ch.7 데이터 주무르기

문장열 부분만

참고자료: https://whatisthenext.tistory.com/116

문자열 포매팅

- 옛날 스타일: c에서 하던거처럼 %s, %d 이렇게써주고 ,%('aa'',1)이렇게 포매팅해줌 → 포매팅해주는 문자의 타입을 다 알고 있어야 한다. →넘 귀찮쓰
- '{}' . format(1)

```
#{}안에 index로 접근 가능
print('{2} {0} {1}'.format(1,4.27,'aa'))
#output
aa 1 4.27
```

```
#{}안에 변수 이름으로 접근 가능
print('{s} {d} {f}'.format(d=1,f=4.27,s='aa'))
#outputaa 1 4.27
```

{ : } : 타입지정 가능, 인자에 이름 정하기 가능, 최소 필드길이/최대 문자 길이/ 정렬 등 지원

정규표현식

- 파이썬에서는 정규 표현식을 지원하기 위해 re(regular expression)모듈을 제공한다
- import re 를 통해 모듈을 임포트
- p = re.compile('정규표현식')을 통해 정규표현식을 컴파일하고, 컴파일된 패턴 객체(p)를 이용하여 이 객체가 가지고 있는 메서드를 통해 작업을 수행할 수 있다.

1. 컴파일 후 매칭

```
import re # 정규표현식 모듈

p = re.compile('[a-z]+') # re 내장모듈 내(.) compile 메서드를 사용.
                         # compile 메서드는 "패턴 객체"를 반환한다.

m = p.match("python")    # 패턴 객체(p)에는 또다시 검색 메서드가 있다.
```

2. 축약형

```
m = re.match('정규표현식', 문자열 소스)
```

- diff : 컴파일을 활용하는 경우, 한 줄 더 써야하지만 패턴 객체(p)를 만들고 나서 여러 번 재사용 가능

  반복적인 매칭 작업이 필요한 경우에는 미리 컴파일해서 시간을 단축할 수 있다.
- patterm: 정규식을 컴파일한 결과
- 패턴 객체: 정규식을 컴파일한 결과를 담은 변수 ; p

첫번째 일치하는 패턴 찾기: search()

```
#'Young Frankenstein' 소스 문자열에서 'Frank'패턴을 찾기 위해 search를 사용한다
m = re.search('Frank',source)
if m:
        print(m.group()) #Frank
```

일치하는 모든 패턴 찾기: findall()

```
#'n'이 몇개 있는지 알려면?
m = re.findall('n',source)
print(m) # 리스트
print(m(len(m)) #4
```

패턴으로 나누기: split()

```
m = re.split('n', source)
m
```

```
#output
['You', 'g Fra', 'ke', 'stei', '']
```

일치하는 패턴 대체하기: sub()

```
m = re.sub('n', '?', source)
m #sub는 문자열을 반환 -> n자리에 ? 넣음
```

메타문자

메타 문자	설명
[ ]	문자 클래스
.	\n을 제외한 모든 문자와 매치 (점 하나는 글자 하나를 의미)
*	0회 이상 반복 (업어도 상관 없음)
+	1회 이상 반복 (무조건 한번 이상 등장해야 함)
{m, n}	m회 이상 n회 이하
|	or 조건식을 의미
^	문자열의 시작 의미
$	문자열의 끝을 의미
?	0회 이상 1회 이하
\	이스케이프, 또는 메타 문자를 일반 문자로 인식하게 한다
( )	그룹핑, 추출할 패턴을 지정한다.