

# 데이터 시각화 이해와 실습

## Lecture 03. 기온 공공데이터

동덕여자대학교  
데이터사이언스 전공  
권 범

# 목차

- ❖ 01. 기온 데이터 분석 시작하기
- ❖ 02. 서울의 기온 데이터 분석하기
- ❖ 03. 서울이 가장 더웠던 날은 언제였을까

# 01. 기온 데이터 분석 시작하기

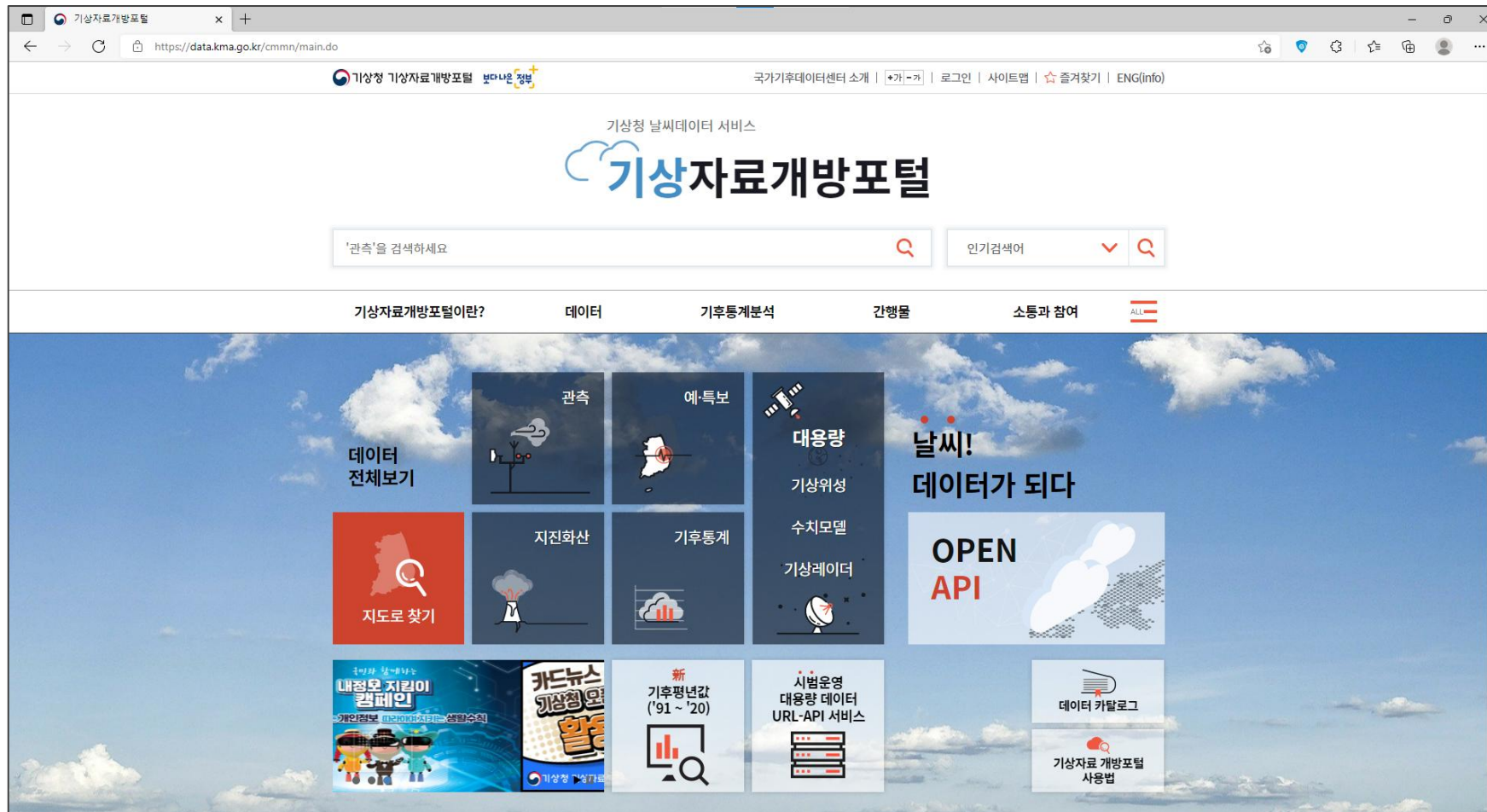
02. 서울의 기온 데이터 분석하기

03. 서울이 가장 더웠던 날은 언제 였을까

# 01. 기온 데이터 분석 시작하기

## ❖ ① 기온 공공데이터 살펴보기 (1/5)

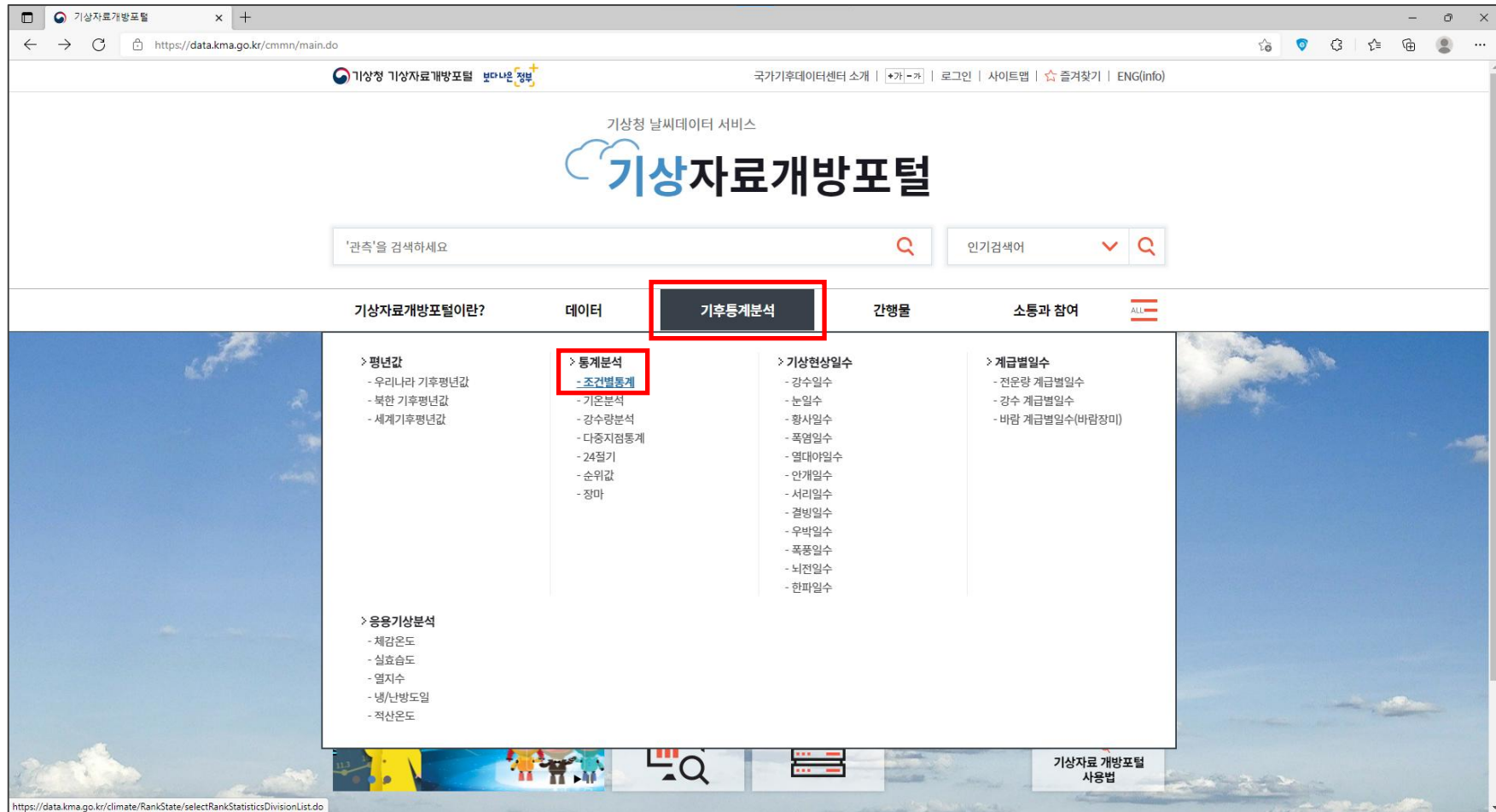
- 기상 관련 데이터 수집 → 기상자료개방포털(<https://data.kma.go.kr/>)



