Day 2

데이터를 수집하고 다루기

지난 시간에는..

- 기본 파이썬 문법
- 형태소 분석
- 단어 개수 세기, 워드클라우드
- 함께 나온 단어 세기, 네트워크

이번 시간에 해볼 것들

- 데이터 다루기
  - 원하는 값만 뽑아내기
  - 값 집계하기
  - 간단한 통계 처리
- 시각화
  - Line plot
  - Bar plot
  - Box plot
  - Histogram
- 데이터 수집
  - 웹 사이트 데이터 긁어오기

데이터 다루기

데이터 다루기

- 정형 데이터 ↔ 비정형 데이터
- 컴퓨터는 정형 데이터를 쉽게 인식/저장/계산할 수 있다.
- 표, 테이블, 또는 데이터프레임(Data Frame)
- 각종 통계 분석 도구
  - 데이터프레임을 쓰기 쉽게,
  - 데이터프레임으로 다양한 통계 분석을 할 수 있도록 지원

데이터프레임

	column 1	column 2	column 3
row 1			
row 2			
row 3			
row 4			

→ 개체

↓ 속성

데이터프레임

	가격	칼로리	판매량
아메리카노	4,100	5	342
카페라떼	4,600	180	475
카라멜 마끼아또	5,600	200	286
핫초코	5,700	435	225

→ 개체

↓ 속성

데이터프레임

	수면 시간	걸음 수	최대 심박수
8/12	6	6857	154
8/13	8	5921	144
8/14	5	8887	183
8/15	7	7642	139

→ **개체**

↓

속성

집계

aggregation

집계

	가격	칼로리	종류	판매량
아메리카노	4,100	5	커피	342
카페라떼	4,600	180	커피	475
카라멜 마끼아또	5,600	200	커피	286
딸기주스	5,500	100	주스	148
망고주스	5,500	112	주스	195
토마토주스	5,000	55	주스	99
루이보스	4,900	0	차	142
캐모마일	4,900	0	차	95
다즐링	4,900	0	차	117

종류별 평균 판매량

	가격	칼로리	종류	판매량
아메리카노	4,100	5	커피	342
카페라떼	4,600	180	커피	475
카라멜 마끼아또	5,600	200	커피	286
딸기주스	5,500	100	주스	148
망고주스	5,500	112	주스	195
토마토주스	5,000	55	주스	99
루이보스	4,900	0	차	142
캐모마일	4,900	0	차	95
다즐링	4,900	0	차	117

	평균 판매량
커피	367.67
주스	147.33
차	118

종류별 평균 칼로리

	가격	칼로리	종류	판매량
아메리카노	4,100	5	커피	342
카페라떼	4,600	180	커피	475
카라멜 마끼아또	5,600	200	커피	286
딸기주스	5,500	100	주스	148
망고주스	5,500	112	주스	195
토마토주스	5,000	55	주스	99
루이보스	4,900	0	차	142
캐모마일	4,900	0	차	95
다즐링	4,900	0	차	117

	평균 칼로리
커피	128
주스	89
차	0

칼로리 그룹별 판매량

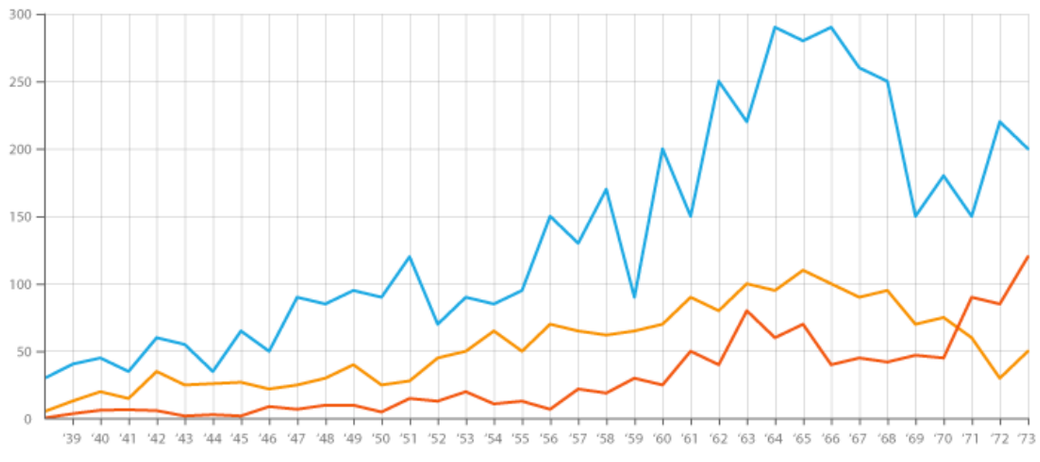
	가격	칼로리	종류	판매량
아메리카노	4,100	5	커피	342
카페라떼	4,600	180	커피	475
카라멜 마끼아또	5,600	200	커피	286
딸기주스	5,500	100	주스	148
망고주스	5,500	112	주스	195
토마토주스	5,000	55	주스	99
루이보스	4,900	0	차	142
캐모마일	4,900	0	차	95
다즐링	4,900	0	차	117

