

데이터 분석 기초

- boxplot()
- pandas
 - 엑셀 데이터 읽기
 - 데이터 결합
 - 필요한 데이터 컬럼 추가
 - 각종 통계량 파악
 - 데이터 집계
- 웹 크롤링 기초

2021년 4월 8일

상자 그래프(box plot)

- 데이터에서 얻어낸 최대값, 최소값, 상위 1/4, 2/4(중앙), 3/4에 위치한 값을 보여주는 그래프
- 장점 : 데이터 분포가 한 눈에 보임
- 상자 그래프로 데이터 표현하는 방법
 - randint() 함수를 사용하여 임의의 데이터 만들기 : 1~1000 내의 정수
 - 만들어진 데이터로 상자 그래프 작성 : plt.boxplot() 함수 사용

```
1 import matplotlib.pyplot as plt
```

```
1 import random
```

```
1 result = []  
2 for i in range(13):  
3     result.append(random.randint(1, 1000))  
4 print(sorted(result))  
5  
6 plt.boxplot(result)  
7 plt.show()
```

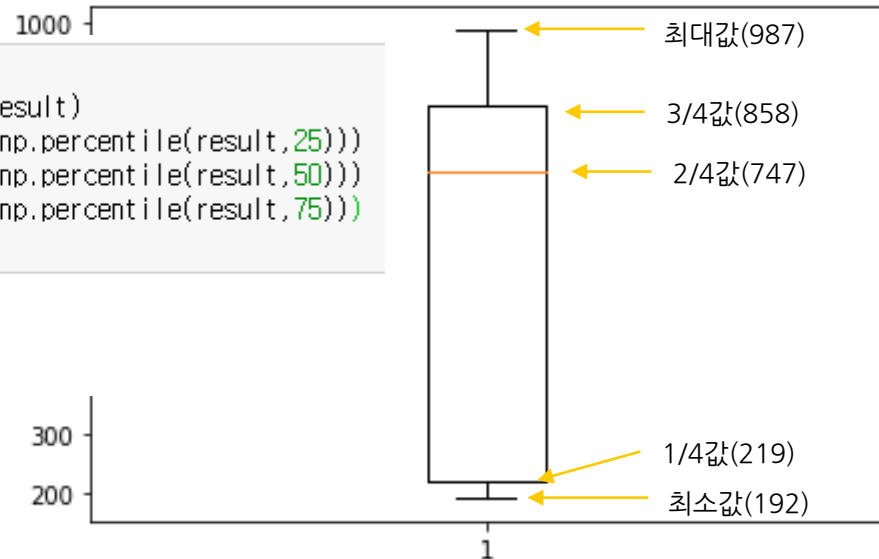
```
[192, 204, 217, 219, 625, 709, 747, 772, 801, 858, 883, 923, 987]
```

```
1 import numpy as np  
2 result = np.array(result)  
3 print('1/4:' + str(np.percentile(result,25)))  
4 print('2/4:' + str(np.percentile(result,50)))  
5 print('3/4:' + str(np.percentile(result,75)))  
6
```

```
1/4:219.0
```

```
2/4:747.0
```

```
3/4:858.0
```

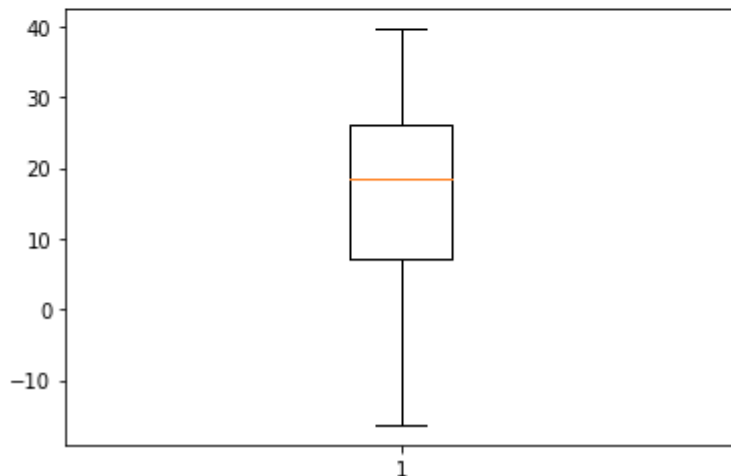


상자 그래프(box plot) : 서울 최고 기온 데이터 나타내기

- 히스토그램 그릴 때 작성했던 코드와 대부분 동일
- hist() 대신 boxplot() 로

```
1 import csv
```

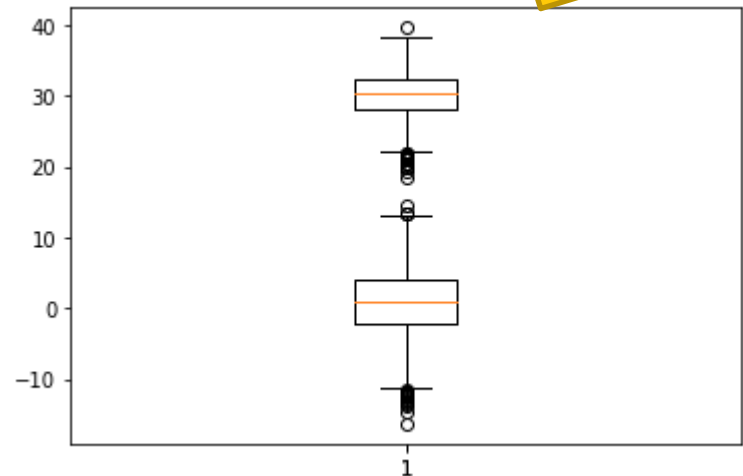
```
1 f = open('seoul.csv')
2 data = csv.reader(f)
3 next(data)
4 result = []
5 for row in data :
6     if row[-1] != '':
7         result.append(float(row[-1]))
8 plt.boxplot(result)
9 plt.show()
```



```
1 result = np.array(result)
2 print('1/4:' + str(np.percentile(result,25)))
3 print('2/4:' + str(np.percentile(result,50)))
4 print('3/4:' + str(np.percentile(result,75)))
5
```

