02. 데이터 분석 실습

학습 목표

- 파이썬의 자료형, 기본 문법을 활용하여 데이터를 분석 할 수 있다.
- 파이썬의 정규 표현식을 활용하여 데이터 를 분석 할 수 있다.
- 실제 데이터를 직접 분석하고 활용할 수 있다.

간단한 텍스트 데이터 분석하기

- 일부 데이터 추출

. 실제 데이터에서 원하는 데이터만 출력하기

> 데이터의 구조를 파악한 뒤 패턴 을 찾음

> 불필요한 데이터를 하나씩 제거함

. 1,2,3,4,5 에서 콤마 를 제거한 12345 데이터 를 추출 하기

```
In [3]: a = "1,2,3,4,5"
        print(a)
```

```
1,2,3,4,5
```

> 반복문, 조건문을 활용

```
In [3]: a = "1,2,3,4,5"
        print(a)
```

```
1,2,3,4,5
```

```
In [4]: for i in a:
            if i != ',' :
                print(i,end="")
```

```
12345
```

> 문자열 함수를 활용한 특정 문자열제거

```
In [5]: print(a.replace(',',''))
```

```
12345
```

> 문자열 자료형의 범위 선택 연산 활용

* a[::2] 처음부터 두칸씩 건너뛰면서 표현

```
In [6]: print(a[::2])
```

```
12345
```

패턴이 있는 여러줄의 텍스트 데이터 분석하기

- 복잡한 데이터 에서 이름 만 추출 하기

```
In [ ]: b = '''
김범수: 1234, Kimbum1223
나얼: 2233, UI2244
이수: 3333, leesujjang
박효신: 4444, somol2
'''
```

. 데이터 구조 파악
> 이름뒤에 콜론(:) 이 있음
> 구별되는 데이터 는 행으로 처리 되며 엔터 로 구분되어 있음
> 데이터 간에 공백이 있음
> 가장 앞쪽에 위치하고 있음

. 데이터 좌우 공백 제거하기
> .strip() 함수를 이용하여 공백을 제거

```
In [8]: b = b.strip()
print(b)
```

```
김범수: 1234, Kimbum1223
나얼: 2233, UI2244
이수: 3333, leesujjang
박효신: 4444, somol2
```

. 패턴을 찾아 데이터 를 작게 분리 하기
> .split() 함수로 데이터 행 ('₩n') 별로 나누어 List 에 등록

```
In [9]: b = b.split('₩n')
print(b)
```

```
['김범수: 1234, Kimbum1223', '나얼: 2233, UI2244', '이수: 3333, leesujjang', '박효신: 4444, somol2']
```

. 반복문을 활용하여 분리한 데이터에서 원하는 데이터 추출

```
In [30]: for i in b:
    i = i.split(':')
    i = i[0].strip()
    print(i)
```

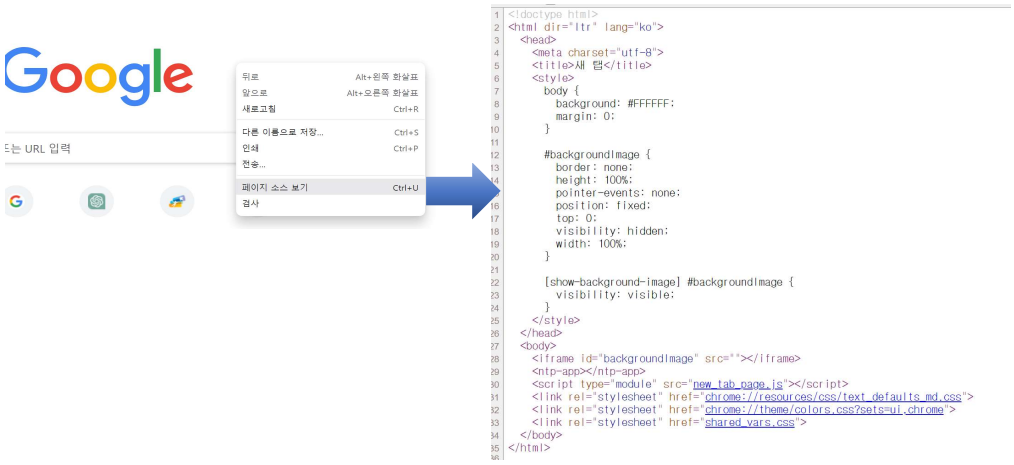
> b 의 요소 만큼 반복
> 추출한 요소의 : 기준으로 자료형 나누기 '김범수: 1234, Kimbum1223'
> 첫번째 데이터의 공백 없애기

```
김범수
나얼
이수
박효신
```