	판매량 평균
칼로리 100 미만	276
칼로리 100 이상	159

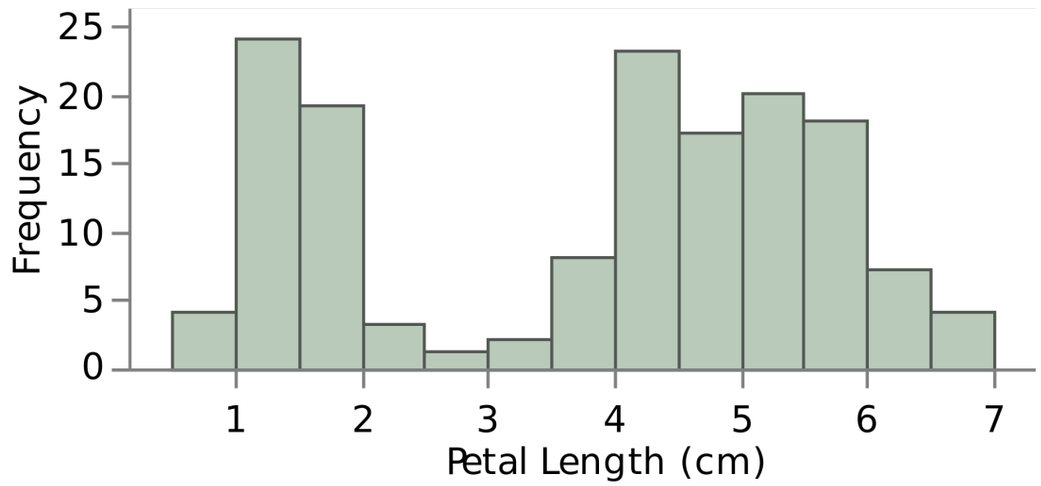
시각화

데이터 시각화

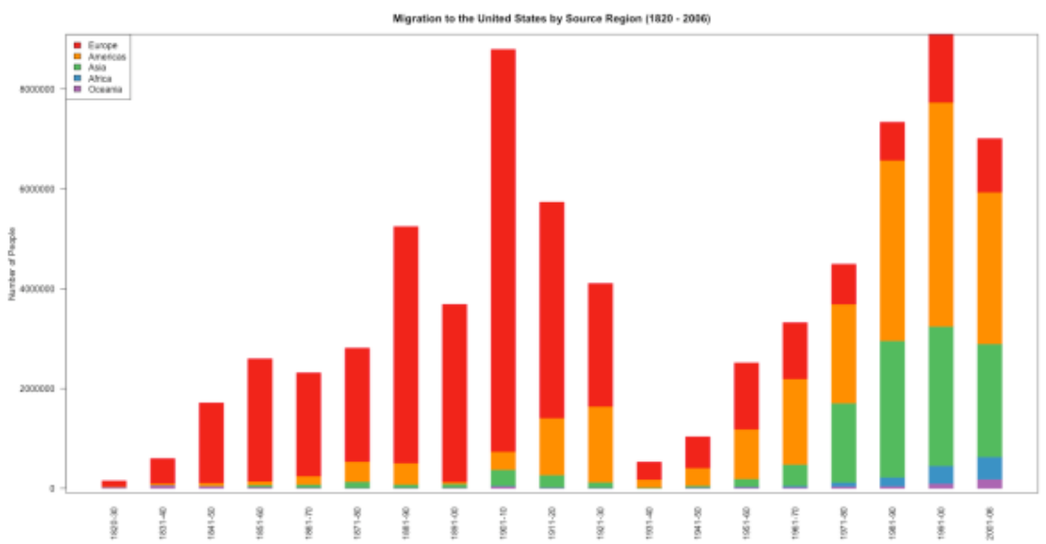
Line chart



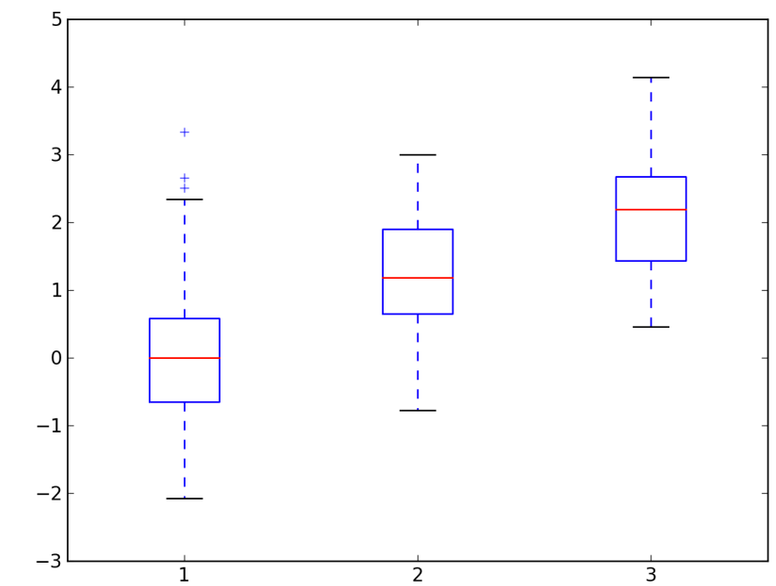