1/4:7.2
2/4:18.6
3/4:26.2

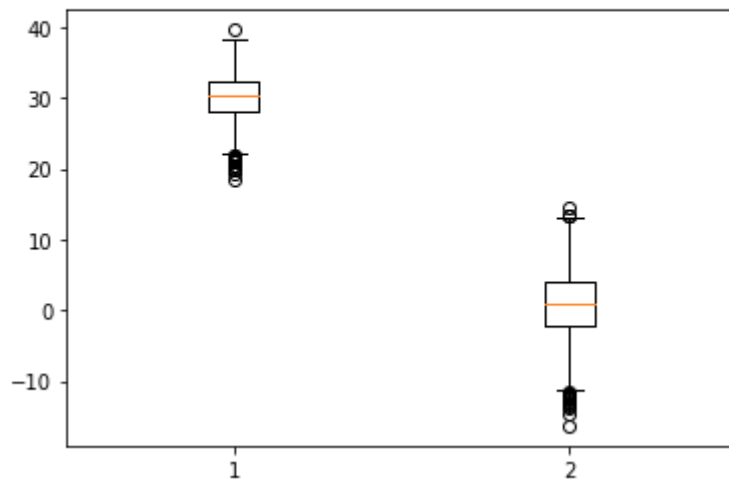
1월과 8월 그래프
동그라미 : 이상치
(outlier) 값 표현



상자 그래프(box plot) : 서울 최고 기온 데이터 나타내기

- 이상치(outlier)
 - 상자 그래프 위, 아래에 그려진 동그라미로 표현
 - 다른 수치에 비해 너무 크거나 작은 값을 자동으로 나타냄
- 8월과 1월의 최고 기온 데이터를 원소로 하는 리스트를 상자 그래프로 나타내면

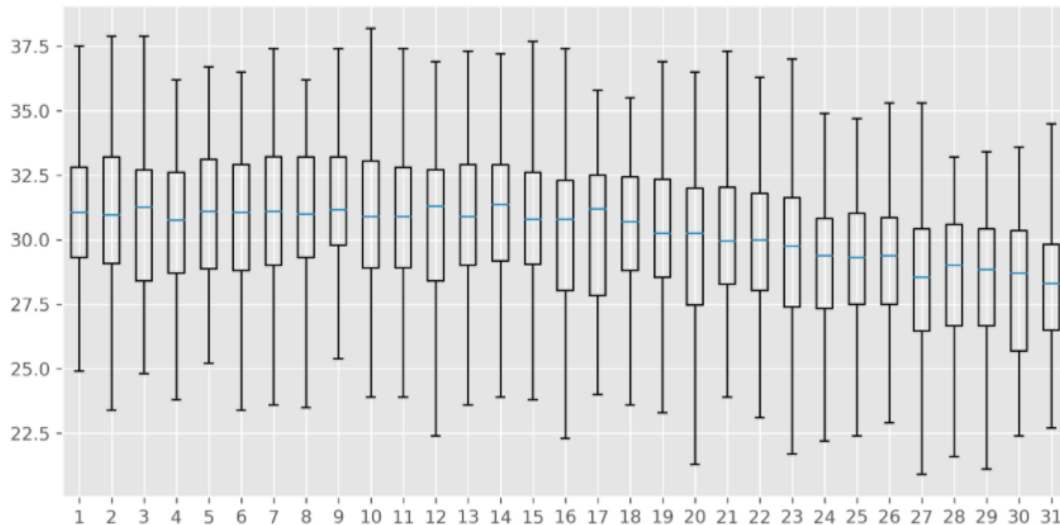
```
1 plt.boxplot([aug, jan])  
2 plt.show()
```



상자 그래프(box plot) : 서울 최고 기온 데이터 나타내기

▪ 8월 일별 기온 데이터를 나타내기

```
1 f = open('seoul.csv')
2 data = csv.reader(f)
3 next(data)
4
5 day = []
6 for i in range(31):
7     day.append([])
8     for row in data:
9         month = row[0].split('-')[1]
10        if row[-1] != '':
11            if month == '08':
12                day[int(row[0].split('-')[2])-1].append(float(row[-1]))
13 plt.style.use('ggplot')
14 plt.figure(figsize=(10,5), dpi=300)
15 plt.boxplot(day, showfliers=False)
16
17 plt.show()
```



- day
 - 일별 데이터를 저장할 리스트
 - day 리스트 안에 31개의 리스트 추가함
- plt.style.use(ggplot) :
 - 그래프 스타일 지정
 - 그래프 배경이 회색 격자 무늬로 변경됨
 - 2/4 값을 의미하는 선 색상이 변경됨
- 그래프 크기 변경
- showfliers = False
 - 이상치 값이 보이지 않게 설정

pandas

■ 가장 많이 접하는 데이터 형태

- 엑셀과 같은 스프레드시트
- 로우(row, 행), 컬럼(column, 열)으로 구성된 테이블 형태
- 판다스는 테이블 형태의 데이터를 쉽게 다룰 수 있는 파이썬 라이브러리

■ 엑셀 데이터 불러오기

- read_excel() 사용
- sample_1.xlsx 파일 특징
 - 1행 : 해당 데이터의 제목, 분석에 불필요한 정보
 - 2행 : 4개의 열이름 중 국적코드, 성별, 입국객수 사용
 - 3~8행 : 분석하고자 하는 데이터가 열이름에 맞춰 구성됨
 - 9~10행 : 총합계와 전년동기라는 요약정보로 구성됨
불필요한 정보

sample_1.xlsx

	A	B	C	D
1	2019년 11월 입국객 수			
2	국적코드	성별	입국객수	전년동기
3	A01	남성	106320	85815
4	A01	여성	191436	125241
5	A31	남성	319	299
6	A31	여성	42	54
7	A18	남성	158912	124486
8	A18	여성	232943	163466
9	총 합계	689972		
10	전년동기	499361		

pandas

■ 데이터 분석에 필요한 부분 불러오기

■ header = 1

- 컬럼명(열이름)이 있는 위치
- 파이썬의 인덱스 시작 번호는 0부터 시작, 그러므로 두번째 행(row)을 표현할 때 1로 나타냄

```
1 sample_1 = pd.read_excel('sample_1.xlsx',
2                           header=1,
3                           skipfooter=2,
4                           usecols='A:C')
```

```
1 sample_1.head(3)
```

	국적코드	성별	입국객수
0	A01	남성	106320
1	A01	여성	191436
2	A31	남성	319

```
1 sample_1.tail(3)
```