# 01. 기온 데이터 분석 시작하기

## ❖ ① 기온 공공데이터 살펴보기 (2/5)

- [기후통계분석] → [조건별통계] 버튼 클릭



# 01. 기온 데이터 분석 시작하기

## ❖ ① 기온 공공데이터 살펴보기 (3/5)

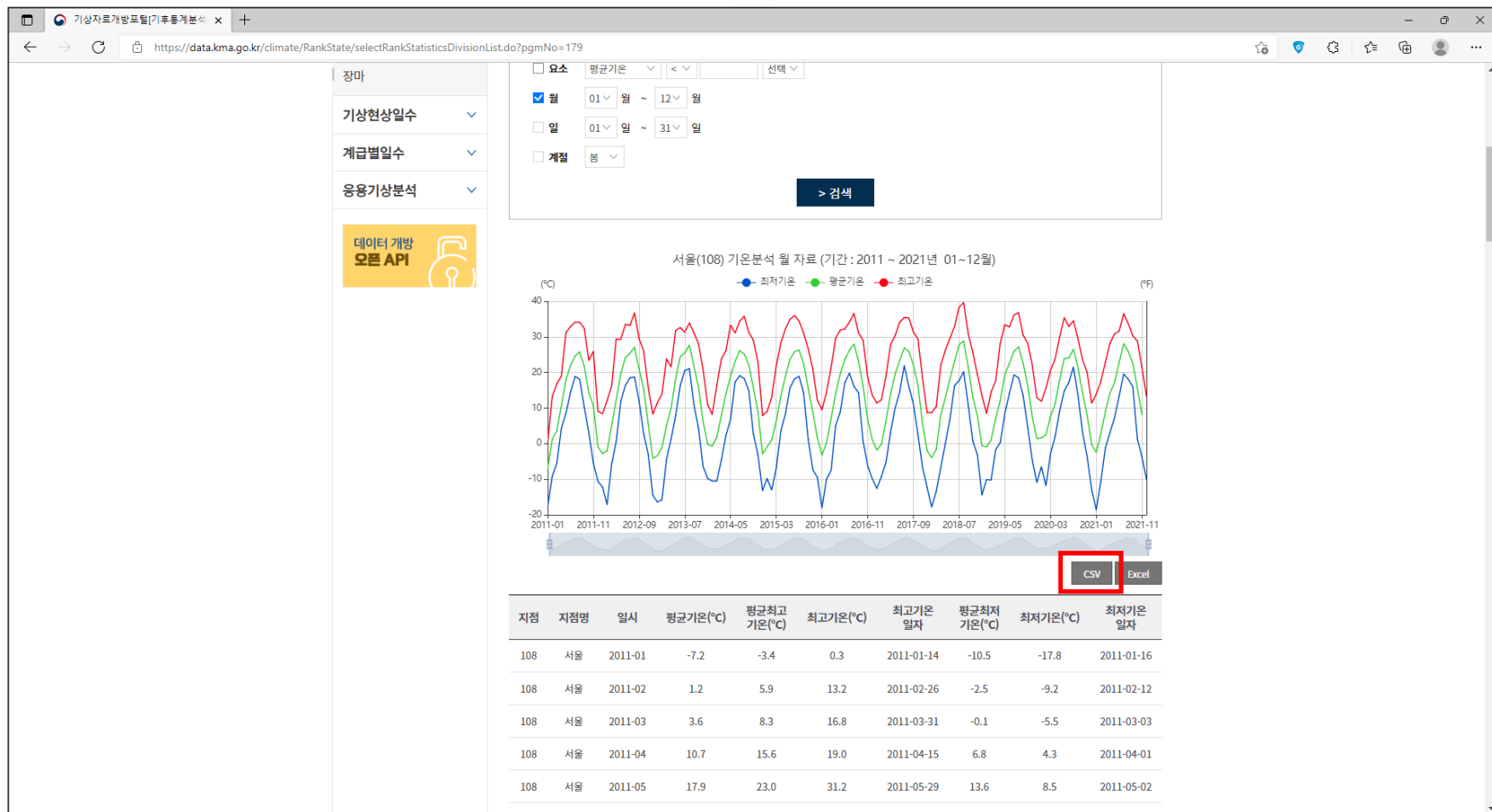
- [검색조건]에서 조건을 지정하고 [검색] 버튼 클릭

The screenshot shows the '기온통계분석' (Temperature Statistics Analysis) page on the KMA Climate Data Portal. The page is titled '조건별통계' (Statistics by Condition). The left sidebar contains a menu with options like '평년값', '통계분석', '조건별통계', '기온분석', '강수량분석', '다중지점통계', '24절기', '순위값', '장마', '기상현상일수', '계급별일수', and '응용기상분석'. The main content area is divided into sections: '자료설명' (Data Description) and '검색조건' (Search Conditions). The '검색조건' section is highlighted with a red box. It contains several dropdown menus and checkboxes for specifying search criteria. The '분류' (Classification) dropdown is set to '기온' (Temperature). The '지역/지점' (Region/Point) dropdown is set to '서울' (Seoul). The '기간' (Period) dropdown is set to '2011' ~ '2021'. The '조건' (Condition) section has checkboxes for '요소' (Element), '월' (Month), '일' (Day), and '계절' (Season). The '월' checkbox is checked, and the date range is set to '01' ~ '12'. The '계절' dropdown is set to '봄' (Spring). A red box highlights the '> 검색' (Search) button at the bottom right of the search conditions section.