홈페이지 소스와 같이 복잡한 텍스트 데이터 분석 하기

- 인터넷 의 페이지 소스 발췌 하기

. 브라우저 에서 오른쪽 마우스 클릭 후 소스 보기



- 복잡하게 구성 되어있는 웹 페이지의 내용 중 사이트의 주소 만 추출하고 싶을때

```
<div class="select_div">
    <span data-i18n="footer.relationSites">연관사이트</span>
    <ul>


    </ul>
</div>
<ul class="list_sns">
    <li><a href="https://www.facebook.com/ekoreatech" target="_blank"><img src="/res/lms/img/common/btn_footer_sns1.png" alt="페이스북"></a></li>
    <li><a href="https://blog.naver.com/e-koreatech" target="_blank"><img src="/res/lms/img/common/btn_footer_sns2.png" alt="블로그"></a></li>
    <li><a href="https://www.youtube.com/user/koreatecholei" target="_blank"><img src="/res/lms/img/common/btn_footer_sns3.png" alt="유트브"></a></li>
</ul>
```

. 데이터 구조 파악하기

> 추출할 데이터 는 list_sns 라는 텍스트 아래에 위치함

> 연관 사이트 는 li 태그 로 구분되어 있음

> 각 연관 사이트 이름은 alt, 링크는 href 태그의 큰따옴표 " " 안에 있음

. 데이터 작게 분리하기

> find() 함수로 특정 위치의 index 확인 후 나머지 모든 부분 가져오기

```
In [32]: print(c.find('list_sns'))
```

```
119
```

```
In [33]: c = c[119:]
         print(c)
```

```
list_sns">
    <li><a href="http://www.facebook.com/SmartEducation" target="_blank"><img src="/res/ims/common/btn_footer_sns1.png" alt="페이스북"
</a></li>
    <li><a href="http://blog.naver.com/Samer-edu" target="_blank"><img src="/res/ims/common/btn_footer_sns2.png" alt="네이버 블로그"</a
></li>
    <li><a href="http://www.facebook.com/SmartEducation" target="_blank"><img src="/res/ims/common/btn_footer_sns3.png" alt="유투브"</a
></li>
</ul>
```

. 데이터 작게 분리하기

> 위의 결과에서 '
' 로 문단 나누기

```
In [34]: c = c.split('</li>')
         print(c)
```

```
['list_sns">₩n    <li><a href="http://www.facebook.com/SmartEducation" target="_blank"><img src="/res/ims/common/btn_footer_sns1.png" a
lt="페이스북"</a>', '₩n    <li><a href="http://blog.naver.com/Samer-edu" target="_blank"><img src="/res/ims/common/btn_footer_sns2.png"
alt="네이버 블로그"</a>', '₩n    <li><a href="http://www.facebook.com/SmartEducation" target="_blank"><img src="/res/ims/common/btn_foo
ter_sns3.png" alt="유투브"</a>', '₩n</ul>']
```

. 일부 데이터에서 원하는 데이터 추출 하기

> 첫번째 데이터에서 URL 추출하기

* 생성 된 c 리스트 에서 첫번째 요소 추출

```
In [35]: print(c[0])
```

```
list_sns">
    <li><a href="http://www.facebook.com/SmartEducation" target="_blank"><img src="/res/ims/common/btn_footer_sns1.png" alt="페이스북"
</a>
```