	국적코드	성별	입국객수
3	A31	여성	42
4	A18	남성	158912
5	A18	여성	232943

sample_1.xlsx

	A	B	C	D
1	2019년 11월 입국객 수			
2	국적코드	성별	입국객수	전년동기
3	A01	남성	106320	85815
4	A01	여성	191436	125241
5	A31	남성	319	299
6	A31	여성	42	54
7	A18	남성	158912	124486
8	A18	여성	232943	163466
9	총 합계	689972		
10	전년동기	499361		

pandas

■ 데이터 구성 확인

- sample_1.info() : 데이터의 요약 정보 확인
 - 이 데이터는 pandas의 데이터 프레임(DataFrame) 클래스임
 - RangeIndex : 0~5까지 총 6개의 원소(행)으로 구성됨
 - Data Columns
 - » 총 3개의 컬럼으로 구성됨
 - » 국적코드, 성별은 6개의 빈 칸 없는(non-null) 객체(문자열)
 - » 입국객수는 널이 없는 6개의 정수형 속성
 - dtypes : 정수형 1개, 문자열 2개로 구성됨
 - memory usage : 이 데이터는 272 바이트 차지

```
1 sample_1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6 entries, 0 to 5
Data columns (total 3 columns):
국적코드      6 non-null object
성별          6 non-null object
입국객수      6 non-null int64
dtypes: int64(1), object(2)
memory usage: 272.0+ bytes
```

1	sample_1
	국적코드 성별 입국객수
0	A01 남성 106320
1	A01 여성 191436
2	A31 남성 319
3	A31 여성 42
4	A18 남성 158912
5	A18 여성 232943

pandas

■ 데이터 기초 통계량 확인

- sample_1.describe()
 - 숫자형 원소에 대한 여러 가지 통계량 출력하는 함수
 - 예제 데이터에서 숫자형 원소는 입국객수 1개
 - 입국객수에 대한 여러 가지 통계량 표출
 - » count : 개수
 - » mean : 평균 값
 - » std : 표준편차
 - » 최소값, 1사분위, 2사분위(중위수), 3사분위, 최대값

```
1 sample_1.describe()
```

입국객수	
count	6.000000
mean	114995.333333
std	98105.752006
min	42.000000
25%	26819.250000
50%	132616.000000
75%	183305.000000
max	232943.000000

1	sample_1		
	국적코드	성별	입국객수
0	A01	남성	106320
1	A01	여성	191436
2	A31	남성	319
3	A31	여성	42
4	A18	남성	158912
5	A18	여성	232943

pandas

■ 데이터 선택 - 컬럼 기준

- 원하는 부분만 선택하는 방법
- 입국객수 컬럼만 선택하여 확인

```
1 sample_1['입국객수']
```

```
0    106320
1    191436
2         319
3          42
4    158912
5    232943
```

Name: 입국객수, dtype: int64

- 국적코드, 입국객수 컬럼 선택하여 확인

	국적코드	입국객수
0	A01	106320
1	A01	191436
2	A31	319
3	A31	42
4	A18	158912
5	A18	232943

여러 개의 컬럼 선택은
리스트로 묶어서 추출
해야 함

1	sample_1		
	국적코드	성별	입국객수
0	A01	남성	106320
1	A01	여성	191436
2	A31	남성	319
3	A31	여성	42
4	A18	남성	158912
5	A18	여성	232943

pandas

데이터 선택 - 컬럼 생성

- 기준년월 컬럼 생성 후 '2019-11' 값 부여 방법

1	sample_1['기준년월'] = '2019-11'
2	sample_1

	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
1	A01	여성	191436	2019-11
2	A31	남성	319	2019-11
3	A31	여성	42	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

1	sample_1		
	국적코드	성별	입국객수
0	A01	남성	106320
1	A01	여성	191436
2	A31	남성	319
3	A31	여성	42
4	A18	남성	158912
5	A18	여성	232943



1	sample_1			
	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
1	A01	여성	191436	2019-11
2	A31	남성	319	2019-11
3	A31	여성	42	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

pandas

데이터 선택 - 로우(row, 행) 기준

- 특정 조건에 맞는 데이터를 찾을 때
- 즉, 필터링(filtering)한 결과를 찾을 때 사용
- sample_1에서 성별이 남성인 데이터 찾기

```
1 condition = (sample_1['성별'] == '남성')
2 condition
```

```
0    True
1    False
2     True
3    False
4     True
5    False
Name: 성별, dtype: bool
```

```
1 sample_1[condition]
```

	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
2	A31	남성	319	2019-11
4	A18	남성	158912	2019-11

	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
1	A01	여성	191436	2019-11
2	A31	남성	319	2019-11
3	A31	여성	42	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

입국객수 150,000 이상인 데이터 추출

	국적코드	성별	입국객수	기준년월
1	A01	여성	191436	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

- 입국객수 150,000 이상이고 남성인 데이터 추출
 - & : and, | : or

pandas

■ 데이터 선택 - 한 컬럼에 여러 조건으로 필터링

- 국적코드가 A01 또는 A18인 데이터 추출
 - `isin()` : 찾고 싶은 값들을 리스트 형태로 조건 추가

```
1 conditions = (sample_1['국적코드'] == 'A01') &&  
2               | (sample_1['국적코드'] == 'A18')  
3 sample_1[conditions]
```

	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
1	A01	여성	191436	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

```
1 conditions = (sample_1['국적코드'].isin(['A01', 'A18']))  
2 conditions
```

```
0    True  
1    True  
2   False  
3   False  
4    True  
5    True  
Name: 국적코드, dtype: bool
```

```
1 conditions = (sample_1['국적코드'].isin(['A01', 'A18']))  
2 sample_1[conditions == False]
```

	국적코드	성별	입국객수	기준년월
2	A31	남성	319	2019-11
3	A31	여성	42	2019-11

1	sample_1			
	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
1	A01	여성	191436	2019-11
2	A31	남성	319	2019-11
3	A31	여성	42	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

pandas

데이터 통합 - 가로로 통합

- merge() 사용
- 어느 나라 입국객인지 알고 싶다.
 - sample_codemaster.xlsx 읽기
 - » 국적코드와 국적명을 매핑(mapping) 해 놓은 국적코드 표
 - 국적코드 표를 통해 국적코드에 해당하는 국적명을 추가
 - sample_1 데이터에 국적명 컬럼 추가

```
1 sample_1_code = pd.merge(left=sample_1,  
2                           right=code_master,  
3                           how='left',  
4                           left_on='국적코드',  
5                           right_on='국적코드')  
6 sample_1_code
```

	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	106320	2019-11	일본
1	A01	여성	191436	2019-11	일본
2	A31	남성	319	2019-11	NaN
3	A31	여성	42	2019-11	NaN
4	A18	남성	158912	2019-11	중국
5	A18	여성	232943	2019-11	중국