# 01. 기온 데이터 분석 시작하기

## ❖ ① 기온 공공데이터 살펴보기 (4/5)

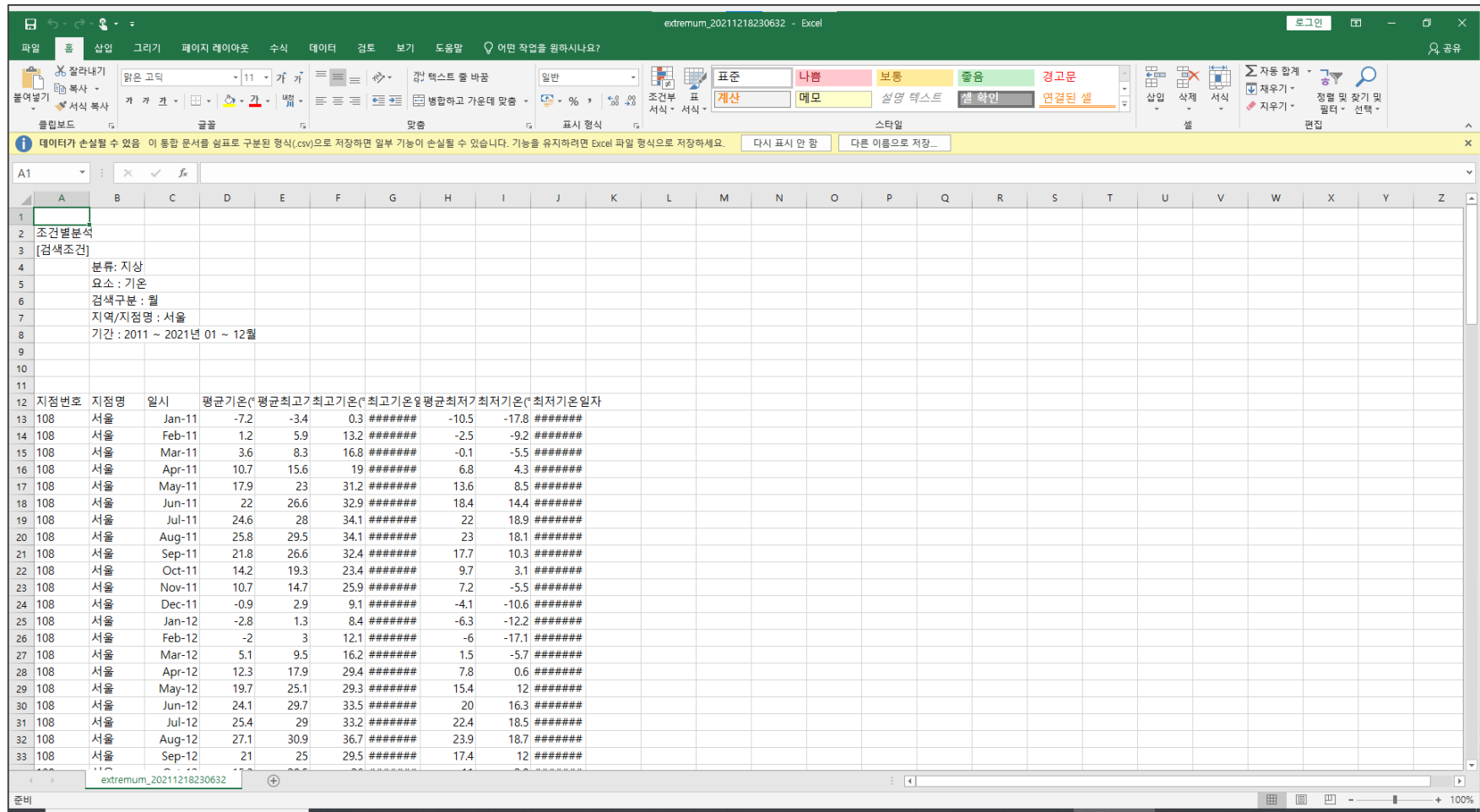
- 그래프 오른쪽 아래쪽에 위치한 [CSV] 버튼 클릭



# 01. 기온 데이터 분석 시작하기

## ❖ ① 기온 공공데이터 살펴보기 (5/5)

- 다운로드(Download) 폴더에 저장된 CSV 파일 열기



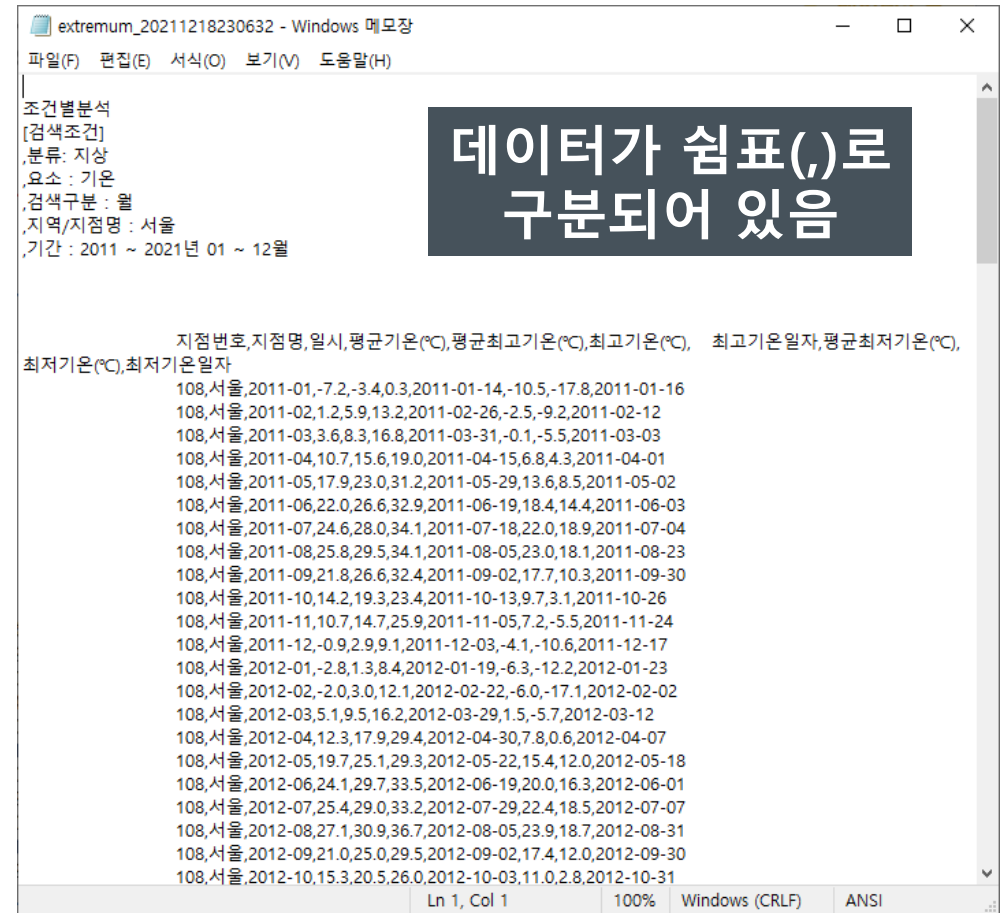
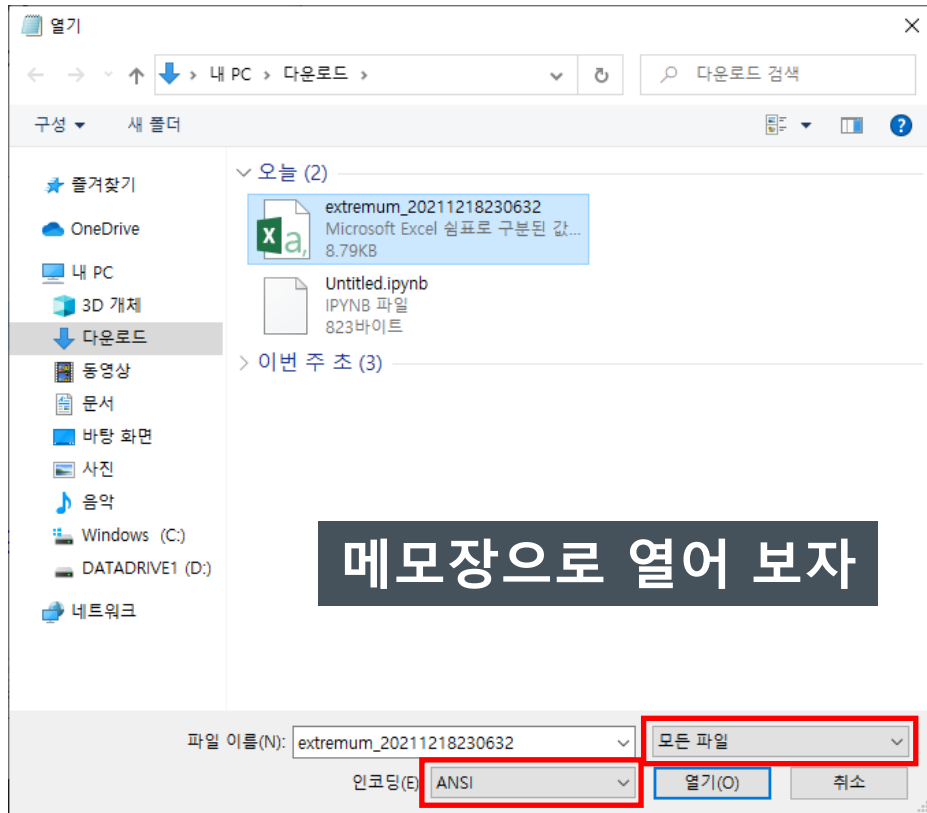
지점번호	지점명	일시	평균기온	평균최고기온	최고기온	최저기온	평균최저기온	최저기온	최저기온일자
108	서울	Jan-11	-7.2	-3.4	0.3	-10.5	-17.8		
108	서울	Feb-11	1.2	5.9	13.2	-2.5	-9.2		
108	서울	Mar-11	3.6	8.3	16.8	-0.1	-5.5		
108	서울	Apr-11	10.7	15.6	19	6.8	4.3		
108	서울	May-11	17.9	23	31.2	13.6	8.5		
108	서울	Jun-11	22	26.6	32.9	18.4	14.4		
108	서울	Jul-11	24.6	28	34.1	22	18.9		
108	서울	Aug-11	25.8	29.5	34.1	23	18.1		
108	서울	Sep-11	21.8	26.6	32.4	17.7	10.3		
108	서울	Oct-11	14.2	19.3	23.4	9.7	3.1		
108	서울	Nov-11	10.7	14.7	25.9	7.2	-5.5		