Histogram



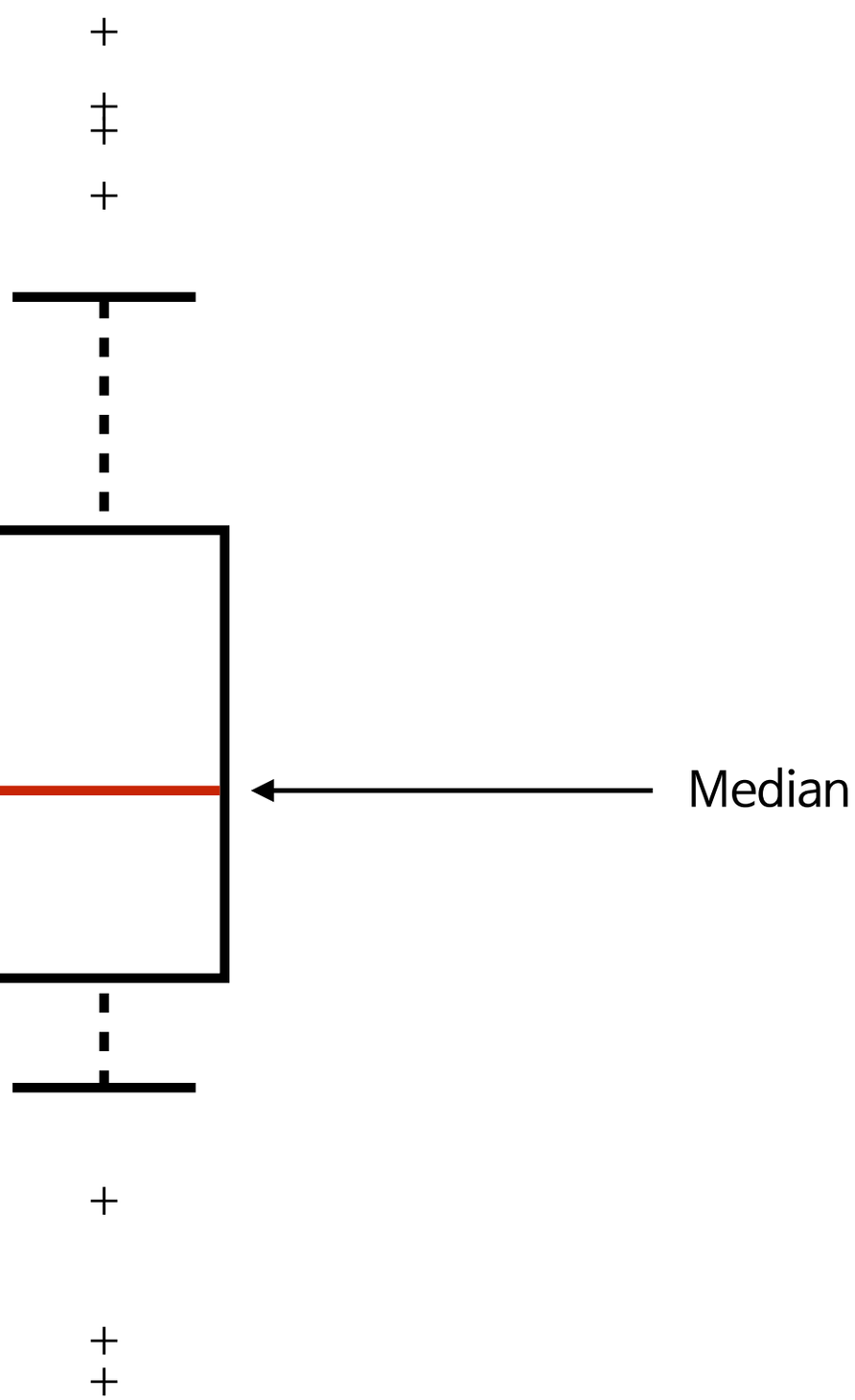
Bar plot



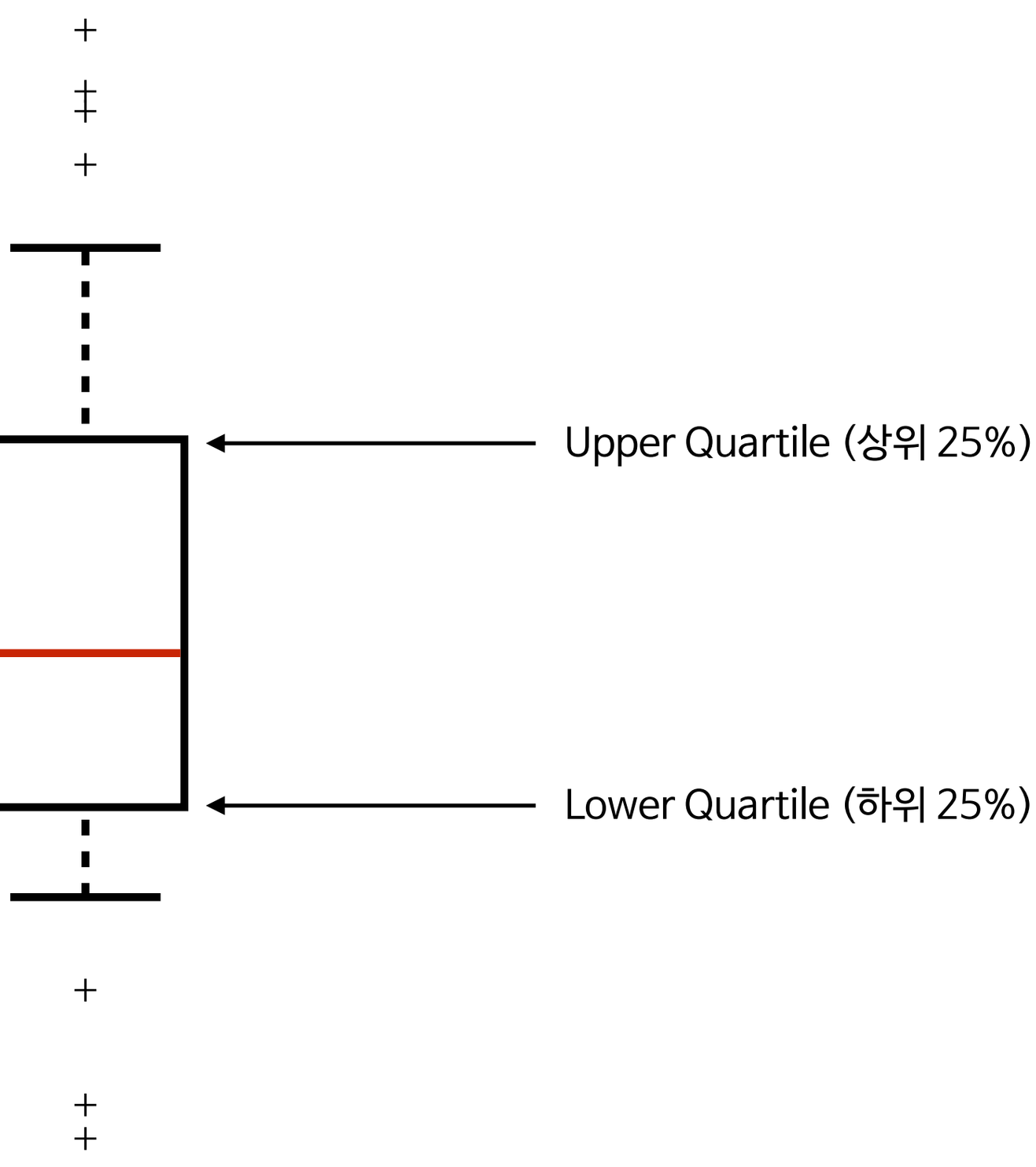
Box plot



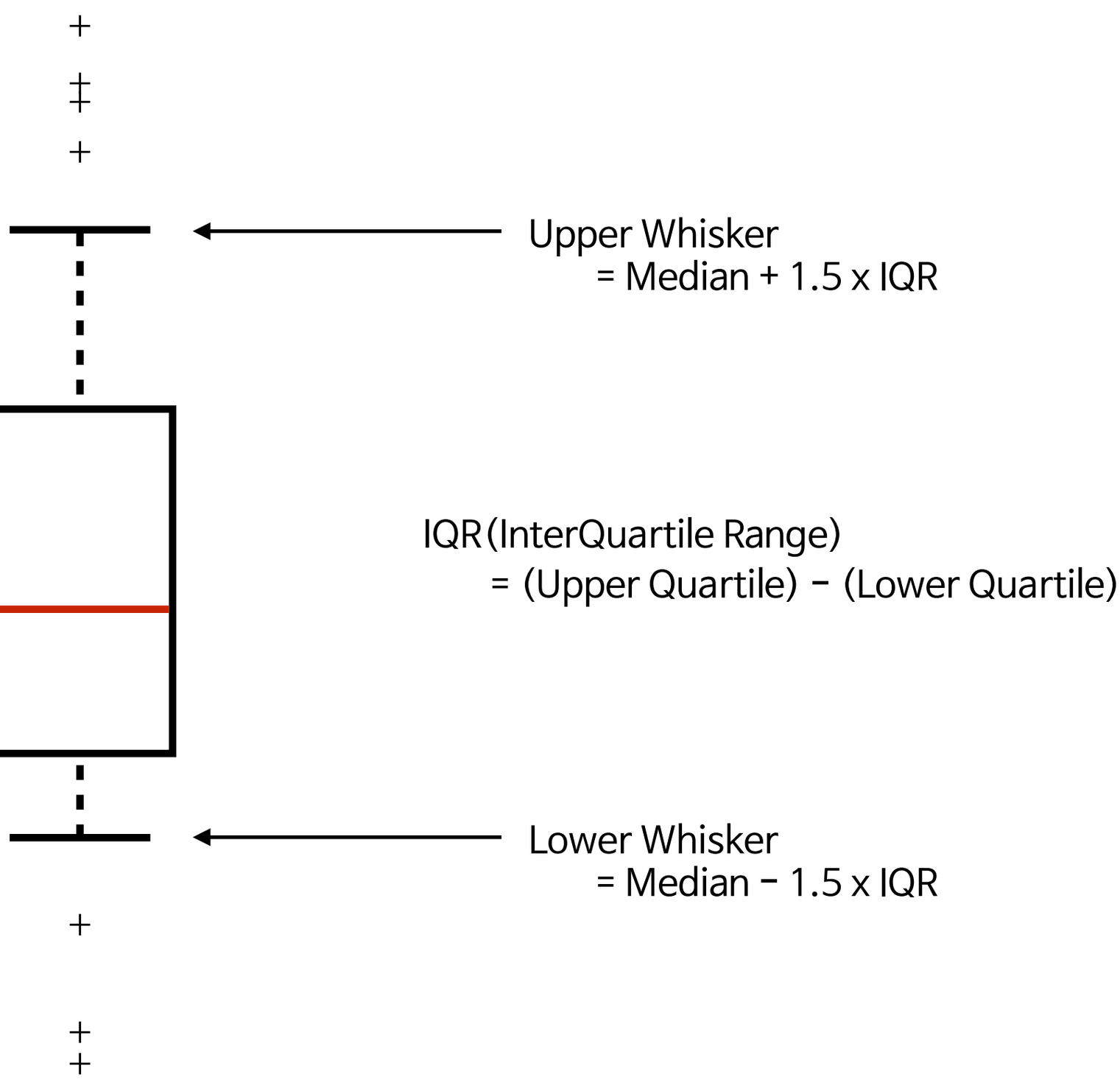
Box plot