나타내고 싶지 않으면
how='inner'로 변경
수행

1	sample_1			
	국적코드	성별	입국객수	기준년월
0	A01	남성	106320	2019-11
1	A01	여성	191436	2019-11
2	A31	남성	319	2019-11
3	A31	여성	42	2019-11
4	A18	남성	158912	2019-11
5	A18	여성	232943	2019-11

1	code_master	
	국적코드	국적명
0	A01	일본
1	A02	대만
2	A03	홍콩
3	A18	중국
4	A19	이란
5	A22	우즈베키스탄
6	A23	카자흐스탄
7	A99	아시아 기타

pandas

데이터 통합 - 세로로 통합

- append()나 concat() 사용
- 실제 업무에서는 월별 파일로 관리하는 경우가 많음
- 여러 개의 데이터 파일을 대상으로 시계열 분석
 - ⇒ 데이터 통합 먼저 수행
 - sample_2.xlsx 읽기
 - » 2019년 12월 데이터
 - » sample_1처럼 기준년월, 국적명 컬럼 추가
 - 두 데이터 세로로 통합
 - » ignore_index = True : 기존 인덱스 무시

```
1 sample = sample_1_code.append(sample_2_code, ignore_index=True)
```

```
1 sample_c = pd.concat([sample_1_code, sample_2_code], ignore_index=True)
2 sample_c
```

```
1 sample_1_code.append(sample_2_code)
```

1	sample_1_code				
	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	106320	2019-11	일본
1	A01	여성	191436	2019-11	일본
2	A31	남성	319	2019-11	NaN
3	A31	여성	42	2019-11	NaN
4	A18	남성	158912	2019-11	중국
5	A18	여성	232943	2019-11	중국

1	sample_2_code				
	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	92556	2019-12	일본
1	A01	여성	163737	2019-12	일본
2	A18	남성	155540	2019-12	중국
3	A18	여성	249023	2019-12	중국

pandas

■ 데이터 통합 - 세로로 통합

```
1 sample = sample_1_code.append(sample_2_code, ignore_index=True)
```

```
1 sample
```

	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	106320	2019-11	일본
1	A01	여성	191436	2019-11	일본
2	A31	남성	319	2019-11	NaN
3	A31	여성	42	2019-11	NaN
4	A18	남성	158912	2019-11	중국
5	A18	여성	232943	2019-11	중국
6	A01	남성	92556	2019-12	일본
7	A01	여성	163737	2019-12	일본
8	A18	남성	155540	2019-12	중국
9	A18	여성	249023	2019-12	중국

```
1 sample_c = pd.concat([sample_1_code, sample_2_code], ignore_index=True)
2 sample_c
```

	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	106320	2019-11	일본
1	A01	여성	191436	2019-11	일본
2	A31	남성	319	2019-11	NaN
3	A31	여성	42	2019-11	NaN
4	A18	남성	158912	2019-11	중국
5	A18	여성	232943	2019-11	중국
6	A01	남성	92556	2019-12	일본
7	A01	여성	163737	2019-12	일본
8	A18	남성	155540	2019-12	중국
9	A18	여성	249023	2019-12	중국

```
1 sample_1_code.append(sample_2_code)
```

	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	106320	2019-11	일본
1	A01	여성	191436	2019-11	일본
2	A31	남성	319	2019-11	NaN
3	A31	여성	42	2019-11	NaN
4	A18	남성	158912	2019-11	중국
5	A18	여성	232943	2019-11	중국
0	A01	남성	92556	2019-12	일본
1	A01	여성	163737	2019-12	일본
2	A18	남성	155540	2019-12	중국
3	A18	여성	249023	2019-12	중국

pandas

■ 데이터 저장

■ to_excel() 사용

```
1 sample.to_excel('sample.xlsx')
```

	A	B	C	D	E	F
1		국적코드	성별	입국객수	기준년월	국적명
2	0	A01	남성	106320	2019-11	일본
3	1	A01	여성	191436	2019-11	일본
4	2	A31	남성	319	2019-11	
5	3	A31	여성	42	2019-11	
6	4	A18	남성	158912	2019-11	중국
7	5	A18	여성	232943	2019-11	중국
8	6	A01	남성	92556	2019-12	일본
9	7	A01	여성	163737	2019-12	일본
10	8	A18	남성	155540	2019-12	중국
11	9	A18	여성	249023	2019-12	중국

```
1 sample.to_excel('sample_index_false.xlsx', index=False)
```

	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	106320	2019-11	일본
1	A01	여성	191436	2019-11	일본
2	A31	남성	319	2019-11	NaN
3	A31	여성	42	2019-11	NaN
4	A18	남성	158912	2019-11	중국
5	A18	여성	232943	2019-11	중국
6	A01	남성	92556	2019-12	일본
7	A01	여성	163737	2019-12	일본
8	A18	남성	155540	2019-12	중국
9	A18	여성	249023	2019-12	중국

pandas

■ 데이터 집계

- pivot_table() 사용
- 피벗 테이블
 - 기존 데이터의 컬럼을 재구성해서 데이터에 대한 통계를 한 눈에 파악할 수 있게 정리한 표
- 피벗 기능을 이용하면 데이터를 원하는 형태로 손쉽게 집계
 - 데이터를 날짜별로 집계해 주세요.
 - 데이터를 연령별로 추출해 주세요.
 - ??별로 데이터 집계 할 때 쓰는 강력한 기능
- 국적과 기준년월에 따른 입국객수의 평균값 추출

```
1 sample_pivot = sample.pivot_table(values='입국객수',
2                                   index='국적명',
3                                   columns='기준년월',
4                                   aggfunc='mean' )
```

```
1 sample_pivot
```

기준년월	2019-11	2019-12
국적명		
일본	148878.0	128146.5
중국	195927.5	202281.5

1	sample				
	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	106320	2019-11	일본
1	A01	여성	191436	2019-11	일본
2	A31	남성	319	2019-11	NaN
3	A31	여성	42	2019-11	NaN
4	A18	남성	158912	2019-11	중국
5	A18	여성	232943	2019-11	중국
6	A01	남성	92556	2019-12	일본
7	A01	여성	163737	2019-12	일본
8	A18	남성	155540	2019-12	중국
9	A18	여성	249023	2019-12	중국