108	서울	Dec-11	-0.9	2.9	9.1	-4.1	-10.6		
108	서울	Jan-12	-2.8	1.3	8.4	-6.3	-12.2		
108	서울	Feb-12	-2	3	12.1	-6	-17.1		
108	서울	Mar-12	5.1	9.5	16.2	1.5	-5.7		
108	서울	Apr-12	12.3	17.9	29.4	7.8	0.6		
108	서울	May-12	19.7	25.1	29.3	15.4	12		
108	서울	Jun-12	24.1	29.7	33.5	20	16.3		
108	서울	Jul-12	25.4	29	33.2	22.4	18.5		
108	서울	Aug-12	27.1	30.9	36.7	23.9	18.7		
108	서울	Sep-12	21	25	29.5	17.4	12		



# 01. 기온 데이터 분석 시작하기

## ❖ ② CSV 파일이란?

- CSV(=Comma-Separated Values)
- 각 데이터를 쉼표(,)로 구분하여 저장하는 파일 형식



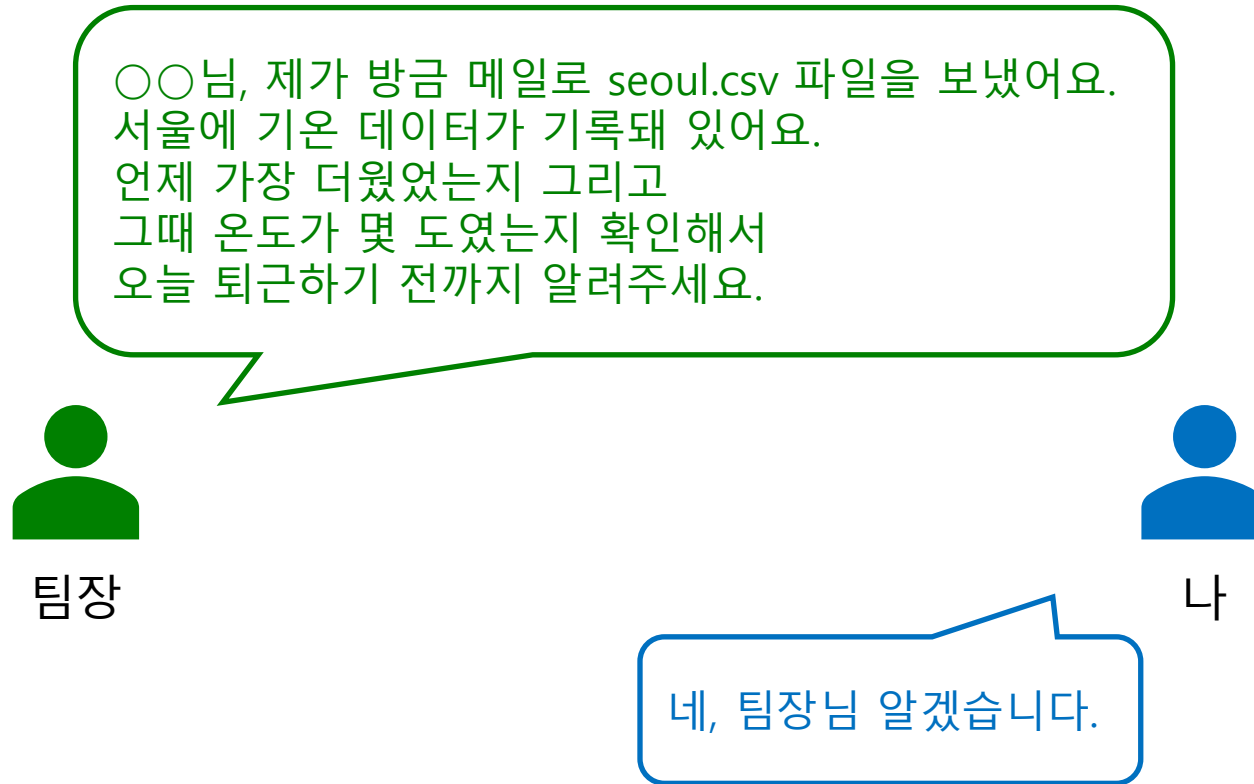
## 02. 서울의 기온 데이터 분석하기

01. 기온 데이터 분석 시작하기

03. 서울이 가장 더웠던 날은 언제 였을까

## 02. 서울의 기온 데이터 분석하기

### ❖ 상황 가정



## 02. 서울의 기온 데이터 분석하기

### ❖ CSV 파일 다운로드

- 기상자료개방포털(<https://data.kma.go.kr/>)
- [기후통계분석] - [기온분석] - [검색조건 설정] - [CSV 다운로드]