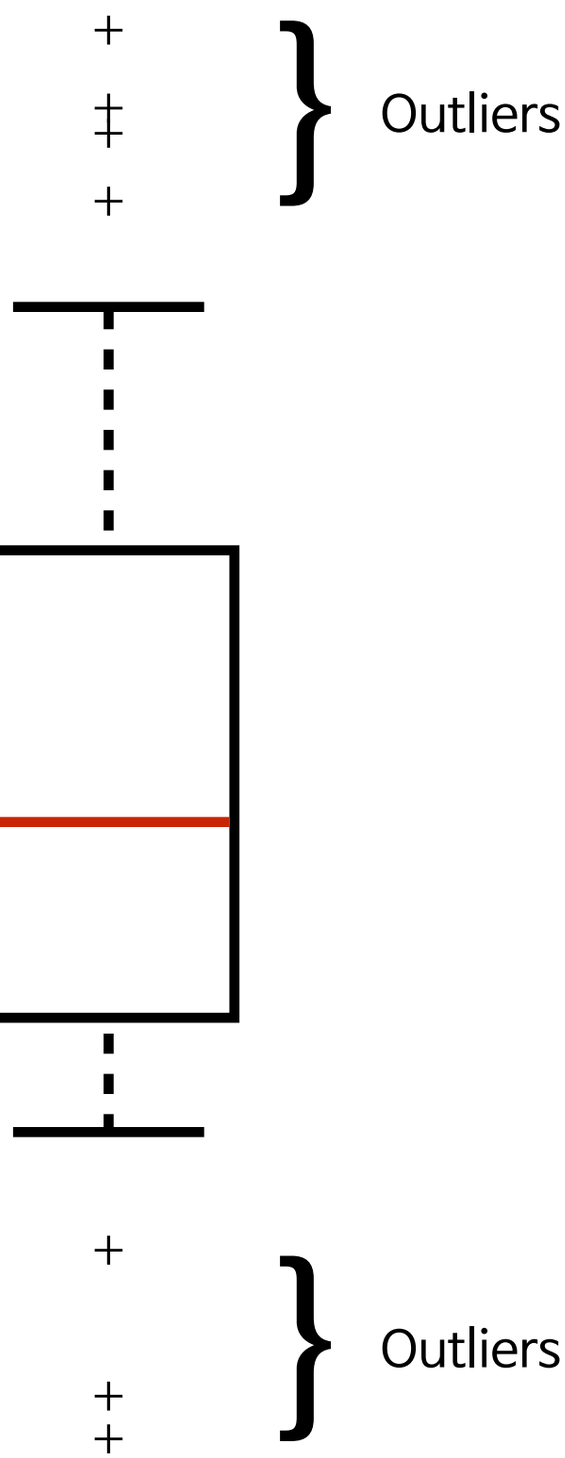
Box plot



Box plot


Upper Whisker
= Median + 1.5 x IQR
IQR (InterQuartile Range)
= (Upper Quartile) − (Lower Quartile)
Lower Whisker
= Median − 1.5 x IQR

