

# 데이터분석입문

## Lecture 04. 서울의 기온 데이터 분석하기

동양미래대학교  
인공지능소프트웨어학과  
강 환수

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

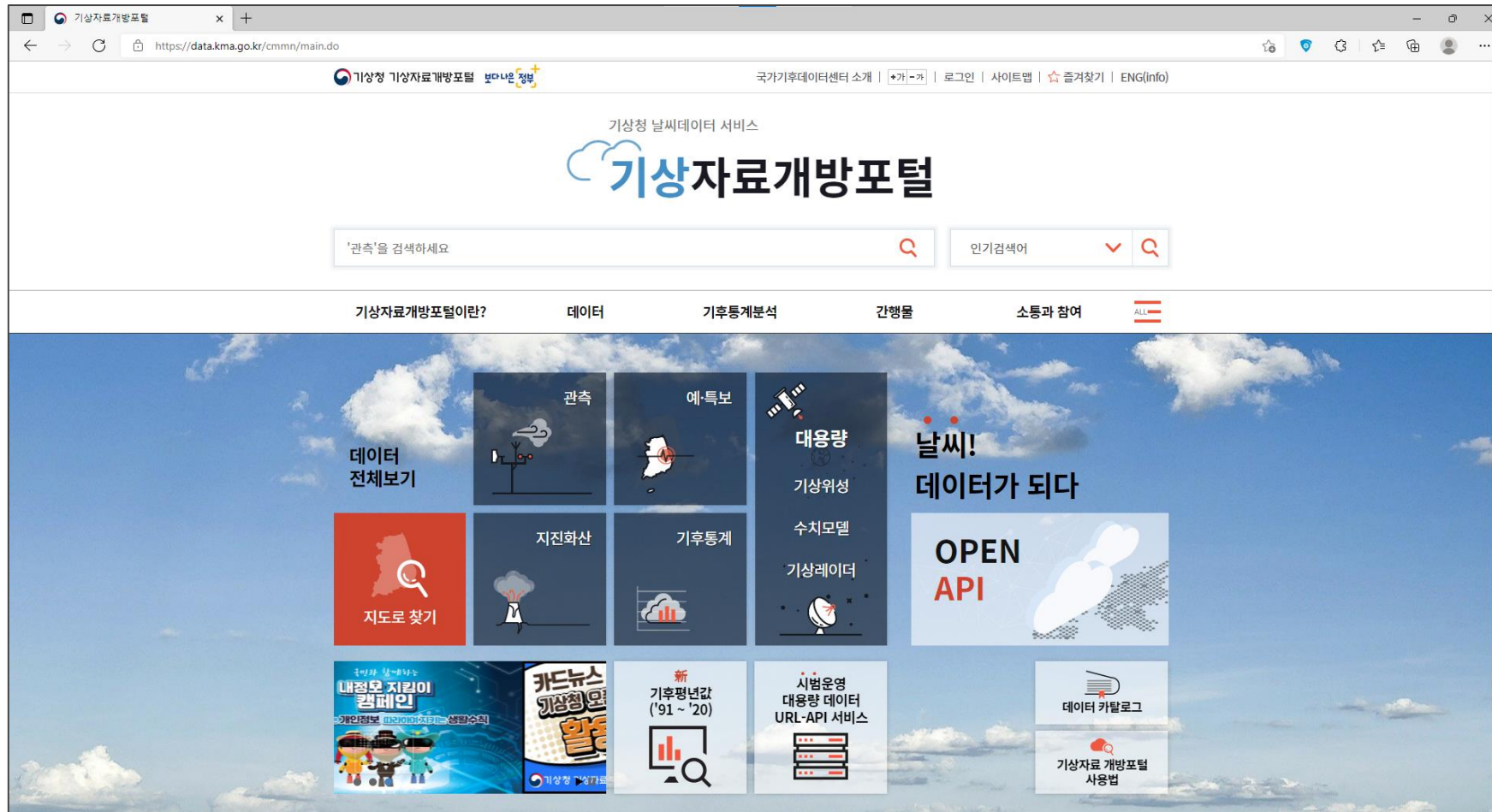
# 01. 기온 데이터 분석 시작하기

02. 서울의 기온 데이터 분석하기

03. 서울이 가장 더웠던 날은 언제 였을까

## ❖ ① 기온 공공데이터 살펴보기 (1/5)

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



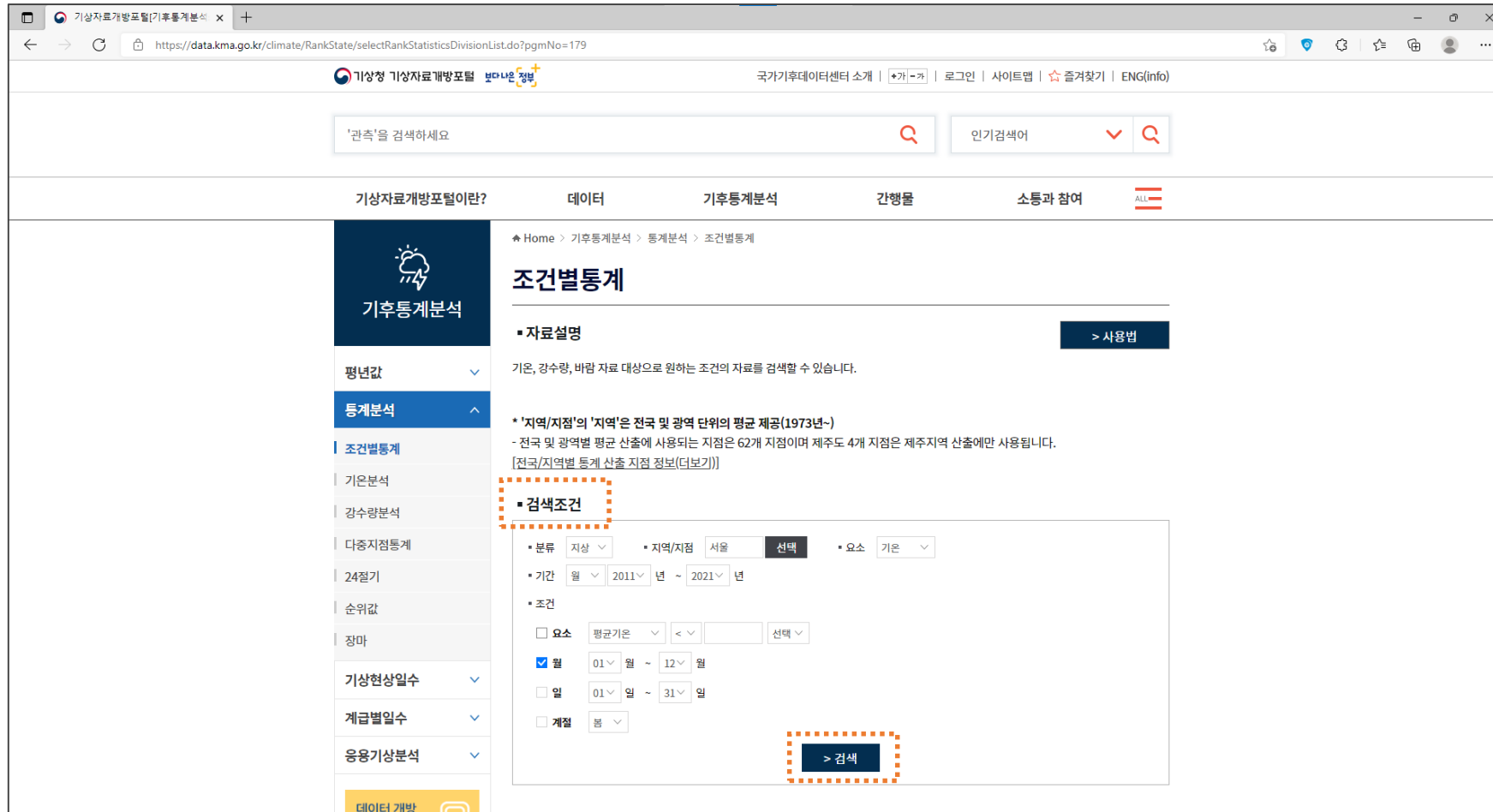