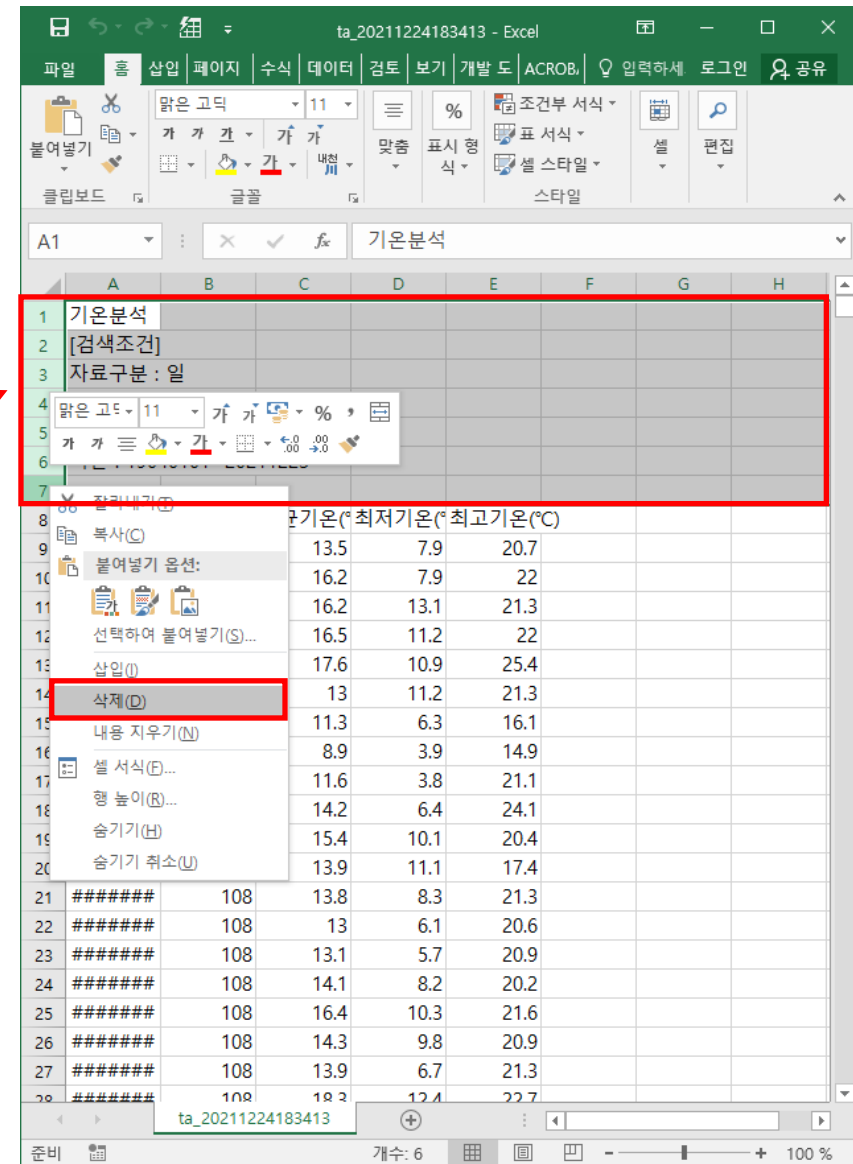
The screenshot displays the '기후통계분석' (Climate Statistics Analysis) page on the '기상자료개방포털' (KMA Data Open Portal). The left sidebar contains navigation options: '평년값', '통계분석', '조건별통계', '기온분석' (highlighted with a red box), '강수량분석', '다중지점통계', '24절기', '순위값', '장마', '기상현상일수', '계급별일수', and '응용기상분석'. The main content area is titled '기온분석 - 그래프' (Temperature Analysis - Graph). It includes a '자료설명' (Data Description) section with text about the data source and a '검색조건' (Search Conditions) section. In the search conditions, '자료구분' is set to '일' (Daily), '자료형태' is '기온' (Temperature), '기간' (Period) is '19040101 ~ 20211223', and '지역/지점' (Region/Point) is '서울' (Seoul) (all highlighted with red boxes). A '> 검색' (Search) button is also highlighted. Below the search conditions, there are 'CSV' and 'Excel' download buttons (the CSV button is highlighted with a red box). The bottom of the page shows a line graph titled '기온분석 기본 서울(108) 일자료 기간: 19040101 ~ 20211223' (Temperature Analysis Basic Seoul(108) Daily Data Period: 19040101 ~ 20211223) with a legend for '최저기온' (Minimum Temperature), '평균기온' (Average Temperature), and '최고기온' (Maximum Temperature).