Box plot



Tutorial

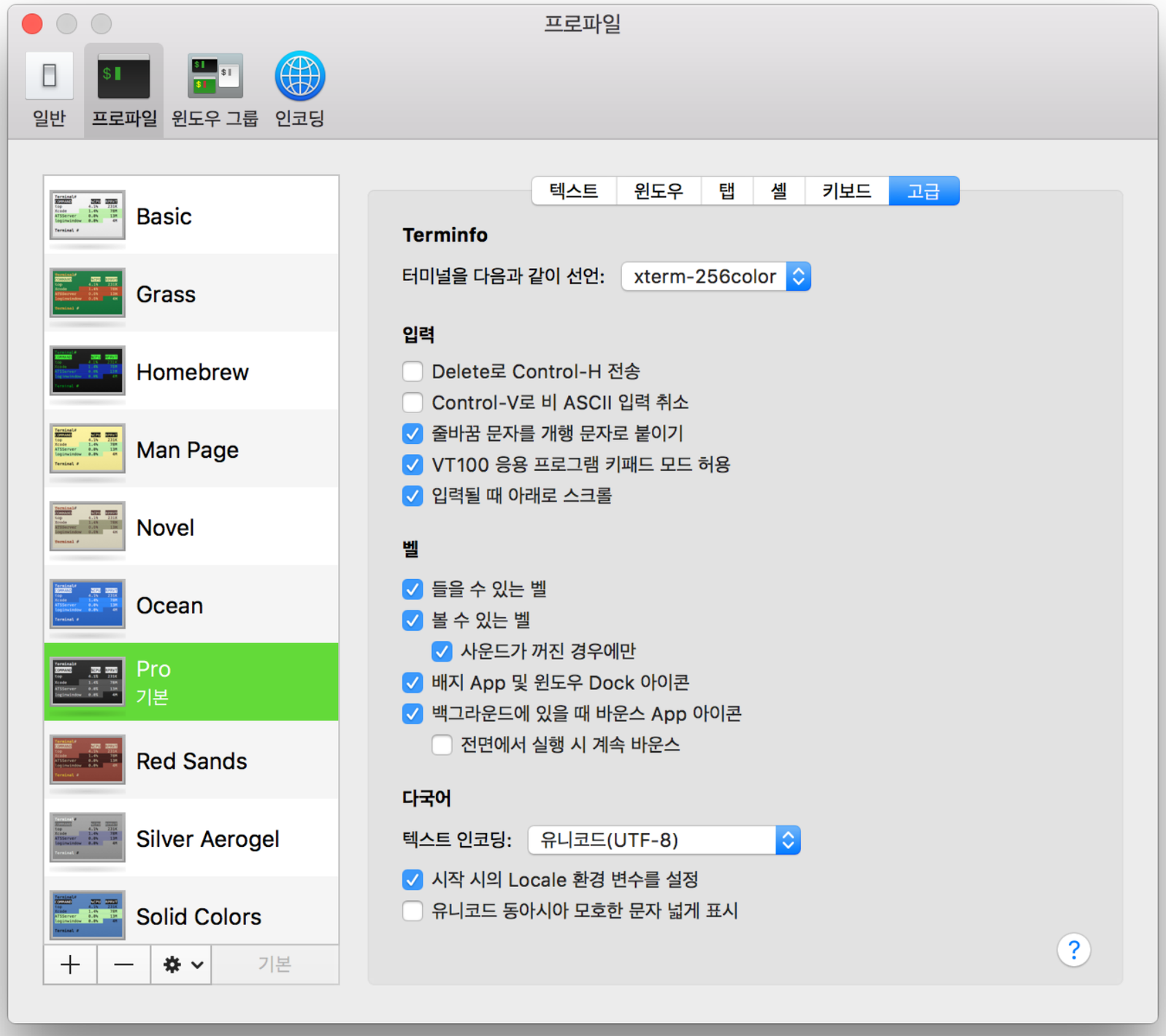
To. 어제 맥 터미널에서 로케일 문제가 발생하셨던 분들

터미널 → Cmd + ,

프로파일 〉 고급 〉 다국어

텍스트 인코딩: 유니코드(UTF-8)

시작 시의 Locale.. ← 체크



1. pandas를 사용해 데이터 다루기

다뤄볼 데이터

- 알바몬 서울 지역 시급 데이터(3일 치)
  - data/alba_data.csv

시각화 준비

```
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt

# 맥
matplotlib.rc('font', family='AppleGothic')

# 윈도우
matplotlib.rc('font', family='Malgun Gothic')
```

데이터 살펴보기


```
!head ../data/alba_data.csv
```

데이터 로드


```
import pandas as pd

df = pd.read_csv('../data/alba_data.csv', delimiter='\t')
```

데이터프레임 구경하기

알바 시급 데이터

```
df.head()
```

	area	company	pay
0	용산구	서울디지털대학교	7000
1	서대문구	서울디지털대학교	7000
2	강서구	서울디지털대학교	7000
3	은평구	서울디지털대학교	7000
4	강동구	맥도날드 서울둔촌DT점	7400

전체 시급 히스토그램 그리기

```
df.hist('pay')
df.hist('pay', bins=30)
df.hist('pay', bins=30, figsize=(15, 10))

plt.show()
```

각 구 별 시급 분포 박스플롯


```
df.boxplot('pay', by='area')
df.boxplot('pay', by='area', vert=False, figsize=(20, 10))
plt.show()
```

아웃라이어 그룹 나누기


```
df_over_10k = df[df['pay'] >= 10000]
df_ord = df[df['pay'] < 10000]
```

아웃라이어 확인

알바 시급 데이터

```
In [23]: df_over_10k
```

Out[23]:

	area	company	pay
6	동대문구	작은고양이샤똥	10000
9	관악구	Healing	20000
38	구로구	Nine9Bar	25000
39	관악구	T/M/B	30000
40	강서구	굿윌아이엔씨㈜	10500
43	구로구	여의궁	22000
45	양천구	여의궁	22000
46	관악구	여의궁	22000
50	성북구	412 BAR (사일이)	12000
53	양천구	pga전문가집단학원	10000
61	구로구	여의궁	22000
63	양천구	여의궁	22000
64	관악구	여의궁	22000
68	성북구	412 BAR (사일이)	12000
71	양천구	pga전문가집단학원	10000
81	강북구	JADE BAR-제이드 모던바	16000
84	관악구	GlassBar(글라스바)	20000

아웃라이어 제외 데이터 플롯


```
df_ord.boxplot('pay', by='area',
                vert=False, figsize=(15, 10))
plt.show()
```

업종별 비교분석

- 편의점 3사
  - 세븐일레븐, GS25, CU

- 패스트푸드 3사
  - 버거킹, 맥도날드, 롯데리아

- ‘CU’ 같은 경우, 편의점이 아닌 상호에 들어가있을 수 있다. 체크 필요.

이름에 ‘CU’가 들어가는 행 살펴보기


```
df[df.apply(lambda x: 'CU' in x['company'], axis=1)]
```

2066	강남구	CU 역삼지혜점	6500
2299	강남구	CU 수서나산점	6500
2518	동대문구	PCCUS PC방	6400
2523	구로구	CU공단오거리점	6700

업종이 편의점인 데이터만 뽑아내기


```
def is_cvs(row):
    company = row['company']
    if company == 'PCCUS PC방':
        return False

    return 'GS25' in company or \
           '세븐일레븐' in company or \
           'CU' in company


df_cvs = df[df.apply(is_cvs, axis=1)].reset_index()
```

편의점 이름만 뽑아서 새 컬럼에 추가하기


```
def shorten_cvs_name(company):
    if 'GS25' in company:
        return 'GS25'
    elif '세븐일레븐' in company:
        return '세븐일레븐'
    else:
        return 'CU'


df_cvs['shortname'] = df_cvs['company'].apply(shorten_cvs_name)
```

편의점 회사별 시급 분포


```
df_cvs.boxplot('pay', by='shortname', vert=False, figsize=(12, 6))
plt.show()
```

직접 해보세요: 패스트푸드 3사

- 패스트푸드 3사만 걸러내기(롯데리아, 버거킹, 맥도날드)
- shortname 붙여주기
- 박스플롯

알바 시급 데이터

```
df_cvs.boxplot('pay', by='shortname', vert=False, figsize=(12, 6))
plt.show()
```

2. 웹문서 크롤링 및 분석

자동화!

- 철학: 자동으로 할 수 있는 건 컴퓨터에게 맡기자.
- 코딩&테스트에 드는 시간 vs 컴퓨터로 수집하는 시간
- 데이터가 500~1000개 이상이면 자동 수집하는 게 좋다.