옵션	설명
mean	평균
sum	합계
min	최소값
median	중앙값
max	최대값
count	개수
nunique	중복을 제거한 원소 개수

pandas

데이터 집계

pivot_table() 사용

- values : 피벗 테이블 내부에 들어갈 값 지정
- index : 피벗 테이블 행(row)에 들어갈 값 지정
- columns : 피벗 테이블 열(column)에 들어갈 값 지정
- aggfunc : 집계 값의 종류 지정

```
1 sample_pivot = sample.pivot_table(values='입국객수',
2                                   index='국적명',
3                                   columns='기준년월',
4                                   aggfunc='mean')
```

index, columns는 조건부 존재

- 국적에 따른 최대 입국객 수

```
1 sample_pivot_2 = sample.pivot_table(values='입국객수',
2                                     index='국적명',
3                                     aggfunc='max')
```

입국객수	
국적명	
일본	191436
중국	249023

1	sample				
	국적코드	성별	입국객수	기준년월	국적명
0	A01	남성	106320	2019-11	일본
1	A01	여성	191436	2019-11	일본
2	A31	남성	319	2019-11	NaN
3	A31	여성	42	2019-11	NaN
4	A18	남성	158912	2019-11	중국
5	A18	여성	232943	2019-11	중국
6	A01	남성	92556	2019-12	일본
7	A01	여성	163737	2019-12	일본
8	A18	남성	155540	2019-12	중국
9	A18	여성	249023	2019-12	중국

옵션	설명
mean	평균
sum	합계
min	최소값
median	중앙값
max	최대값
count	개수
nunique	중복을 제거한 원소 개수

웹 크롤링

- 웹 페이지의 정보를 가져오는 것
- 짧은 시간 동안 과도하게 데이터를 수집하는 행위는 해당 서버에 무리를 주거나, 디도스 공격 등으로 감지될 수 있음
- 라이브러리 selenium의 webdriver를 활용해 웹 브라우저 조작
- 라이브러리 BeautifulSoup을 활용해 웹 페이지 상의 HTML 데이터에서 필요한 정보를 가져오기

selenium 라이브러리의 webdriver

- 크롬이나 에지 등의 웹 브라우저에서 사이트 접속, 버튼 클릭, 글자 입력과 같이 사람이 할 수 있는 일들을 코드를 통해 제어할 수 있는 라이브러리
- webdriver 를 활용하기 위해서는 사용 중인 웹 브라우저의 종류에 따라 제어 가능한 드라이버 필요
- 크롬 드라이버를 이용한 크롤링 진행

selenium과 크롬 드라이버 설치(1)

- `from selenium import webdriver`
 - 에러 발생 : `! pip install selenium`
 - `!` : 해당 명령어를 콘솔 창에서 실행하라는 의미
 - `pip` : 파이썬 라이브러리를 설치하기 위한 명령어
 - `pip install 라이브러리명` : 지정한 라이브러리를 컴퓨터에 설치

selenium 라이브러리의 webdriver

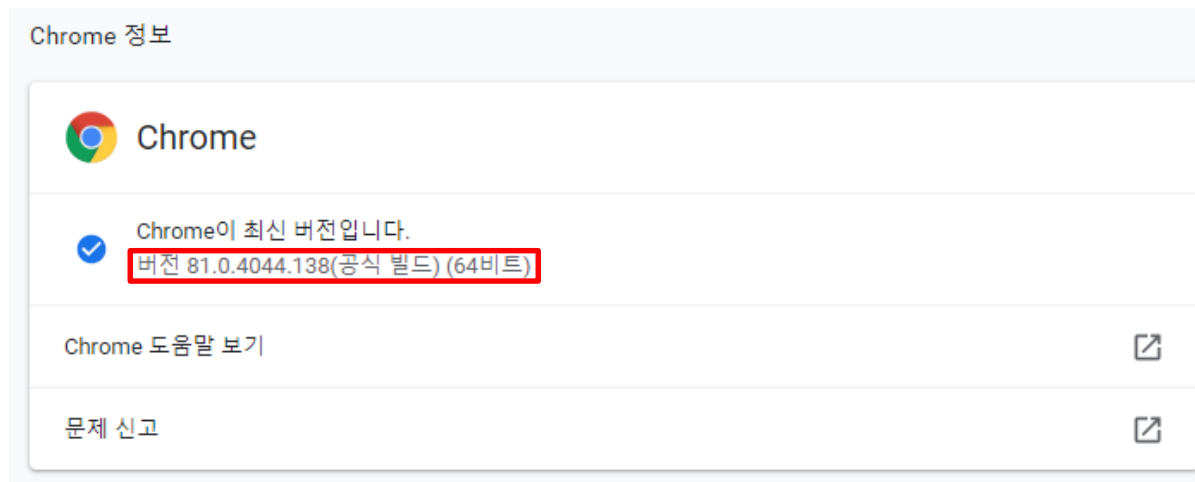
selenium 과 크롬 드라이버 설치(2)

■ 크롬 드라이버 다운

- 크롬 드라이버는 selenium의 webdriver를 통해 파이썬에서 크롬 브라우저를 제어할 수 있게 해 줌
- 기존에 크롬이 설치되어 있더라도 selenium으로 작동하는 크롬드라이버는 별도의 파일 필요
- 현재 사용 중인 크롬과 운영체제의 버전 확인
 - 크롬-오른쪽 상단 (닫기 버튼 아래) 다음 버튼 클릭
 - [도움말]-[Chrome 정보]



메뉴버튼



Chrome이 최신 버전입니다.(2021.04.06 확인)
버전 89.0.4389.114(공식 빌드) (64비트)

selenium 라이브러리의 webdriver

selenium 과 크롬 드라이버 설치(3)

- 크롬 드라이버 다운 : zip 파일 압축을 푼 폴더 위치 기억해야 함
- <https://sites.google.com/a/chromium.org/chromedriver/downloads>

ChromeDriver - WebDriver for Chrome

CHROMEDRIVER

CAPABILITIES & CHROME OPTIONS

CHROME EXTENSIONS

CHROMEDRIVER CANARY

CONTRIBUTING

▼ DOWNLOADS

VERSION SELECTION

▼ GETTING STARTED

ANDROID

CHROME OS

▼ LOGGING

PERFORMANCE LOG

MOBILE EMULATION

▼ NEED HELP?

CHROME DOESN'T START OR CRASHES IMMEDIATELY

CHROMEDRIVER CRASHES

CLICKING ISSUES

DEVTOOLS WINDOW KEEPS CLOSING

KEYBOARD SUPPORT

OPERATION NOT SUPPORTED WHEN USING REMOTE DEBUGGING

SECURITY CONSIDERATIONS

Downloads

Current Releases

- If you are using Chrome version 83, please download [ChromeDriver 83.0.4103.39](#)
- If you are using Chrome version 81, please download [ChromeDriver 81.0.4044.138](#)
- If you are using Chrome version 80, please download [ChromeDriver 80.0.3987.106](#)
- For older version of Chrome, please see below for the version of ChromeDriver that supports it.