## 02. 서울의 기온 데이터 분석하기

### ❖ CSV 파일 편집

- 다운로드한 CSV 파일을 엑셀 프로그램으로 열고,  
데이터 분석에 불필요한 1~7행을 삭제함

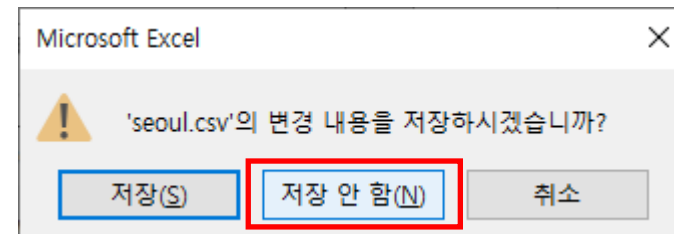
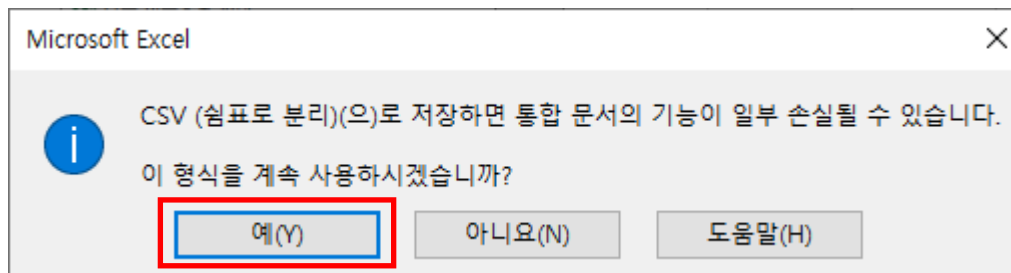
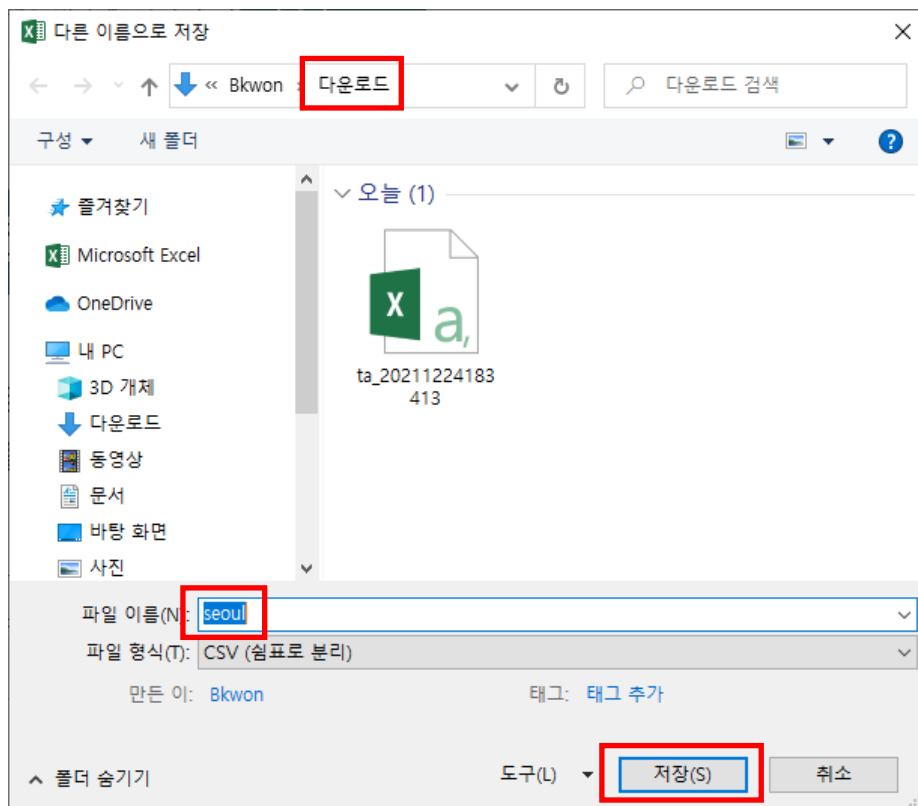
삭제



## 02. 서울의 기온 데이터 분석하기

### ❖ CSV 파일 저장

- 메뉴에서 [파일] – [다른 이름으로 저장]을 눌러 파일 이름을 seoul로 변경하고, 다운로드 폴더에 저장함(이때 경고창이 뜨면 [예(Y)] 버튼을 클릭)
- 완료되면 엑셀을 닫음(이때 저장을 묻는 창이 뜨면 [저장 안 함(N)] 버튼을 클릭)



## 02. 서울의 기온 데이터 분석하기

### ❖ ① CSV 파일에서 데이터 읽어 오기

- 노트북의 빈 셀(Cell) 에 다음과 같이 코드를 작성함

```
1 import csv
2
3 f = open("seoul.csv", 'r', encoding="cp949")
4 data = csv.reader(f)
5
6 print(data)
7 f.close()
```

- ✓ CSV 파일을 Windows 운영체제에서 만들었다면, cp949를 입력함
- ✓ macOS나 Linux에서 만든 파일일 때는, utf-8을 입력해야 오류가 발생하지 않음

#### 실행결과

```
<_csv.reader object at 0x78d1de6325e0>
```

만약 Windows가 아닌 다른 운영체제(macOS, Linux 등)를 사용하고 있거나,  
코랩(Colab) 사용자는 encoding="cp949"를 반드시 입력해야 함

코랩은 Linux 운영체제를 사용함

## 02. 서울의 기온 데이터 분석하기

### ❖ ② 데이터 출력하기 (1/6)

- 셀(Cell)을 하나 추가하여 아래의 코드 내용을 작성하고 실행함

```
1 import csv
2
3 f = open("seoul.csv", 'r', encoding="cp949")
4 data = csv.reader(f)
5
6 for row in data:
7     print(row)
8
9 f.close()
```

#### 실행결과

```
[ '날짜', '지점', '평균기온(°C)', '최저기온(°C)', '최고기온(°C)' ]
[ '1907-10-01', '108', '13.5', '7.9', '20.7' ]
[ '1907-10-02', '108', '16.2', '7.9', '22' ]
[ '1907-10-03', '108', '16.2', '13.1', '21.3' ]
... (중략) ...
```



## 02. 서울의 기온 데이터 분석하기

### ❖ ② 데이터 출력하기 (2/6)

- 출력된 결과화면을 살펴보자

#### 실행결과

```
[ '1950-08-29', '108', '23.1', '16.8', '30.4' ]  
[ '1950-08-30', '108', '24.6', '18', '32.6' ]  
[ '1950-08-31', '108', '25.4', '20.1', '32.5' ]  
[ '1950-09-01', '108', '', '', '' ]  
[ '1950-09-02', '108', '', '', '' ]  
[ '1950-09-03', '108', '', '', '' ]  
... (중략) ...  
[ '1953-11-27', '108', '', '', '' ]  
[ '1953-11-28', '108', '', '', '' ]  
[ '1953-11-29', '108', '', '', '' ]  
[ '1953-11-30', '108', '', '', '' ]  
[ '1953-12-01', '108', '12.2', '7.4', '16.2' ]  
[ '1953-12-02', '108', '5.6', '1.4', '9' ]  
[ '1953-12-03', '108', '0.7', '-4.2', '4.6' ]
```

누락된 데이터가 있음  
6.25전쟁(1950년 6월 25일~1953년 7월 27일)

## 02. 서울의 기온 데이터 분석하기

### ❖ ② 데이터 출력하기 (3/6)

- 출력된 결과화면을 살펴보자

#### 실행결과

```
... (중략) ...  
['2017-10-08', '108', '23', '19.3', '28.7']  
['2017-10-09', '108', '22.5', '19.8', '27.6']  
['2017-10-10', '108', '21.4', '18.6', '24.8']  
['2017-10-11', '108', '15.5', '12.2', '21.7']  
['2017-10-12', '108', '11.4', '8.8', '']  
['2017-10-13', '108', '12.8', '6.1', '18.9']  
['2017-10-14', '108', '14.4', '9', '20.5']  
['2017-10-15', '108', '15.8', '9', '23']  
['2017-10-16', '108', '16.6', '13.6', '22']  
... (중략) ...
```

2017년 10월 12일의 최고기온 데이터도 누락됨

## 02. 서울의 기온 데이터 분석하기

### ❖ ② 데이터 출력하기 (4/6)

- 앞에서 살펴본 것처럼 전체 데이터에서 누락된 값(=결측치, Missing Value)이 있는지 여부를 데이터 분석 전에 확인해 보는 습관을 갖도록 하자

