터가 1번부터 그룹내의 내용이 분할하여 들어간다. 그래서 0번에는 전체 번호인 010-1111-2222가 다들어가있다	29

번호	문제	정답	난이도	해설	관련학습 보기
5	아래 코드는 모든 공백문자를 9로 바꾸려고 합니다. 적당한 함수는? str = "The rain in Spain" x =re.( )("\s", "9", str) print(x) ① match ② search ③ findall ④ sub	4	5	매칭되어있는 데이터를 찾어서 대체문자열로 교환하는 함수는 sub 함수입니다	18
6	**패턴이 다음과 같을때 매칭되는 단어가 아닌것은?** **패**턴 : r"^abc" ① abc ② abcd ③ abc123 ④ dabc	4	5	단어의 시작이 abc 하고 했으므로 마지막 4번은 d로 시작했음로 매칭되지 않는다	25
7	**패턴이 다음과 같을때 매칭되는 단어가 아닌것은?** **패**턴 : r"abc$" ① abc ② abcd ③ 123abc ④ dabc	2	5	단어의 마지막이 abc로 끝나야 한다	26

번호	문제	정답	난이도	해설	관련학습보기
8	패턴이 다음과 같을 때 올바른 것은 ? 패턴 : r"\d*" ["abc", "1abc", "12abc“] ① “abc”는 매칭되지 않는다 ② "1abc“ 는 매칭되지 않는다 ③ “12abc”만 매칭된다. ④ 모두다 매칭된다	4	1	* 는 반복되는 패턴이 없어도 된다. 숫자를 요구하지만 *를 추가 했으므로 숫자 패턴이 있던 없던 매칭된다.	28
9	패턴이 다음과 같을 때 매칭이 되는것을 모두 선택한것은? 패턴 : r"\d{5,6}" [“12345”, “a1234”, “123456”, “12222222”, “1234”, “1abc”] ① “12345”, “a1234” ② “12345”, “123456”’ ③ “12345”, “123456”, “1234” ④ “12345”, “123456”, “1abc”	2	5	r'\d{5,6} 정수 5자리, 6자리만 해당된다.	29
10	**사원번호가 다음과 같은 형태일때 올바른 패턴 수식을 작성하시오** **정규화 그룹을 이용하시오** **12-34-999**	해설참조	1	(\d{2})-(\d{2})-(\d{3}) 각 단위별로 괄호를 해줘야 합니다.	29

과정명		모듈명		회차명	

◈ 핵심요약

▶정규식 이란

- 정규식은 문자열이 특정 유형을 갖고 있을때 각 문자열의 규칙성을 특정한 문자형태로 작성하여 문자열을 검색하거나 추출하고자 할때, 입력된 값의 형태가 정해진 규칙에 맞는지 검색하기 위해 폭넓게 사용하는 방식입니다.

- re 모듈을 이용해서 패턴과 매치된 데이터를 찾습니다. 패턴을 컴파일하여 컴파일된 객체를 이용하여 데이터를 검색할 수 도 있고, 컴파일 하지 않고 re모듈을 통하여 직접 패턴을 찾을 수 있습니다.

- 파이썬에서는 match, search, findall, finditer, sub 함수등을 제공합니다. 패턴이 일치하면 일치하는 MatchObject 객체를 주로 반환하는데 이 객체에는 매칭되는 데이터의 위치값이나 데이터 등에 대한 정보가 있습니다.

▶정규식 패턴

- 정규식을 만들기 위해서는 패턴을 만들어야 합니다. 패턴은 여러가지 문자들의 조합으로 이루어집니다. 시작을 영대문자만 오게 하고 싶다면 ^[A-Z] 라는 패턴을 만들어야 합니다. [], {}, () 등으로 원하는 패턴을 만들어서 사용합니다.

내레이션

(기입하지 않습니다.)

◈ 학습맺음

- 이번 회차에서는 정규식과 정규식을 다루는 함수에 대해서 살펴보았습니다.
  수고 많으셨습니다.

◈ 참고자료

- https://docs.anaconda.com/anaconda/
- https://code.visualstudio.com/docs/getstarted/themes
- http://egloos.zum.com/sweeper/v/3065126
- http://pythonstudy.xyz/python/article/401-%EC%A0%95%EA%B7%9C-%ED%91%9C%ED%98%84%EC%8B%9D-Regex
- https://wikidocs.net/21703

내레이션

(기입하지 않습니다.)

◈ 회차 메타데이터

- 주요학습내용 : 해당 회차 또는 레슨의 주요학습내용을 자세히 기입해 주세요.
- 검색 키워드 : 학습자가 검색창에 어떤 검색어를 입력하면 본 회차 또는 본 레슨이 검색될 수 있을지 검색 키워드를 5개 기입해 주세요.

제목	주요학습내용	검색 키워드1	검색 키워드2	검색 키워드3	검색 키워드4	검색 키워드5
13. **정규식 활용과 문자열**	정규식	regex	match	findall	sub	search