네이버 웹툰 요일별 비교

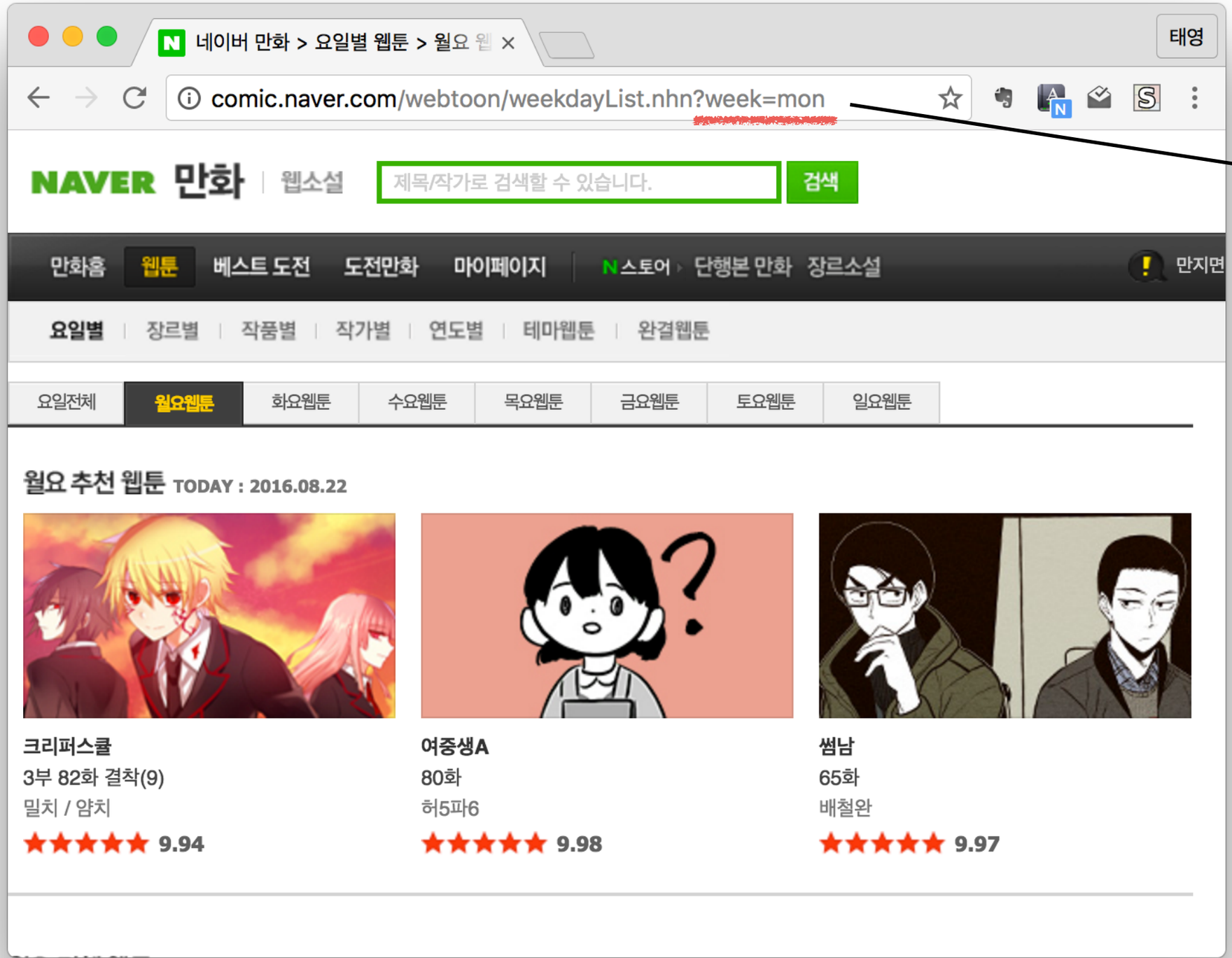
- 얼마나 많은 사람이 볼까?
  - 네이버가 조회수는 안 보여줍니다.
  - ‘별점 준 사람 수’를 proxy로 사용합니다.