#### 결측치가 있는지 어떻게 확인할 수 있나요?

- ✓ `pandas.DataFrame.info()`, `pandas.DataFrame.isna()`와 같은 함수들을 이용할 수 있음
- ✓ 빈 문자열("")을 확인하기 위해서는 사용자가 별도로 관련 기능을 구현해야 함

- NA: Not Available
- NaN: Not a Number

#### 만약 결측치가 있다는 것이 확인되면, 결측치를 어떻게 처리해야 할까?

- ✓ (결측치 대체) 해당 결측치를 평균 값이나 바로 앞(또는 뒤) 데이터 값으로 대체하는 등 여러 방법들이 존재함
- ✓ (결측치 제거) 결측치가 존재하는 행(Row) 또는 열(Column)을 제거함

## 02. 서울의 기온 데이터 분석하기

### ❖ ② 데이터 출력하기 (5/6)

- 서울 기온 데이터에는 결측치가 빈 문자열("") 형태로 존재하니, 결측치를 확인할 수 있는 기능을 아래와 같이 구현해 보자

```
1 import csv
2
3 f = open("seoul.csv", 'r', encoding="cp949")
4 data = csv.reader(f)
5
6 for row in data:
7     if '' in row:
8         print(row)
9
10 f.close()
```

## 02. 서울의 기온 데이터 분석하기

### ❖ ② 데이터 출력하기 (6/6)

- 실행하여 보면, 아래와 같이 결측치를 포함하는 데이터들만 출력됨

#### 실행결과

```
... (중략) ...  
['1953-11-28', '108', '', '', '']  
['1953-11-29', '108', '', '', '']  
['1953-11-30', '108', '', '', '']  
['1967-02-19', '108', '-1.7', '', '']  
['1973-10-16', '108', '12.3', '', '']  
['2017-10-12', '108', '11.4', '8.8', '']  
['2022-08-08', '108', '26.8', '', '28.4']
```

## 02. 서울의 기온 데이터 분석하기

### ❖ ③ 헤더 저장하기 (1/2)

- 헤더(Header)란 데이터 파일에서 각 값이 어떤 의미를 갖는지 표시한 행(Row)을 의미함
- 헤더를 별도로 저장하기 위해서 `next()` 함수를 사용할 수 있음

◆ `next()` 함수

➤ 첫 번째 데이터 행을 읽어오면, 데이터의 탐색 위치를 다음 행으로 이동시킴

```
1 import csv
2
3 f = open("seoul.csv", 'r', encoding="cp949")
4 data = csv.reader(f)
5
6 header = next(data)
7 print(header)
8
9 f.close()
```

#### 실행결과

```
['날짜', '지점', '평균기온(°C)', '최저기온(°C)', '최고기온(°C)']
```

## 02. 서울의 기온 데이터 분석하기

### ❖ ③ 헤더 저장하기 (2/2)

- header = next(data) 코드가 있는 경우와 없는 경우의 출력을 비교하면, next( ) 함수의 기능을 보다 쉽게 이해할 수 있음

```
1 import csv
2
3 f = open("seoul.csv", 'r', encoding="cp949")
4 data = csv.reader(f)
5
6 header = next(data)
7 # print(header)
8
9 for row in data:
10     print(row)
11
12 f.close()
```

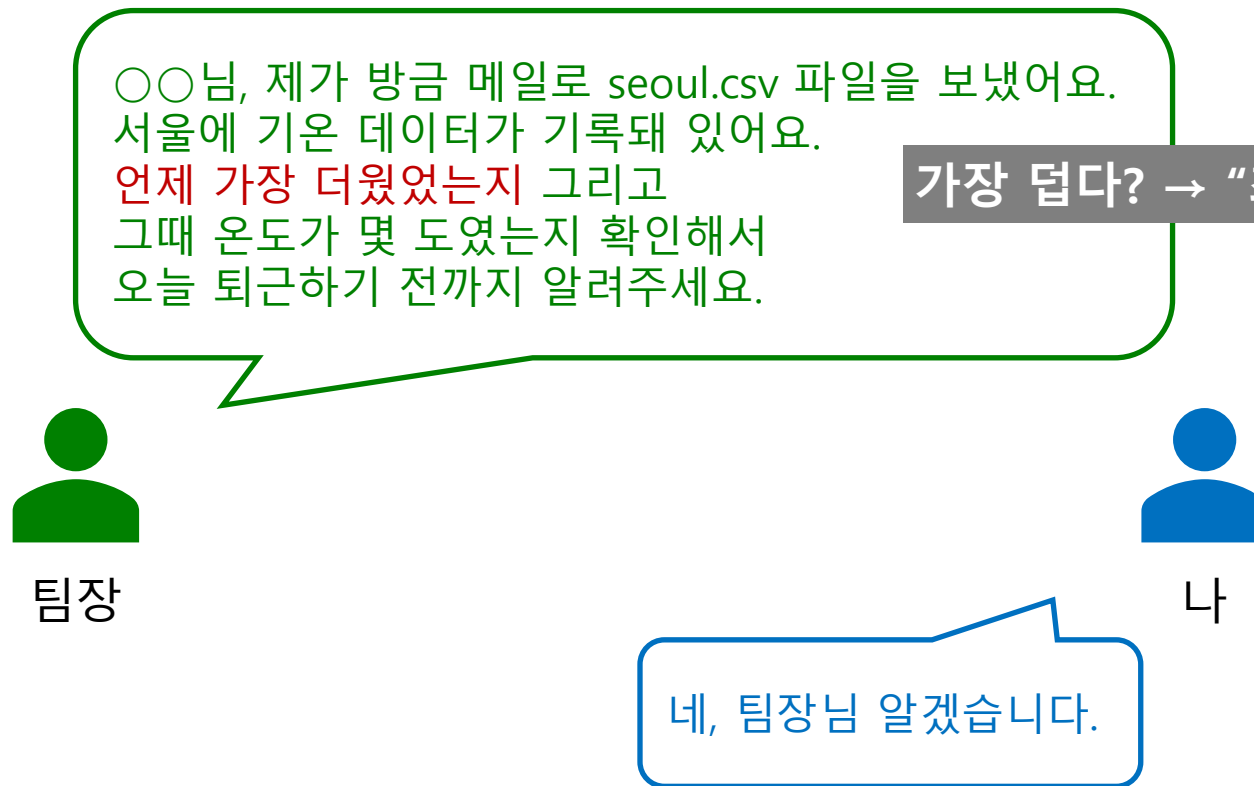
## 03. 서울이 가장 더웠던 날은 언제 었을까

- 01. 기온 데이터 분석 시작하기
- 02. 서울의 기온 데이터 분석하기



### 03. 서울이 가장 더웠던 날은 언제였을까

#### ❖ ① 질문 다듬기



기상 관측 이래, 서울의 최고 기온이 가장 높았던 날은 언제였고, 몇 도였을까?