## ❖ ① 기온 공공데이터 살펴보기 (2/5)

- [기후통계분석] → [조건별통계] 버튼 클릭



## ❖ ① 기온 공공데이터 살펴보기 (3/5)

- 검색조건에서 조건을 지정하고 검색 버튼 클릭



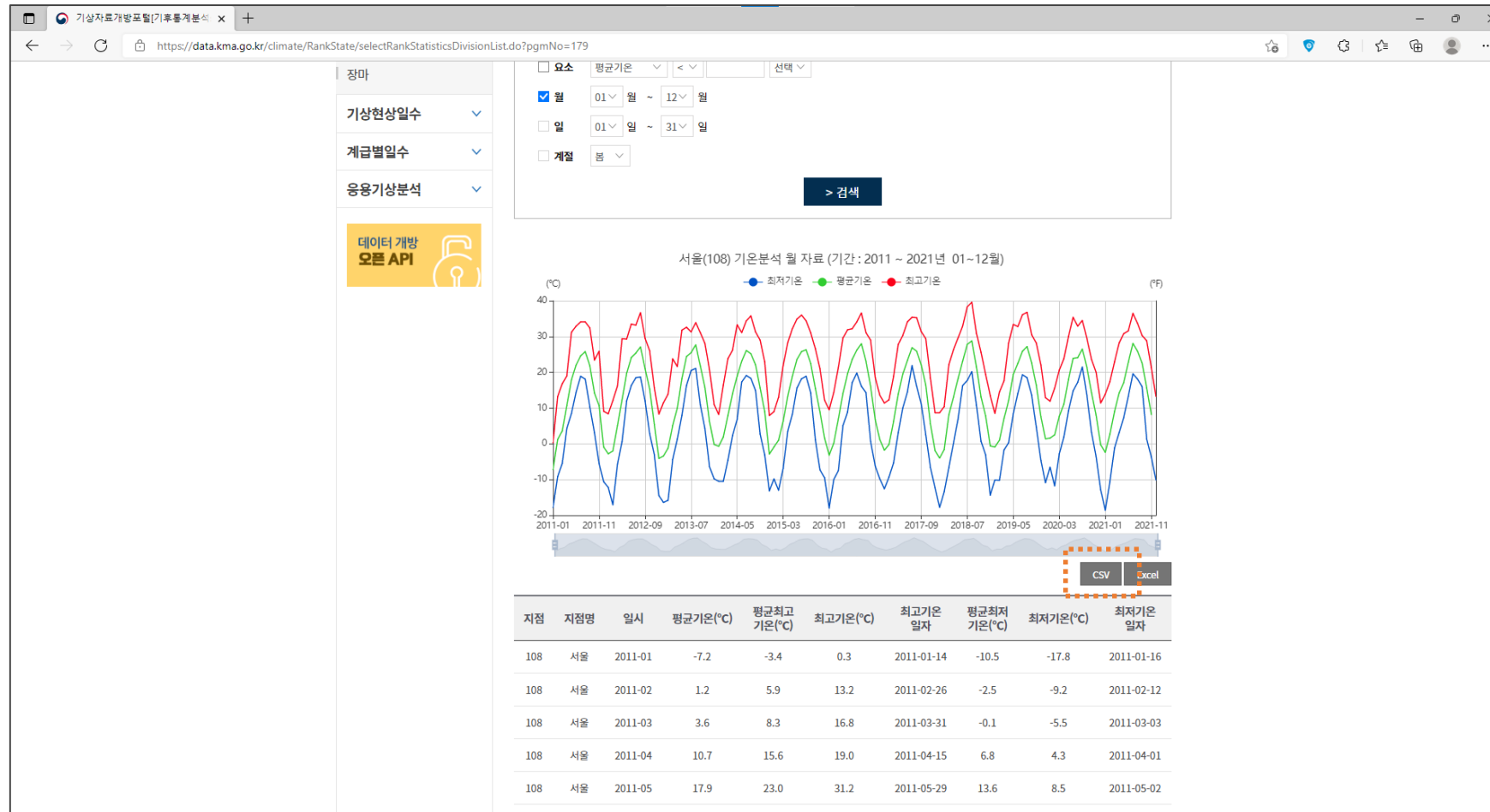
The screenshot shows the KMA Climate Data Portal interface. The main content area is titled '조건별통계' (Statistics by Condition). Under the '검색조건' (Search Conditions) section, the following settings are visible:

- 분류** (Classification): 기상 (Weather)
- 지역/지점** (Region/Point): 서울 (Seoul)
- 요소** (Element): 기온 (Temperature)
- 기간** (Period): 2011년 ~ 2021년
- 조건** (Condition):
  - ☐ 요소: 평균기온 (Average Temperature)
  - ☒ 월: 01월 ~ 12월
  - ☐ 일: 01일 ~ 31일
  - ☐ 계절: 봄

The '> 검색' (Search) button is highlighted with a red dashed box.

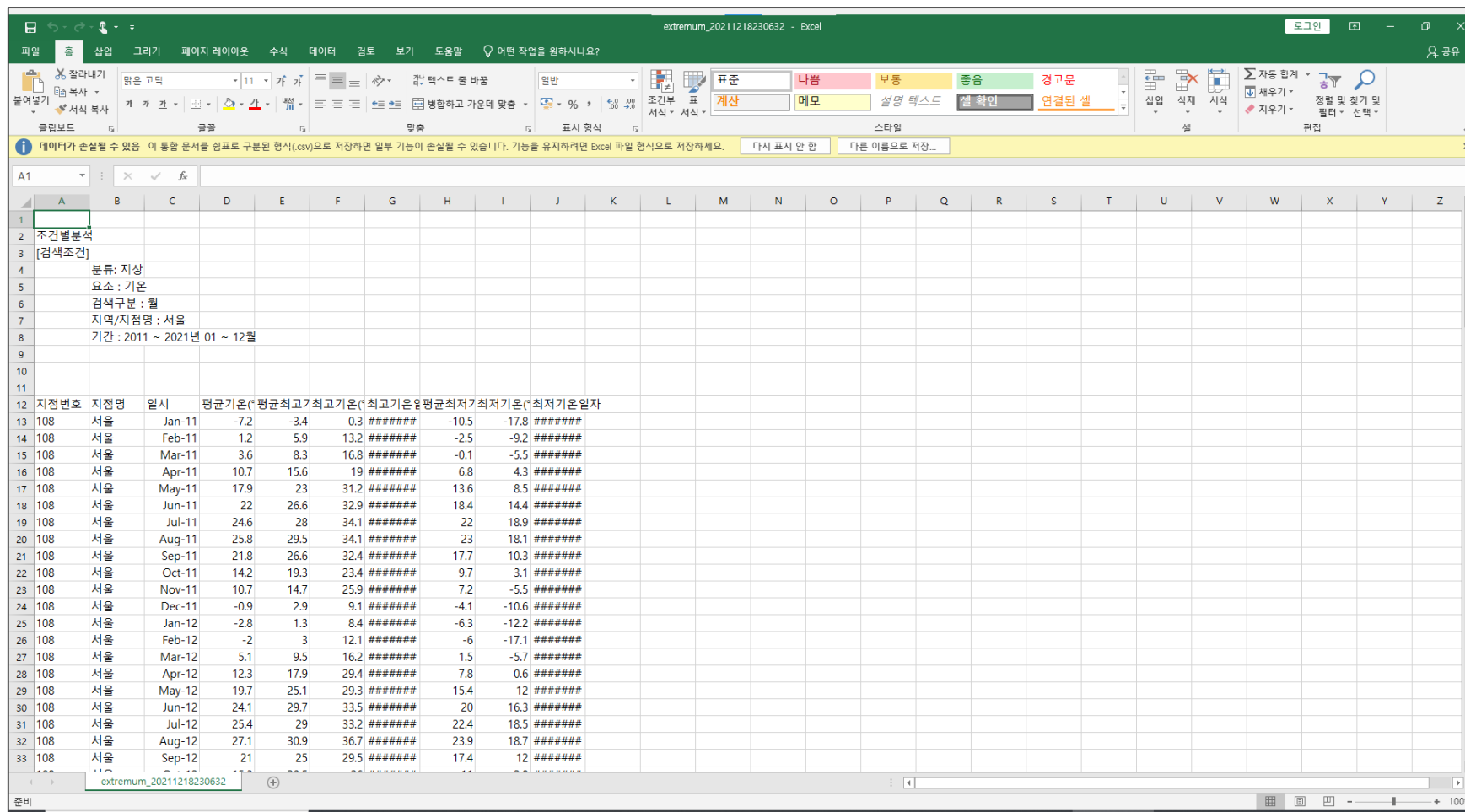
## ❖ ① 기온 공공데이터 살펴보기 (4/5)