- 각 요일 별 별점 등록자 수의 평균, 표준편차 살펴보기

- 평균 별점은?
  - 전체 별점 등록자 수에 대해 가중평균

자동 수집 계획 세우기

요일별 웹툰 목록 받아오기



mon

tue

wed

thu

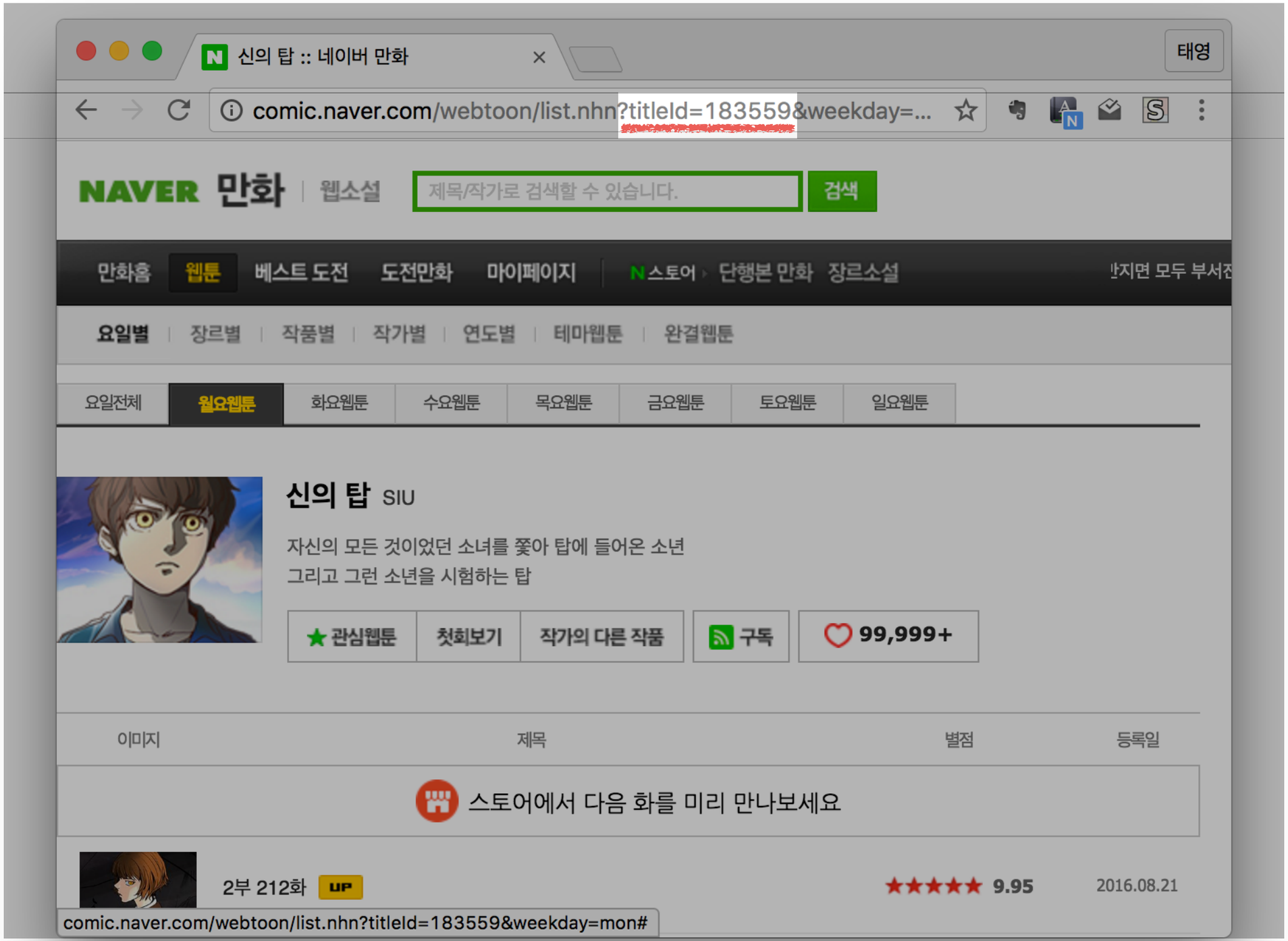
fri

sat

sun

자동 수집 계획 세우기

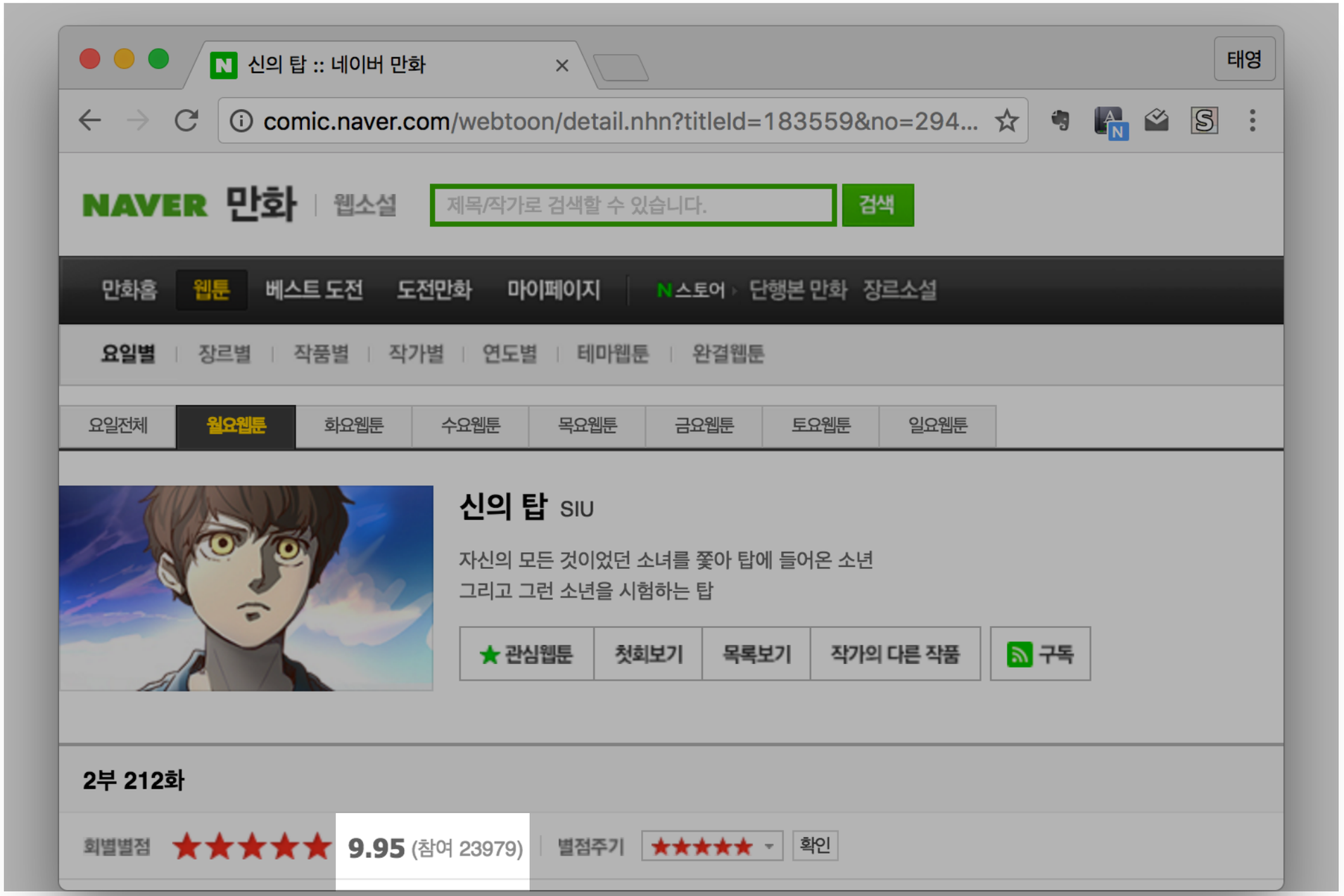
웹툰 ID 알아내기



자동 수집 계획 세우기

웹툰 별점 및 별점 등록자 수

- 각 웹툰의 맨 마지막 화 데이터만 수집
  - 맨 마지막 화의 URL을 알아내는 코드가 필요합니다.



필요 라이브러리 임포트

```
import requests
from bs4 import BeautifulSoup
```

월요웹툰 웹페이지 데이터 가져오기

```
url = 'http://comic.naver.com/webtoon/weekdayList.nhn?week=mon'
g = requests.get(url)
g.content
```

Out[5]:
```
b'\r\n\r\n\r\n  \r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n<!DOCTYPE html>\r\n<html
 lang="ko">\r\n<head>\r\n\t\r\n\t\t\r\n\t\t\r\n\t\t\r\n\t\t\r\n\t\t\r\n\t\t<meta
 http-equiv="X-UA-Compatible" content="IE=edge, chrome=1">\r\n\t\t\r\n\t<
meta http-equiv="Content-type" content="text/html; charset=UTF-8">\r\n\t<
title>\xeb\x84\xa4\xec\x9d\xb4\xeb\xb2\x84 \xeb\xa7\x8c\xed\x99\x94 > \xe
c\x9a\x94\xec\x9d\xbc\xeb\xb3\x84  \xec\x9b\xb9\xed\x88\xb0 > \xec\x9b\x9
4\xec\x9a\x94 \xec\x9b\xb9\xed\x88\xb0</title>\r\n\t\r\n\r\n\r\n\r\n\r\n
\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n\r\n    \r\n    \r\n
   \r\n    \r\n    \r\n    \r\n    \r\n    \r\n    \r\n    \r\n    \r\n
      <meta property="og:title" content="\xeb\x84\xa4\xec\x9d\xb4\xeb\xb2
\x84 \xec\x9b\xb9\xed\x88\xb0" >\r\n        <meta property="og:image" con
tent="http://static.naver.com/comic/images/og_tag.png" >\r\n        <meta
 property="og:description" content="\xeb\xa7\xa4\xec\x9d\xbc\xeb\xa7\xa4
\xec\x9d\xbc \xec\x83\x88\xeb\xa1\x9c\xec\x9a\xb4 \xec\x9e\xac\xeb\xaf\xb
8, \xeb\x84\xa4\xec\x9d\xb4\xeb\xb2\x84 \xec\x9b\xb9\xed\x88\xb0.">\r\n
  \r\n\r\n\r\n<meta property="og:url" content="http://comic.naver.com/web
toon/weekdayList.nhn?week=mon" >\r\n<meta property="og:type" content="art
icle" >\r\n<meta property="og:article:author" content="\xeb\x84\xa4\xec\x
9d\xb4\xeb\xb2\x84 \xec\x9b\xb9\xed\x88\xb0" >\r\n<meta property="og:arti
```