If you are using Chrome from Dev or Canary channel, please following instructions on the [Chrome Dev/Canary page](#).

For more information on selecting the right version of ChromeDriver, please see the [Version Selection page](#).

ChromeDriver 83.0.4103.39

Supports Chrome version 83

- Updated Chromedriver to work correctly with prototype.js.





For more details, please see the [release notes](#).

ChromeDriver 83.0.4103.14

Supports Chrome version 83

- Resolved issue 1778: Deprecate launchApp from ChromeDriver
- Resolved issue 2520: InitSession can wait forever when Chrome is unresponsive
- Resolved issue 3120: Headless mode download from new tab
- Resolved issue 3234: Confirm semicolon found before substring

Index of /81.0.4044.138/

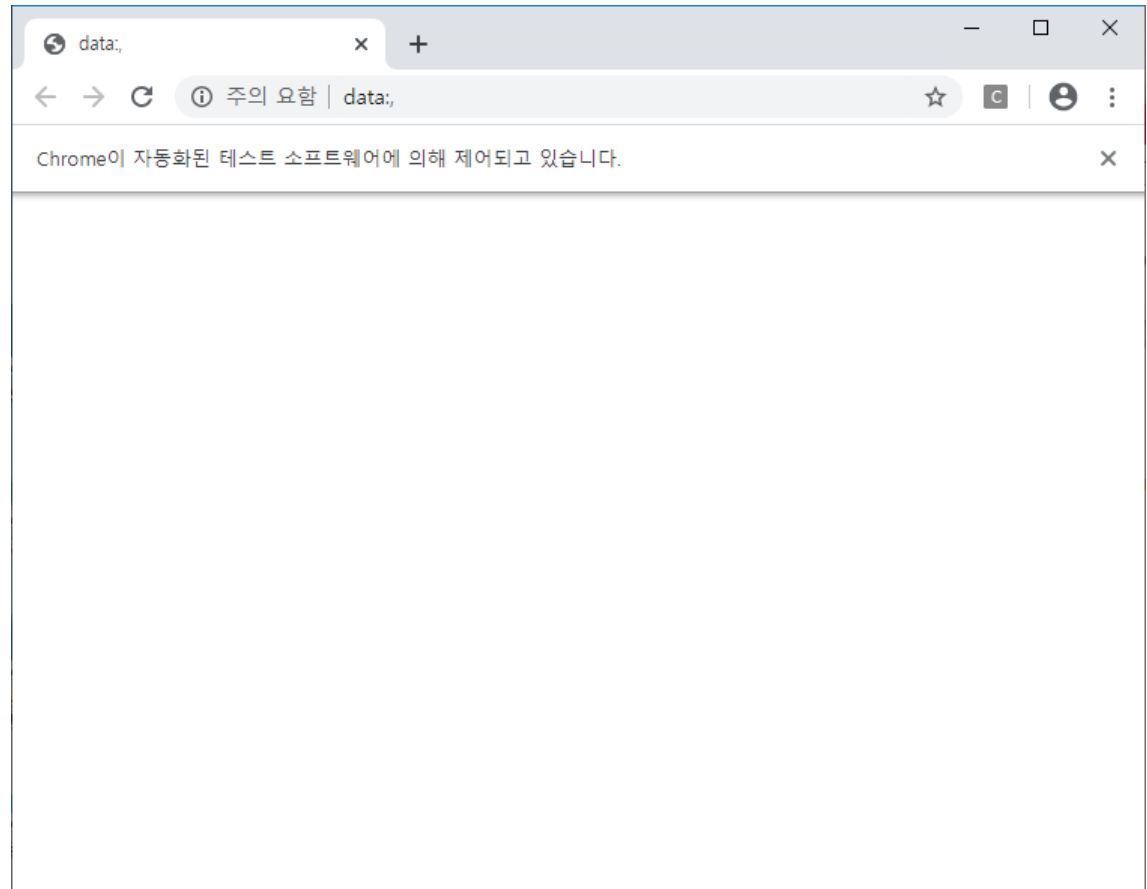
	Name	Last modified	Size	ETag
	Parent Directory		-	
	chromedriver_linux64.zip	2020-05-05 20:33:58	4.74MB	6581e09a8ce12da2239ac21b2a1cd20b
	chromedriver_mac64.zip	2020-05-05 20:34:00	6.70MB	4b2ace862187dc9e53d29c9f12710731
	chromedriver_win32.zip	2020-05-05 20:34:01	4.20MB	d19aef5daf9dbaeac152d27066285a7b
	notes.txt	2020-05-05 20:34:05	0.00MB	a40835254450bd55cb1c9a1895939357

selenium 라이브러리의 webdriver

크롬 드라이버 활용하기

■ 크롬 드라이버 실행

- `driver = webdriver.Chrome('g:/tool/chromedriver.exe')` <= 폴더 위치는 다를 수 있음
- 크롬 브라우저를 driver 라는 이름으로 저장



selenium 라이브러리의 webdriver

웹 페이지 접속

- `driver.get(URL)` : 특정 URL에 접속
- 사이트 접속하거나 클릭하는 등의 작업 후 해당 홈페이지를 다 받을 때 까지 대기 시간 필요