## 03. 서울이 가장 더웠던 날은 언제 였을까

### ❖ ② 문제 해결 방법 구상하기

- 문제를 해결하기 위한 절차(=알고리즘, Algorithm)는?
  - ◆ Step 1) 데이터를 읽음
  - ◆ Step 2) 순차적으로 최고 기온을 확인함
  - ◆ Step 3) 최고 기온이 가장 높았던 날짜의 데이터를 저장함
  - ◆ Step 4) 최종 저장된 데이터를 출력함

### 03. 서울이 가장 더웠던 날은 언제였을까

#### ❖ ③ 파이썬 코드로 구현하기 (1/7)

- 데이터를 읽기 위해 다음과 같이 코드를 작성해 보자

```
1 import csv
2
3 f = open("seoul.csv", encoding="cp949")
4 data = csv.reader(f)
5 header = next(data)
6 print(header)
7
8 for row in data:
9     print(row)
10    break
11
12 f.close()
```

Windows 운영체제를 사용하고 있다면,  
encoding="cp949"는 생략할 수 있음

f = open("seoul.csv")라고 간략하게 작성할 수 있음

우리가 관심있는 "최고기온" 데이터는 리스트(List)의  
가장 마지막에 위치하고 있으며, 자료형이 문자열(String)임

#### 실행결과

```
[ '날짜', '지점', '평균기온(°C)', '최저기온(°C)', '최고기온(°C)' ]
[ '1907-10-01', '108', '13.5', '7.9', '20.7' ]
```

### 03. 서울이 가장 더웠던 날은 언제 었을까

#### ❖ ③ 파이썬 코드로 구현하기 (2/7)

- 최고 기온 값을 찾기 위해서는 기온이 높고 낮다는 대소 관계를 비교해야 함

```
1 import csv
2
3 f = open("seoul.csv", encoding="cp949")
4 data = csv.reader(f)
5 header = next(data)
6 print(header)
7
8 for row in data:
9     row[4] = float(row[4])
10    print(row)
11    break
12
13 f.close()
```

대소 관계를 비교하기 위해서는 어떻게 해야 할까?

자료형을 문자열(String)에서 실수(Float)로  
변환해 주어야 함 → float( ) 함수를 사용하면 됨

# row[4]와 row[-1]은 같음

- ✓ 작은 따옴표(')가 사라졌음
- ✓ 자료형이 더 이상 문자열이 아님

#### 실행결과

```
['날짜', '지점', '평균기온(°C)', '최저기온(°C)', '최고기온(°C)']
['1907-10-01', '108', '13.5', '7.9', 20.7]
```

## 03. 서울이 가장 더웠던 날은 언제 었을까

### ❖ ③ 파이썬 코드로 구현하기 (3/7)

- 서울 기온 데이터에는 빈 문자열("")로 표현되는 결측치(Missing Value)가 있었음
- break 명령어를 주석 처리하고, 코드를 실행하면 아래와 같이 오류가 발생함

#### 실행결과

... (중략) ...

```
['1950-08-30', '108', '24.6', '18', 32.6]
```

```
['1950-08-31', '108', '25.4', '20.1', 32.5]
```

-----  
ValueError Traceback (most recent call last)

<ipython-input-6-3445d25043a5> in <cell line: 8>()

7

8 for row in data:

----> 9 row[4] = float(row[4]) # row[4]와 row[-1]은 같음

10 print(row)

11 # break

ValueError: could not convert string to float: ''

빈 문자열("")을 어떤 실수 값으로  
바꿔야 할지 몰라서 오류가 발생한 것임

## 03. 서울이 가장 더웠던 날은 언제였을까

### ❖ ③ 파이썬 코드로 구현하기 (4/7)

- 결측치를 다른 값으로 대체하는 전략을 사용하자

```
1 import csv
2
3 f = open("seoul.csv", encoding="cp949")
4 data = csv.reader(f)
5 header = next(data)
6 print(header)
7
8 for row in data:
9     if row[4] == '':
10         row[4] = -999
11
12     row[4] = float(row[4])
13     print(row)
14
15 f.close()
```

최고 기온을 찾는 문제이므로, 최고 기온으로  
나오기 어려운 값인 -999으로 결측치를 대체하자

# 만약 최고기온 데이터가 빈 문자열이라면  
# -999를 대입

# row[4]와 row[-1]은 같음

## 03. 서울이 가장 더웠던 날은 언제 었을까

### ❖ ③ 파이썬 코드로 구현하기 (5/7)

- 코드를 다시 실행시켜보면, 오류 없이 수행됨

#### 실행결과

```
[ '날짜', '지점', '평균기온(°C)', '최저기온(°C)', '최고기온(°C)' ]  
[ '1907-10-01', '108', '13.5', '7.9', '20.7' ]  
... (중략) ...  
[ '1950-08-30', '108', '24.6', '18', 32.6 ]  
[ '1950-08-31', '108', '25.4', '20.1', 32.5 ]  
... (중략) ...  
[ '2024-02-05', '108', '2.7', '0.1', 6.6 ]  
[ '2024-02-06', '108', '1.2', '0', 2.9 ]  
[ '2024-02-07', '108', '0.6', '-0.9', 2.7 ]  
[ '2024-02-08', '108', '0.6', '-3', 6.2 ]  
... (중략) ...
```

## 03. 서울이 가장 더웠던 날은 언제였을까

### ❖ ③ 파이썬 코드로 구현하기 (6/7)

- 전체 코드를 완성해 보자

```
1 import csv
2
3 max_temp = -999          # 최고 기온 값을 저장할 변수
4 max_date = ''           # 최고 기온이 가장 높았던 날짜를 저장할 변수
5 f = open("seoul.csv", encoding="cp949")
6 data = csv.reader(f)
7 header = next(data)
8
9 for row in data:
10     if row[4] == '':
11         row[4] = -999
12
13     row[4] = float(row[4])
14     if max_temp < row[4]:
15         max_temp = row[4]
16         max_date = row[0]
17
18 f.close()
```

Step 1) 데이터를 읽음

Step 2) 순차적으로 최고 기온을 확인함

Step 3) 최고 기온이 가장 높았던 날짜의 데이터를 저장함

# 최고 기온 값 업데이트

# 최고 기온 날짜 업데이트



## 03. 서울이 가장 더웠던 날은 언제 였을까

### ❖ ③ 파이썬 코드로 구현하기 (7/7)

```
19 print("기상 관측 이래 서울의 최고 기온이 가장 높았던 날은",  
20       max_date + "로,", max_temp, "도 였습니다.")
```

Step 4) 최종 저장된 데이터를 출력함

#### 실행결과

기상 관측 이래 서울의 최고 기온이 가장 높았던 날은 2018-08-01로, 39.6 도 였습니다.

- ❖ 01. 기온 데이터 분석 시작하기
- ❖ 02. 서울의 기온 데이터 분석하기
- ❖ 03. 서울이 가장 더웠던 날은 언제 였을까

# THANK YOU!

## Q & A

- Name: 권범
- Office: 동덕여자대학교 인문관 B821호
- Phone: 02-940-4752
- E-mail: [bkwon@dongduk.ac.kr](mailto:bkwon@dongduk.ac.kr)