웹페이지 데이터 파싱

```
soup = BeautifulSoup(g.content, 'lxml')

# ‘lxml’: html 문서를 분석할 때 lxml 엔진을 사용하는 옵션
#          파이썬 기본 html 분석 엔진은 매우 느리다.
```

월요웹툰 목록 가져오기

```
soup = BeautifulSoup(g.content, 'lxml')

# ‘lxml’: html 문서를 분석할 때 lxml 엔진을 사용하는 옵션
#          파이썬 기본 html 분석 엔진은 매우 느리다.
```

월요웹툰 목록 가져오기

```
toon_list = soup.find('ul', attrs={'class': 'img_list'})
toon_list = toon_list.find_all('li')
toon_list
```

월요웹툰 링크 가져오기

```
def get_toon_link(item):
    thumb = item.find('div', attrs={'class': 'thumb'})
    link = thumb.find('a')
    link = link['href']
    return link
```

```
toon_links = [get_toon_link(item) for item in toon_list]
toon_links
```

전체 요일 웹툰 링크 가져오기

```
import time

weekdays = ['mon', 'tue', 'wed', 'thu', 'fri', 'sat', 'sun']
base_url = 'http://comic.naver.com/webtoon/weekdayList.nhn?week='

titles = []
links = []
```

전체 요일 웹툰 링크 가져오기

```
for weekday in weekdays:
    url = base_url + weekday
    g = requests.get(url)
    soup = BeautifulSoup(g.content, 'lxml')

    toon_list = soup.find('ul', attrs={'class': 'img_list'})
    toon_list = toon_list.find_all('li')

    toon_titles = [get_toon_title(item) for item in toon_list]
    titles.extend(toon_titles)

    toon_links = [get_toon_link(item) for item in toon_list]
    links.extend(toon_links)

    time.sleep(2)
```

첫 번째 웹툰의 마지막 화 링크 알아보기

```
link = links[0]
toon_url = 'http://comic.naver.com' + link

g = requests.get(toon_url)
soup = BeautifulSoup(g.content, 'lxml')

ep_list = soup.find('table', attrs={'class': 'viewList'})
ep_list = ep_list.find_all('tr')
ep_list
```

첫 번째 웹툰의 마지막 화 링크 알아보기

```
print(ep_list[0])

print(ep_list[1])

print(ep_list[2])

ep_list[2].find('td').find('a')['href']
```

첫 번째 웹툰의 마지막 화 링크 알아보기

```
def get_last_ep_link(ep_list):
    for tr in ep_list[1:]:
        if 'class' not in tr.attrs.keys():
            return tr.find('td').find('a')['href']
```

```
last_ep_link = get_last_ep_link(ep_list)
last_ep_link
```

전체 웹툰의 마지막 화 링크 가져오기

```
last_eps = []

for i, link in enumerate(links):
    print(i + 1, '/', len(links))

    toon_url = 'http://comic.naver.com' + link
    g = requests.get(toon_url)
    soup = BeautifulSoup(g.content, 'lxml')

    ep_list = soup.find('table', attrs={'class': 'viewList'})
    ep_list = ep_list.find_all('tr')

    last_ep = get_last_ep_link(ep_list)
    last_eps.append(last_ep)

    time.sleep(2)
```

첫 번째 웹툰의 마지막 화 회별 별점 및 참여자

```
last_ep = last_eps[0]
last_ep = 'http://comic.naver.com' + last_ep

g = requests.get(last_ep)
soup = BeautifulSoup(g.content, 'lxml')

rating = soup.find('span', attrs={'id': 'topPointTotalNumber'})
rating = rating.get_text()

num_partic = soup.find('span', attrs={'class': 'pointTotalPerson'})
num_partic = num_partic.find('em').get_text()
```

19금 웹툰의 경우..

```
last_ep = 'http://comic.naver.com/webtoon/detail.nhn?
titleId=530312&no=142&weekday=mon'
g = requests.get(last_ep)
soup = BeautifulSoup(g.content, 'lxml')
print(soup.find(‘title'))
```

전체 웹툰의 마지막 화 회별 별점 및 참여자

```
ratings = []
num_partics = []

for i, last_ep in enumerate(last_eps):
    print(i + 1, '/', len(last_eps))

    last_ep = 'http://comic.naver.com' + last_ep
    g = requests.get(last_ep)
    soup = BeautifulSoup(g.content, 'lxml')

    title = soup.find('title').get_text()
    if '로그인' in title:
        ratings.append(-1); num_partics.append(-1)
        continue

    rating = soup.find('span', attrs={'id': 'topPointTotalNumber'})
    ratings.append(float(rating.get_text()))
    num_partic = soup.find('span', attrs={'class': 'pointTotalPerson'})
    num_partics.append(int(num_partic.find('em').get_text()))

    time.sleep(2)
```

요일 분류하기

```
weekdays = [link[-3:] for link in links]
```

pandas DataFrame으로 변환

```
data_dict = {
    'weekday': weekdays,
    'title': titles,
    'rating': ratings,
    'num_partics': num_partics
}

import pandas as pd
df = pd.DataFrame(data_dict)
```

미수집된 웹툰 필터링

```
df = df[df['num_partics'] > 0]
```

참여자 수 Top 20 웹툰

```
df_top_20 = df.sort_values(by='num_partics', ascending=False)
df_top_20 = df_top_20[:20]
df_top_20
```

참여자 수 Top 20 웹툰 시각화

```
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt

matplotlib.rc('font', family='AppleGothic')
matplotlib.style.use('ggplot')
```

참여자 수 Top 20 웹툰 시각화

```
df_top_20.plot.barh(x='title',
                    y='num_partics',
                    figsize=(15, 10))
```

요일별 별점 참여자 평균/표준편차

```
day_mean = df.groupby('weekday')['num_partics'].mean()
day_std = df.groupby('weekday')['num_partics'].std()
```

요일별 별점 참여자 평균/표준편차

```
df2 = pd.DataFrame(index=weekday_names)
df2['mean'] = day_mean
df2['std'] = day_std
```

요일별 별점 참여자 평균/표준편차 시각화

```
df2.plot.bar(figsize=(10, 8))
```