* target 문자열을 기준으로 나누어 생성된 list 의 첫번째 데이터를 url 변수에 담기

```
In [36]: url = c[0].split('" target')[0]
         print(url)
```

```
list_sns">
    <li><a href="http://www.facebook.com/SmartEducation
```

* 결과를 href= 문자열을 기준을 나누어 url 변수에 담은 후 출력

```
In [37]: url = url.split('href="')[1]
         print(url)
```

```
http://www.facebook.com/SmartEducation
```

> 첫번째 데이터에서 이름을 같이 추출하기

```
In [51]: url = c[0].split('" target')[0]
         print(url)
         print('----------------------------------------------------------------------')
         name = c[0].split('alt="')[1]
         print(name)
```

```
list_sns">
    <li><a href="http://www.facebook.com/SmartEducation
----------------------------------------------------------------------
페이스북"</a>
```

```
In [52]: url = url.split('href="')[1]
         print(url)
         print('----------------------------------------------------------------------')
         name = name[:name.find('"')]
         print(name)
```

```
http://www.facebook.com/SmartEducation
----------------------------------------------------------------------
페이스북
```

. 반복문을 사용하여 모든 데이터의 내용을 추출하기

```
for i in range(len(c)-1):
    url = c[i].split('" target')[0]
    url = url.split('href="')[1]
    print(url)
    name = c[i].split('alt="')[1]
    name = name[:name.find('"')]
    print(name)
    print('----------------------------------------------------------------------')
```

```
http://www.facebook.com/SmartEducation
페이스북
----------------------------------------------------------------------
http://blog.naver.com/Samer-edu
네이버 블로그
----------------------------------------------------------------------
http://www.facebook.com/SmartEducation
유투브
----------------------------------------------------------------------
```

정규 표현식을 활용한 데이터 분석

- 정규 표현식이란 ?

. 특정한 규칙을 가진 문자열을 표현하기 위해 사용하는 형식
. 주로 문자열의 검색 및 치환에 활용
. 다양한 문법을 제공함

- 정규표현식의 예시

. **.** : 1개 문자 와 일치
. **[ ]** : [,] 사이의 문자중 하나 선택
  > [abc]d : ab,bd,cd 포함
  > [a-zA-Z] : 알파벳 모두 포함
  > [0-9] : 숫자 를 모두 포함
. ***** : 0 개 이상의 문자 포함
  > a*b : b,aab,ab,aaaaab 등
. **{m,n}** : m회 이상 n 회 이하
  > a{1,3}b -> ab,aab,aaab 포함 (a 의 횟수 를 1 ~ 3개 를 표현 가능함)
. 그 외
  > **?**, **+**, **()**, **^** 등의 다양한 규칙이 존재함

- 파이썬 표준 모듈 re 를 활용한 정규 표현식의 활용

. 콤마를 제거한 데이터만 추출하기.

```
In [1]: import re

In [2]: a = "1,2,3,4,5"

In [4]: pattern = re.compile('[^0-9]')

In [5]: result = re.sub(pattern, '',a)

In [6]: print(result)
```

12345

> compile() 함수를 사용하여 정규 표현식 문법을 적용함.
  * 숫자가 아닌것을 모두 포함하라
> sub() : 원본 a 문자열에서 pattern 과 일치하는 (숫자가 아닌값) 을 '' 로 교체

. 한글이 아닌것은 모두 지우기

```
In [7]: import re

In [8]: b = '''
김범수: 1234, Kimbum1223
나얼: 2233, UI2244
이수: 3333, leesujjang
박효신: 4444, somol2
'''

In [9]: pattern = re.compile('[^가-힣]')
result = re.sub(pattern,'',b)
print(result)
```

김범수나얼이수박효신

> 한글이 아닌것을 pattern 에 담기
  * 가 ~ 힣 : 한글을 모두 포함하는 범위를 나타낸다.
> 원본 b 에서 패턴에 해당하는 모든 문자는 '' 으로 대체