- 그래프 오른쪽 아래쪽에 위치한 CSV 버튼 클릭



## ❖ ① 기온 공공데이터 살펴보기 (5/5)

- 다운로드 (Download) 폴더에 저장된 CSV 파일 열기

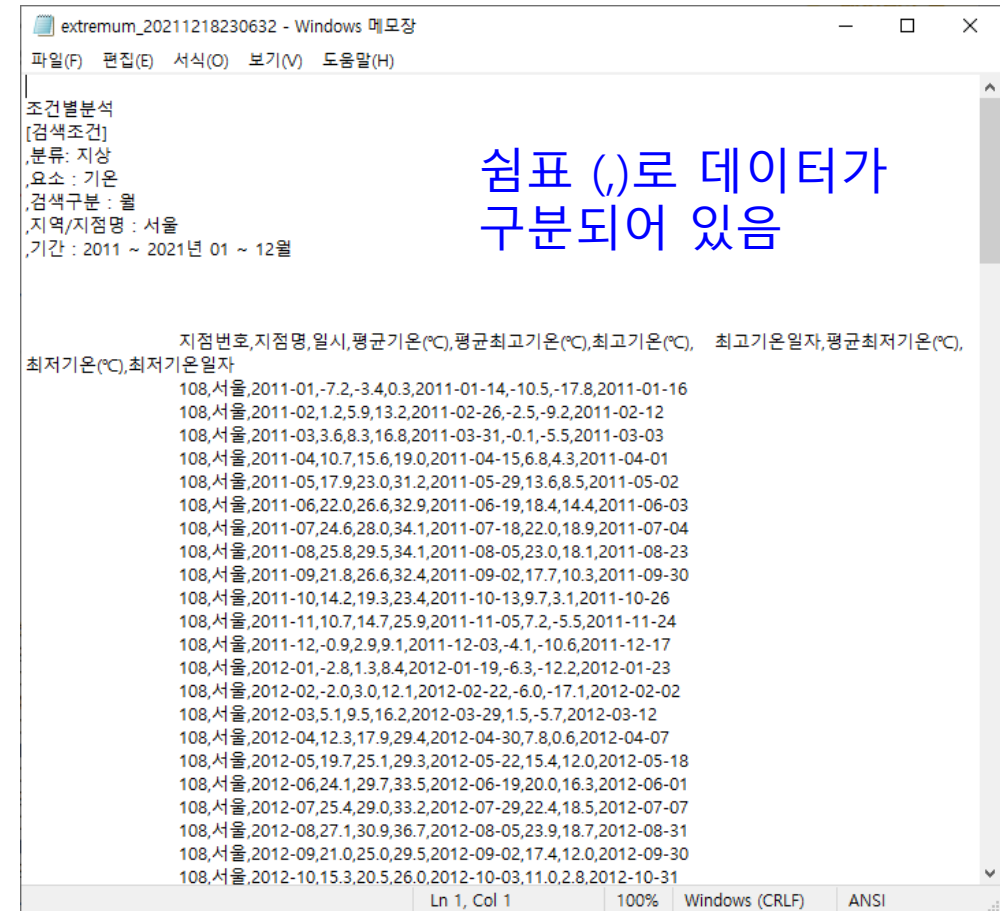
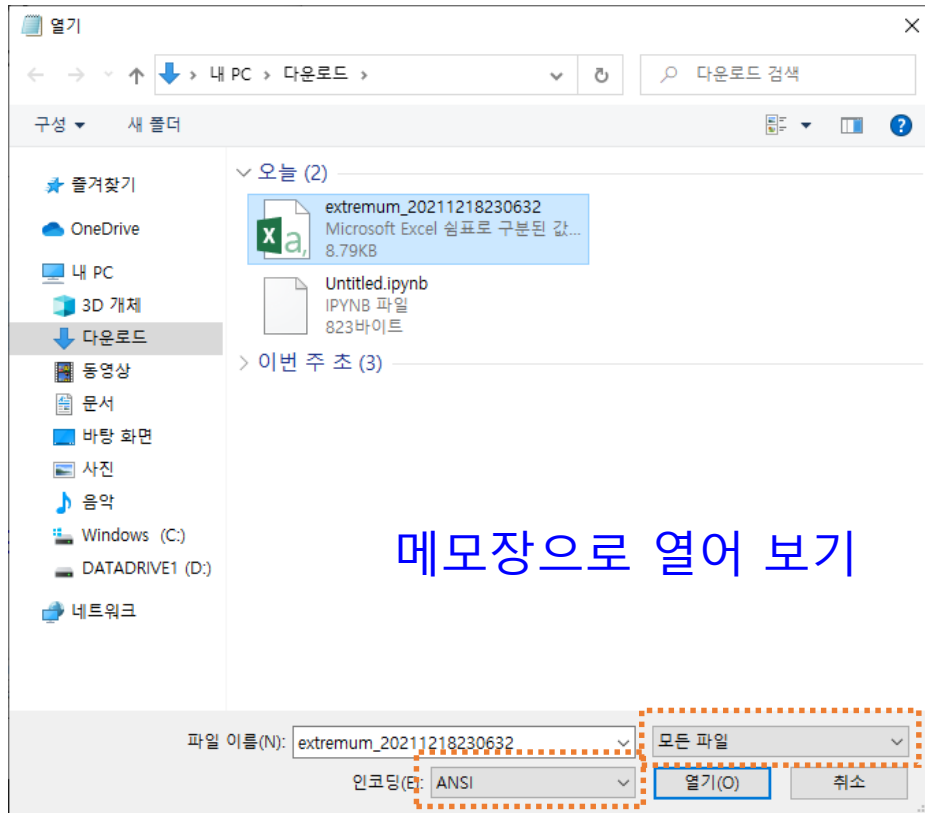