- 다음 홈 페이지 접속

```
url = 'https://www.daum.net/'
```

```
driver.get(url)
```

- 웹 페이지 html 다운로드

```
html = driver.page_source
```

```
print(html)
```

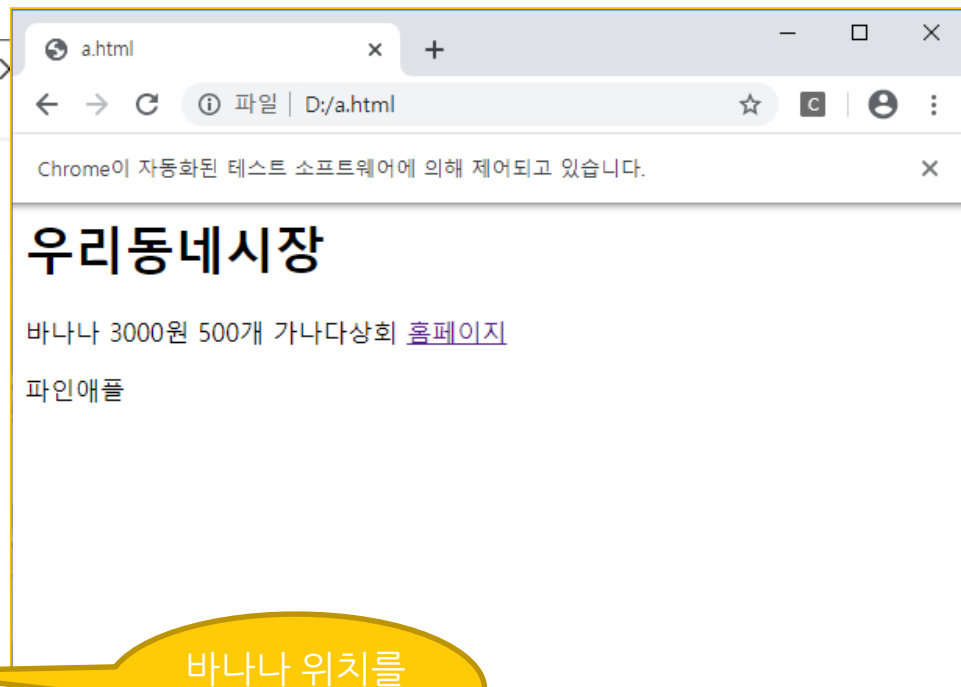
```
<html lang="ko" class="os_window"><head>
<meta charset="utf-8">
<title>Daum</title>
<meta property="og:url" content="https://www.daum.net/">
<meta property="og:type" content="website">
<meta property="og:title" content="Daum">
<meta property="og:image" content="//il.daumcdn.net/svc/image/U03/common_icon/5587C4E4012FCD0001">
<meta property="og:description" content="나의 관심 콘텐츠를 가장 즐겁게 볼 수 있는 Daum">
<meta name="msapplication-task" content="name=Daum;action-uri=https://www.daum.net/icon-uri=/favicon.ico">
<meta name="msapplication-task" content="name=미디어다음;action-uri=https://news.daum.net/icon-uri=/media_favicon.ico">
<meta name="msapplication-task" content="name=메일;action-uri=http://mail.daum.net/icon-uri=/mail_favicon.ico">
<meta name="referrer" content="origin">
<link rel="search" type="application/opensearchdescription+xml" href="//search.daum.net/OpenSearch.xml" title="다음">
<style type="text/css">
@charset "utf-8";
body,div,dl,dt,dd,ul,ol,li,h1,h2,h3,h4,h5,h6,pre,code,form,fieldset,legend,textarea,p,blockquote,th,td,input,select,button{margin:0;padding:0}
fieldset,img{border:0 none}
dl,ul,ol,menu,li{list-style:none}
```

HTML 구조

- 페이지 다운로드 후 필요한 정보 위치 파악 필요
- 즉, 특정 정보가 존재하는 위치를 지정 가능해야 함
- 규칙
 - 시작과 끝 존재
 - <태그>.....</태그>
 - div, p, span, a, tr, td
 - 태그는 다른 태그에 속할 수 있음 : 시작과 끝 사이에 다른 태그 존재 가능
 - 태그의 시작과 끝 사이에 화면에 표시되는 정보 존재
 - 태그가 속성을 가질 수 있음
 - <태그 속성1=값1 속성2=값2>Hello</태그>

HTML 예

```
a.html - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말
<html>
<head>
</head>
<body>
  <h1>우리동네시장</h1>
  <div class = 'sale'>
    <p id='fruits1' class='fruits'>
      <span class = 'name'>바나나</span>
      <span class = 'price'>3000원</span>
      <span class = 'inventory'>500개</span>
      <span class = 'store'>가나다상회</span>
      <a href = 'https://map.kakao.com/'>홈페이지</a>
    </p>
  </div>
  <div class = 'prepare'>
    <p id='fruits2' class='fruits'>
      <span class = 'name'>파인애플</span>
    </p>
  </div>
</body>
</html>
```