. 웹 페이지 소스 에서 연관 SNS 사이트 주소 만 추출하기
  > search() : 특정 패턴을 찾아주는 함수
  > group() : 일치한 문자들을 반환하는 함수

```
c = '''<div class="select_div">
    <span data-i18n="footer.relationDites">연관사이트</span>
    <ul>

    </ul>
</div>
<ul class="list_sns">
    <li><a href="http://www.facebook.com/SmartEducation/" target="_blank"><img src="/res/ims/common/btn footer sns1.png" alt="페이스북"></
    <li><a href="http://blog.naver.com/Samart-edu/" target="_blank"><img src="/res/ims/common/btn footer sns2.png" alt="네이버 블로그"></a
    <li><a href="http://www.facebook.com/SmartEducation/" target="_blank"><img src="/res/ims/common/btn footer sns3.png" alt="유투브"></a>
</ul>'''
```

* url 주소를 포함하는 웹페이지

> url 주소 추출하기

```
import re

pattern = re.compile('http://.*/"')
result = pattern.search(c)
print(result.group())

pattern = re.compile('[a-zA-Z:/.]*')
result = pattern.search(result.group())
print(result.group())
```

```
http://www.facebook.com/SmartEducation/"
http://www.facebook.com/SmartEducation/
```

> 'http// .* /" target"'
  . http// 로 시작 하여 /" target 으로 끝나는 문자열
> searh() 함수 를 사용하여 찾기
>group() 으로 찾아낸 데이터 표현하기
>대소문자 알파벳 (a-z 및 A-Z),
  콜론 (:), 슬래시 (/), 점 (.) 만 표현

실습

실습 : 아래의 데이터 에서 주민등록 번호 를 * 로 변경 (마스킹) 하여 표현 하세요

- eX) text = '나의 주민등록번호는 940101-1234567입니다.'

나의 주민등록번호는 ******-*******입니다.

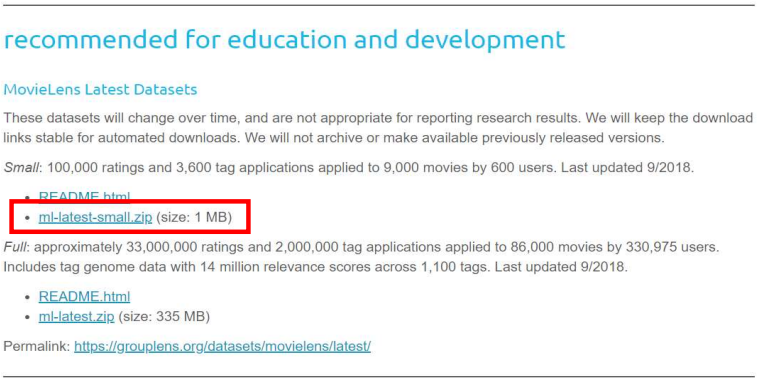
영화 장르 구분하기

- GroupLens

. 미네소타 대학의 GroupLense 라는 연구 프로젝트에서 제공하는 영화장르를 분석하여 모아놓은 데이터 자료집

> https://grouplens.org/datasets/movielens/

> 사이트 접속 후 ml-latest-small.zip 다운로드


recommended for education and development

MovieLens Latest Datasets

These datasets will change over time, and are not appropriate for reporting research results. We will keep the download links stable for automated downloads. We will not archive or make available previously released versions.

Small: 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users. Last updated 9/2018.

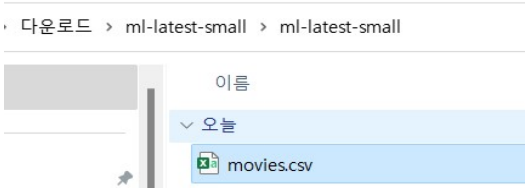
- README.html
- ml-latest-small.zip (size: 1 MB)

Full: approximately 33,000,000 ratings and 2,000,000 tag applications applied to 86,000 movies by 330,975 users. Includes tag genome data with 14 million relevance scores across 1,100 tags. Last updated 9/2018.