지점번호	지점명	일시	평균기온	평균평균기온	최고기온	최저기온	일평균기온	최저기온	최저기온일자
108	서울	Jan-11	-7.2	-3.4	0.3	#####	-10.5	-17.8	#####
108	서울	Feb-11	1.2	5.9	13.2	#####	-2.5	-9.2	#####
108	서울	Mar-11	3.6	8.3	16.8	#####	-0.1	-5.5	#####
108	서울	Apr-11	10.7	15.6	19	#####	6.8	4.3	#####
108	서울	May-11	17.9	23	31.2	#####	13.6	8.5	#####
108	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	#####



## ❖ ② CSV 파일이란?

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

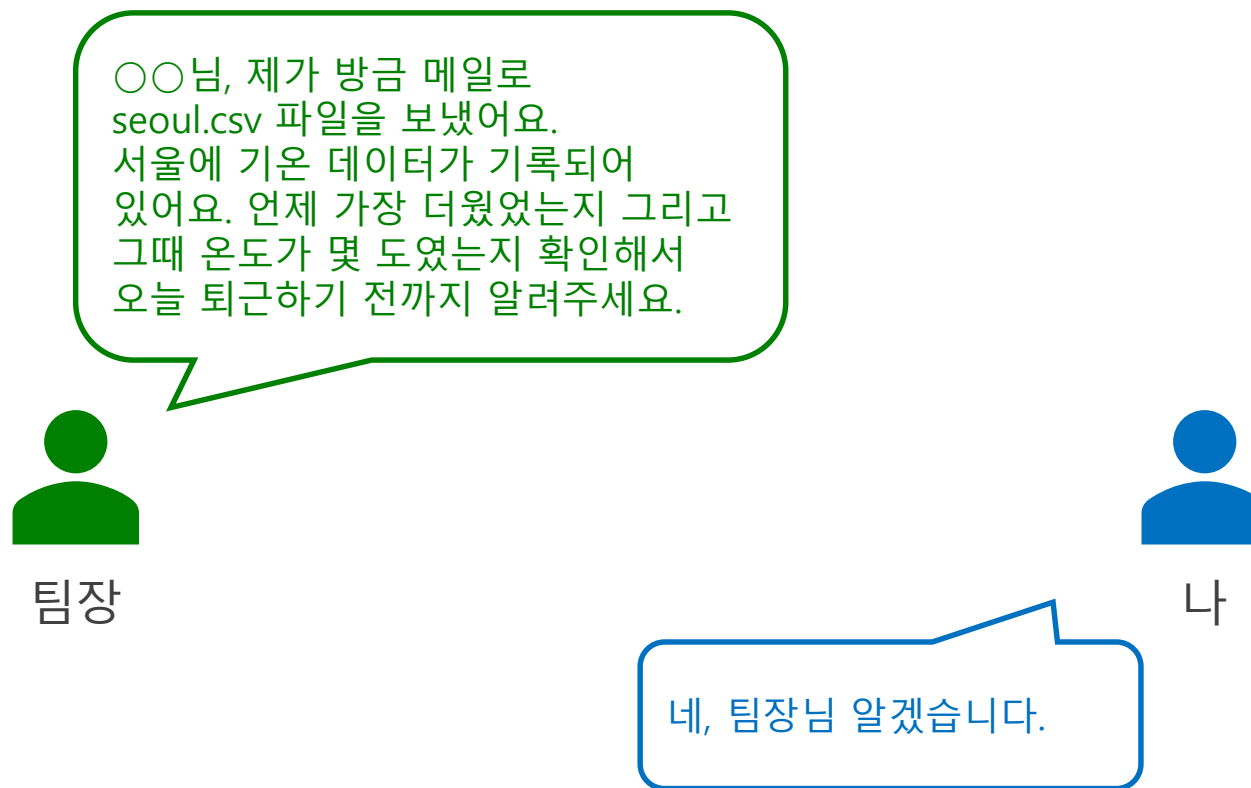


## 02. 서울의 기온 데이터 분석하기

01. 기온 데이터 분석 시작하기

03. 서울이 가장 더웠던 날은 언제 였을까

### ❖ 상황 가정



### ❖ CSV 파일 다운로드

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

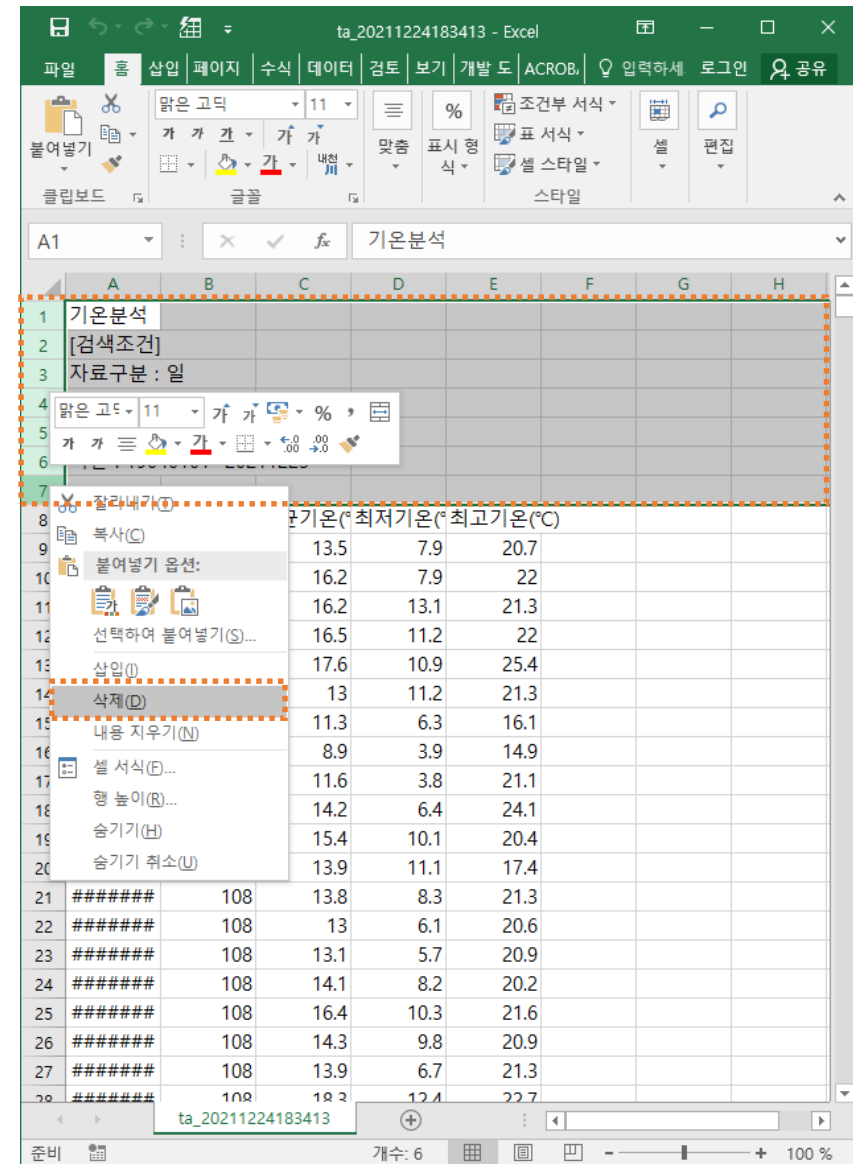


## 02. 서울의 기온 데이터 분석하기

### ❖ CSV 파일 편집

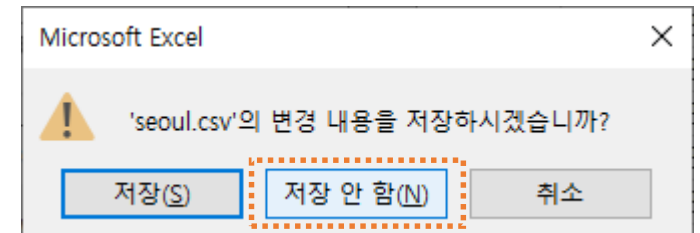
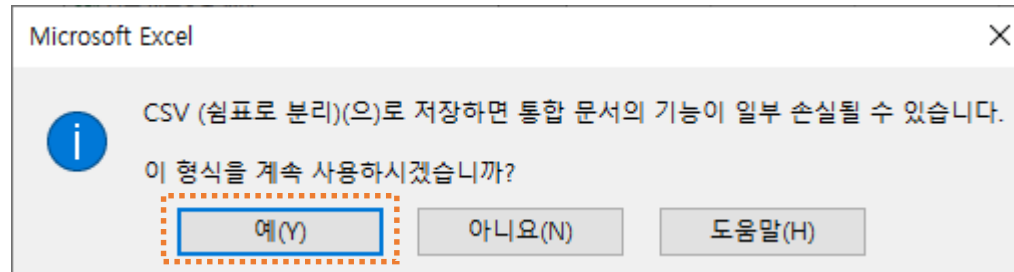
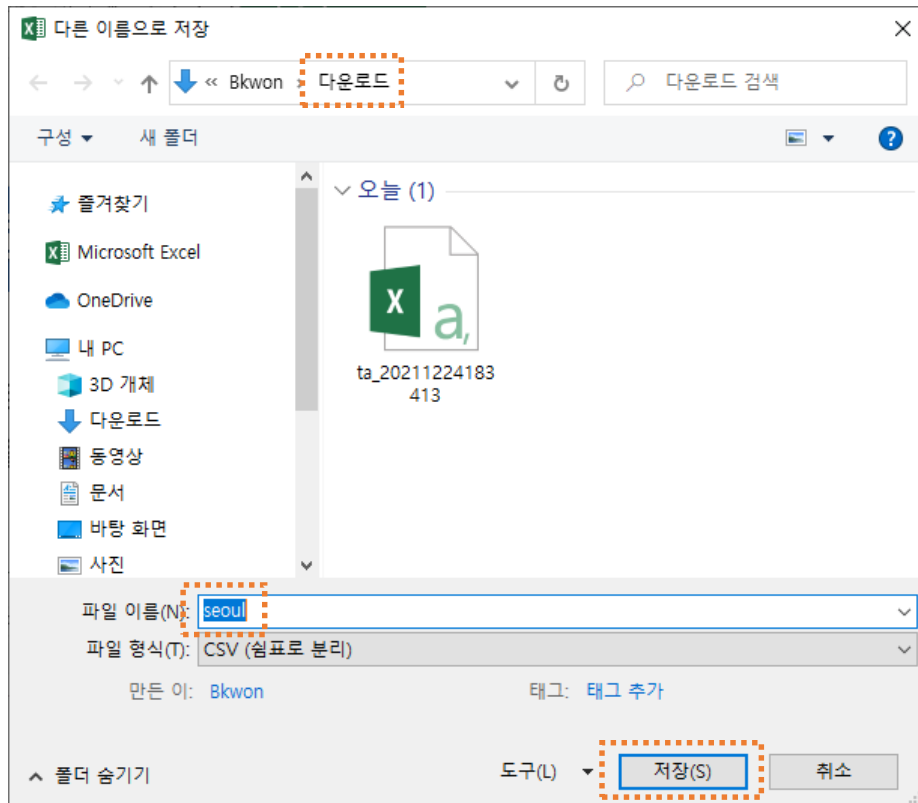
- 다운로드한 CSV 파일을 엑셀 프로그램으로 열고  
데이터 분석에 불필요한 1~7행을 삭제합니다.