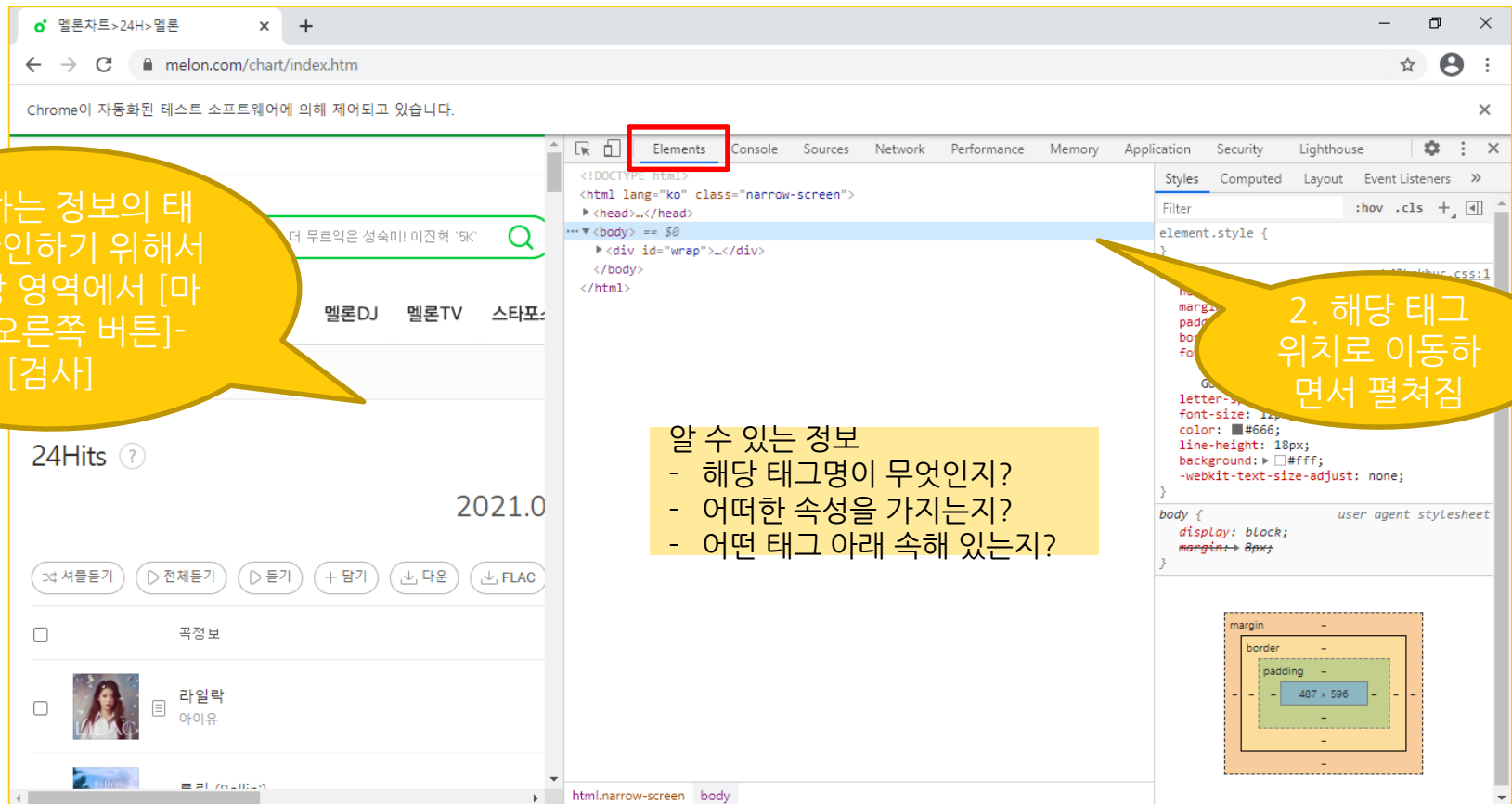
바나나 위치를
찾고 싶다

span 태그에 속함
class 명이 name임
id 값이 fruits1인 태그 아래에 있음

크롬 브라우저에서 웹 페이지의 HTML 확인

- 원하는 정보를 찾기 위해서는 크롬 개발자 도구 활용
- www.melon.com/chart/index.htm
- 브라우저의 메뉴버튼-[도구 더보기]-[개발자도구] 또는 F12 키 또는 마우스 오른쪽 버튼-[검사]

1. 원하는 정보의 태그를 확인하기 위해서는 해당 영역에서 [마우스 오른쪽 버튼]-[검사]



BeautifulSoup을 이용한 정보 찾기

- 실습용 html : a.html
 - a.html에서 ”은 여러 줄에 걸친 문자열 입력시 사용
 - 실제 홈페이지 html은 여러 문자열이 연결된 문자열 데이터
- BeautifulSoup 라이브러리
 - 문자열 데이터를 html 형식으로 읽고 정보를 쉽게 찾을 수 있음
 - html 문자열을 BeautifulSoup으로 해석하기
 - # bs4 패키지 내의 BeautifulSoup 라이브러리 불러오기
from bs4 import BeautifulSoup
 - html 변수에 들어있는 문자열 정보를 html 형식으로 해석
soup = BeautifulSoup(html, 'html.parser')

HTML 정보 찾기(1) - 태그 속성 활용

- BeautifulSoup 명령어 select() 사용
- select('조건') : html 내에서 입력한 조건을 만족하는 태그 모두 선택
- 조건
 - 해당 태그의 태그명이나 속성값 지정
 - 태그 간의 구조 지정
 - 두 방법 모두 활용

Beautifulsoup을 이용한 정보 찾기

■ 태그 선택을 위한 조건 입력 방법(1)

- 태그명과 해당 태그의 속성 정보 이용
- select(TAG) : html 범위 내에서 태그명이 TAG인 태그를 모두 선택
- 태그명으로 태그 찾기 예

```
tags_span = soup.select('span')
```

```
tags_p = soup.select('p')
```

```
1 print(tags_span)
```

```
[<span class="name"> 바나나 </span>, <span class="price"> 3000원 </span>, <span class="inventory"> 500개 </span>, <span class="store"> 가  
나다상회 </span>, <span class="name"> 파인애플 </span>]
```

```
<html>  
  <head>  
  </head>  
  <body>  
    <h1> 우리동네시장</h1>  
    <div class = 'sale'>  
      <p id='fruits1' class='fruits'>  
        <span class = 'name'> 바나나 </span>  
        <span class = 'price'> 3000원 </span>  
        <span class = 'inventory'> 500개 </span>  
        <span class = 'store'> 가나다상회 </span>  
        <a href = 'https://map.kakao.com/' > 홈페이지 </a>  
      </p>  
    </div>  
    <div class = 'prepare'>  
      <p id='fruits2' class='fruits'>  
        <span class = 'name'> 파인애플 </span>  
      </p>  
    </div>  
  </body>  
</html>
```

변수 html,
soup

```
1 print(tags_p)
```

```
[<p class="fruits" id="fruits1">  
  <span class="name"> 바나나 </span>  
  <span class="price"> 3000원 </span>  
  <span class="inventory"> 500개 </span>  
  <span class="store"> 가나다상회 </span>  
  <a href="https://map.kakao.com/"> 홈페이지 </a>  
</p>, <p class="fruits" id="fruits2">  
  <span class="name"> 파인애플 </span>  
</p>]
```

Beautifulsoup을 이용한 정보 찾기

■ 태그 선택을 위한 조건 입력 방법(2)

- id 값과 class 명을 이용해 태그 찾기 예
 - 조건 부분에 #id값 또는 .class명 넣어 찾기

```
ids_fruits1 = soup.select('#fruits1')
```

```
class_price = soup.select('.price')
```

```
tags_span_class_price = soup.select('span.price')
```

class 속성

- 글꼴, 배경색 등의 서식을 지정하기 위해 사용
- Html 내에서 동일한 class 명 여러 번 사용 가능

id 값

- 특정 대상을 지정하기 위해 사용
- Html 내에서 한 번만 사용
- 특정 태그 손쉽게 검색

```
<html>
<head>
</head>
<body>
  <h1> 우리동네시장</h1>
  <div class = 'sale'>
    <p id='fruits1' class='fruits'>
      <span class = 'name'> 바나나 </span>
      <span class = 'price'> 3000원 </span>
      <span class = 'inventory'> 500개 </span>
      <span class = 'store'> 가나다상회 </span>
      <a href = 'https://map.kakao.com/' > 홈페이지 </a>
    </p>
  </div>
  <div class = 'prepare'>
    <p id='fruits2' class='fruits'>
      <span class = 'name'> 파인애플 </span>
    </p>
  </div>
</body>
</html>
```

변수 html,
soup

```
1 print(ids_fruits1)
```

```
[<p class="fruits" id="fruits1">
  <span class="name"> 바나나 </span>
  <span class="price"> 3000원 </span>
  <span class="inventory"> 500개 </span>
  <span class="store"> 가나다상회 </span>
  <a href="https://map.kakao.com"> 홈페이지 </a>
</p>]
```

```
1 print(class_price)
```

```
[<span class="price"> 3000원 </span>]
```

```
1 print(tags_span_class_price)
```

```
[<span class="price"> 3000원 </span>]
```