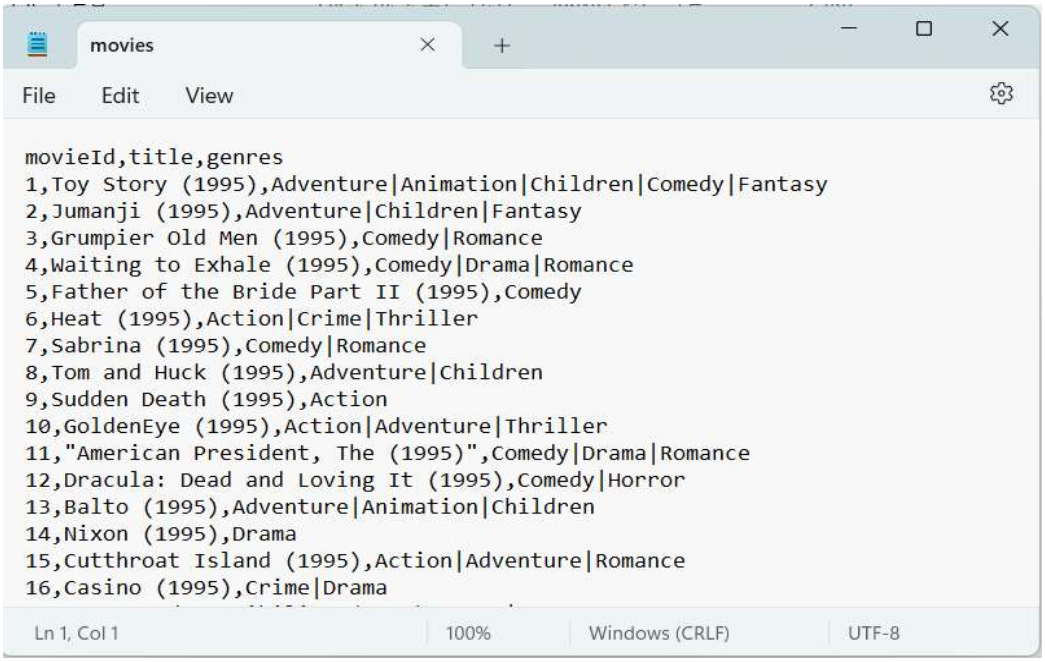
- README.html
- ml-latest.zip (size: 335 MB)

Permalink: https://grouplens.org/datasets/movielens/latest/

. 압축을 푼 다음 movies.csv 파일 살펴 보기



. 데이터 구조 파악 하기



> CSV 파일 값이 콤마(,) 로 구분되어 있음
> 하나의 영화에 여러 개의 장르가 있을 수 있음
> 영화 순번, 제목, 장르 로 구분되어 있음
> 각 영화의 장르가 2개 이상인 경우 ' | (버티컬) ' 로 구분 되어 있음
> 영화 제목마다 뒤에 괄호로 영화의 연도가 적혀 있음

- 영화 아이디, 제목, 장르 확인하기

. CSV 모듈을 통해 영화 CSV 파일 로드
> 영화아이디, 제목, 장르 가 순서대로 저장된 것을 확인 할 수 있음

```
In [3]: import csv

In [4]: f = open('grouplense_sampledata/ml-latest-small/movies.csv','r',encoding = 'utf-8')

In [5]: data = csv.reader(f)

In [7]: print(next(data))
        print(next(data))
        print(next(data))
        print(next(data))
        print(next(data))

['1', 'Toy Story (1995)', 'Adventure|Animation|Children|Comedy|Fantasy;']
['2', 'Jumanji (1995)', 'Adventure|Children|Fantasy;']
['3', 'Grumpier Old Men (1995)', 'Comedy|Romance;']
['4', 'Waiting to Exhale (1995)', 'Comedy|Drama|Romance;']
['5', 'Father of the Bride Part II (1995)', 'Comedy;']
```