삭제



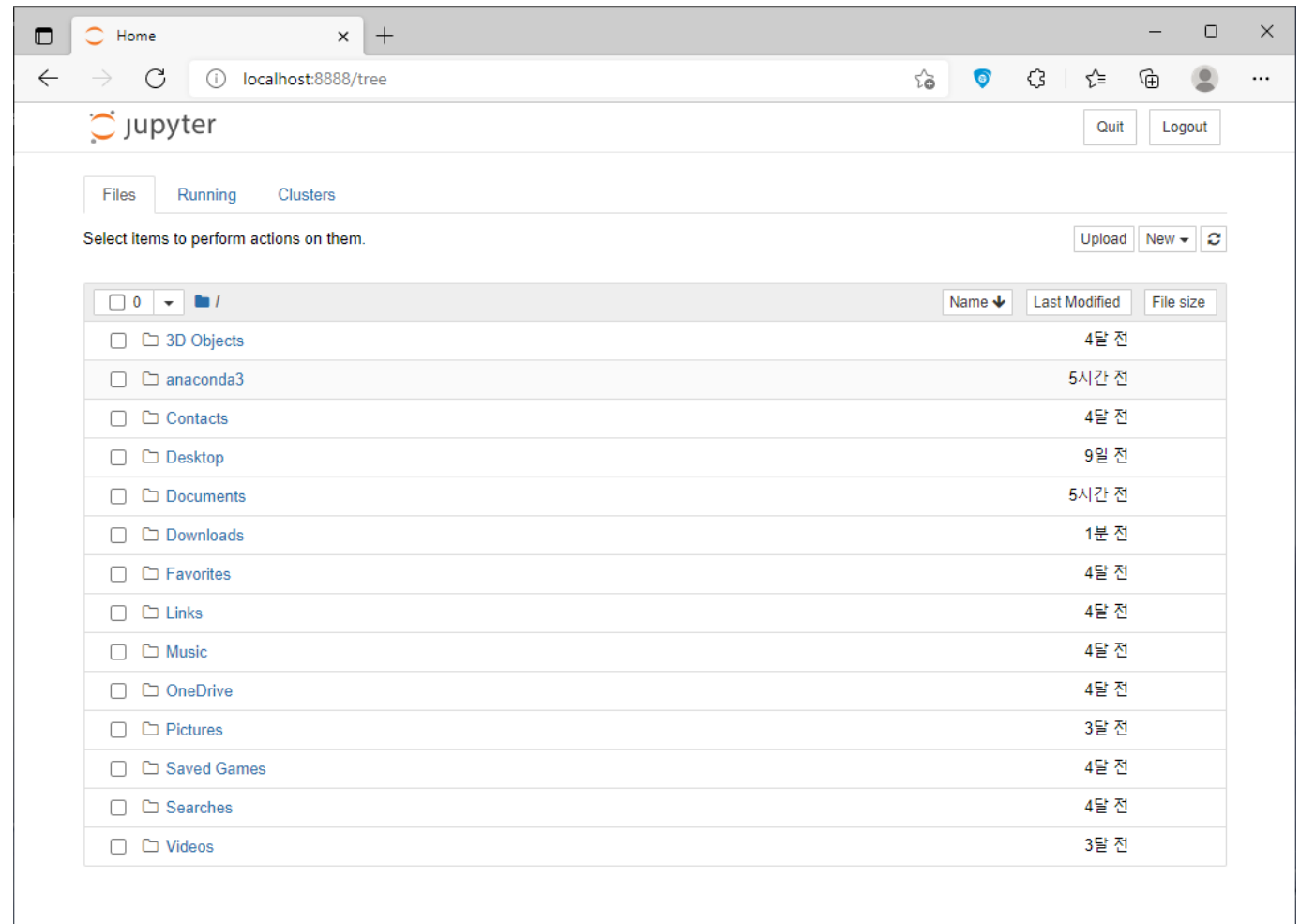
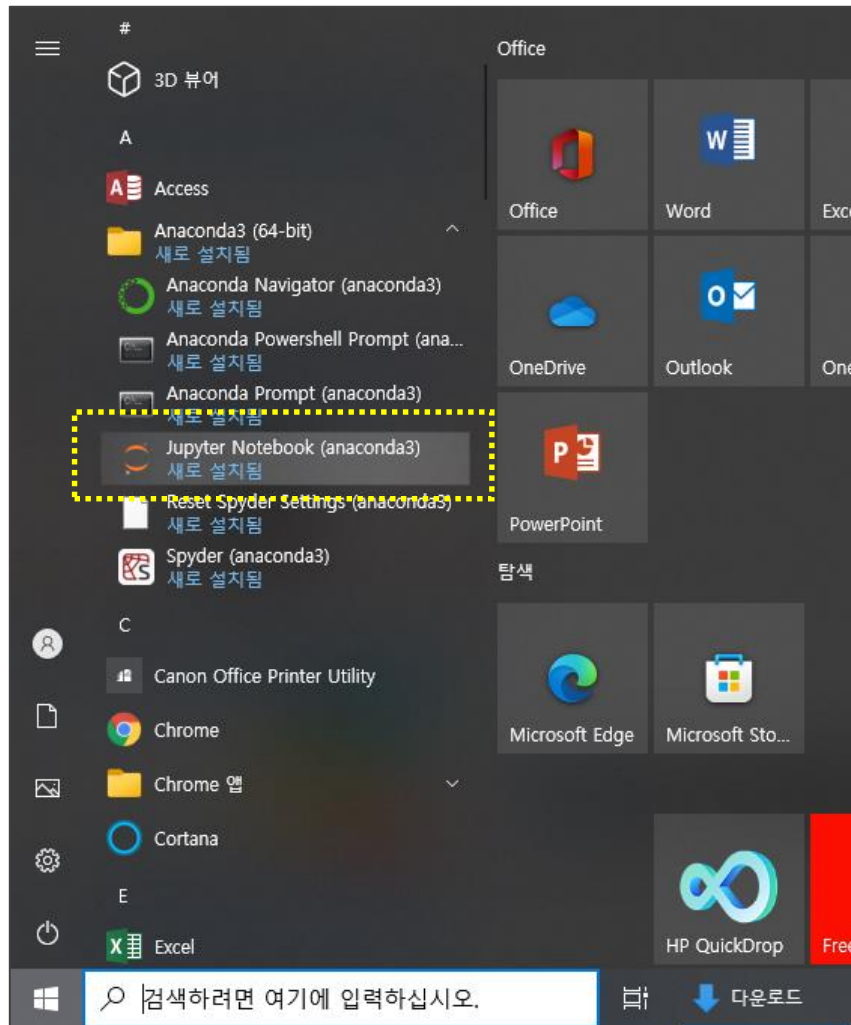
### ❖ CSV 파일 저장