>csv 파일이 있는 위치 설정

>csv.reader() 함수로 csv 파일을 읽어 data 에 할당
>next() 함수로 데이터를 하나씩 추출함

. 새로운 리스트에 각 영화의 장르 분리
> slpit() 를 이용하여 2번째 index 에 있는 장르를 '|' 를 기준으로 데이터 분리 및 저장
* extend() : 기존의 리스트 에 데이터 split 으로 나눈 리스트 를 더한다.

```
In [95]: import csv
         f = open('grouplense_sampledata/ml-latest-small/movies.csv','r',encoding = 'utf-8')
         data = csv.reader(f)
         next(data)
         result = []

In [96]: for i in data:
             result.extend(i[2].split('|'))
         print(len(result))
         print(result)

160
['Adventure', 'Children', 'Fantasy', 'Comedy', 'Romance', 'Comedy', 'Drama', 'Romance', 'Comedy', 'Action', 'Crime', 'Thriller', 'Comed
y', 'Romance', 'Adventure', 'Children', 'Action', 'Action', 'Adventure', 'Thriller', 'Comedy', 'Horror', 'Adventure', 'Animation', 'Chi
ldren', 'Drama', 'Action', 'Adventure', 'Romance', 'Crime', 'Drama', 'Drama', 'Romance', 'Comedy', 'Comedy', 'Action', 'Comedy', 'Crim
```

- 중복 된 장르 명칭을 제외 하고 조회하기

. 집합 자료형으로 데이터 변환하기 (집합 자료형 set 은 중복을 허용하지 않는 자료형이다)

> list -> set -> list 의 패턴

In [97]:
```
import csv
f = open('grouplense_sampledata/ml-latest-small/movies.csv','r',encoding = 'utf-8')
data = csv.reader(f)
next(data)
result = []
for i in data:
    result.extend(i[2].split('|'))

genre = list(set(result))
print(len(genre))
print(genre)
```

```
16
['Horror', 'Sci-Fi', 'Fantasy', 'Adventure', 'Crime', 'Thriller', 'War', 'Musical', 'Documentary', 'Drama', 'Mystery', 'Action', 'Comed
y', 'Romance', 'Children', 'Animation']
```

- 장르 별 영화의 개수 확인

. 영화마다 해당 장르가 포함되는 경우 장르 + 1

. 딕셔너리 자료형 활용 { }

> 해당 장르가 딕셔너리에 없는경우 + 1

> 해당 당르가 딕셔너리에 있는경우 + 1

In [99]:
```
import csv
f = open('grouplense_sampledata/ml-latest-small/movies.csv','r',encoding = 'utf-8')
data = csv.reader(f)
next(data)
result = []
for i in data:
    result.extend(i[2].split('|'))
```

In [100]:
```
count = {}
for i in result:
    if count.get(i):
        count[i] += 1
    else:
        count[i] = 1
print(count)
```

```
{'Adventure': 8, 'Children': 8, 'Fantasy': 2, 'Comedy': 26, 'Romance': 19, 'Drama': 36, 'Action': 15, 'Crime': 11, 'Thriller': 17, 'Hor
ror': 5, 'Animation': 2, 'Mystery': 3, 'Sci-Fi': 4, 'War': 1, 'Musical': 1, 'Documentary': 2}
```

실습

1. 위의 예제에서 장르 가 가장 다양한 영화를 찾아 보세요 (최대 장르 개수 가진 영화 모두 구하기)

Rubber 10

2. 연도별 가장 많이 개봉된 영화의 장르 를 찾아 보세요

1995 년도는 Drama 장르 가 123 회로 가장많이 상영 되었습니다.
1994 년도는 Drama 장르 가 114 회로 가장많이 상영 되었습니다.
1996 년도는 Drama 장르 가 128 회로 가장많이 상영 되었습니다.
1976 년도는 Drama 장르 가 21 회로 가장많이 상영 되었습니다.
1992 년도는 Drama 장르 가 78 회로 가장많이 상영 되었습니다.
1967 년도는 Drama 장르 가 23 회로 가장많이 상영 되었습니다.
1993 년도는 Drama 장르 가 99 회로 가장많이 상영 되었습니다.
1964 년도는 Drama 장르 가 17 회로 가장많이 상영 되었습니다.
1977 년도는 Drama 장르 가 27 회로 가장많이 상영 되었습니다.