- 메뉴에서 [파일] – [다른 이름으로 저장]을 눌러 파일 이름을 seoul로 변경하고 다운로드 폴더에 저장합니다. 이때 경고창이 뜨면 "예(Y)" 버튼을 클릭합니다.
- 완료되면 엑셀을 닫습니다. 이때 저장을 묻는 창이 뜨면 "저장 안 함(N)" 버튼을 클릭합니다.



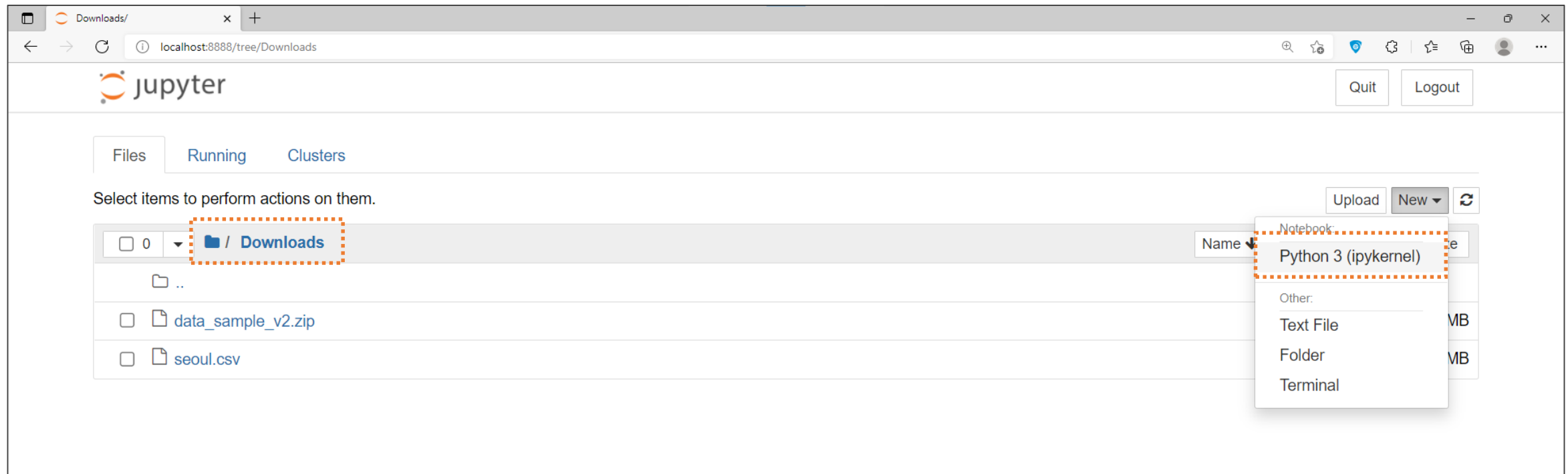
### ❖ ① CSV 파일에서 데이터 읽어오기 (1/5)

- 주피터 노트북을 실행합니다.



### ❖ ① CSV 파일에서 데이터 읽어오기 (2/5)

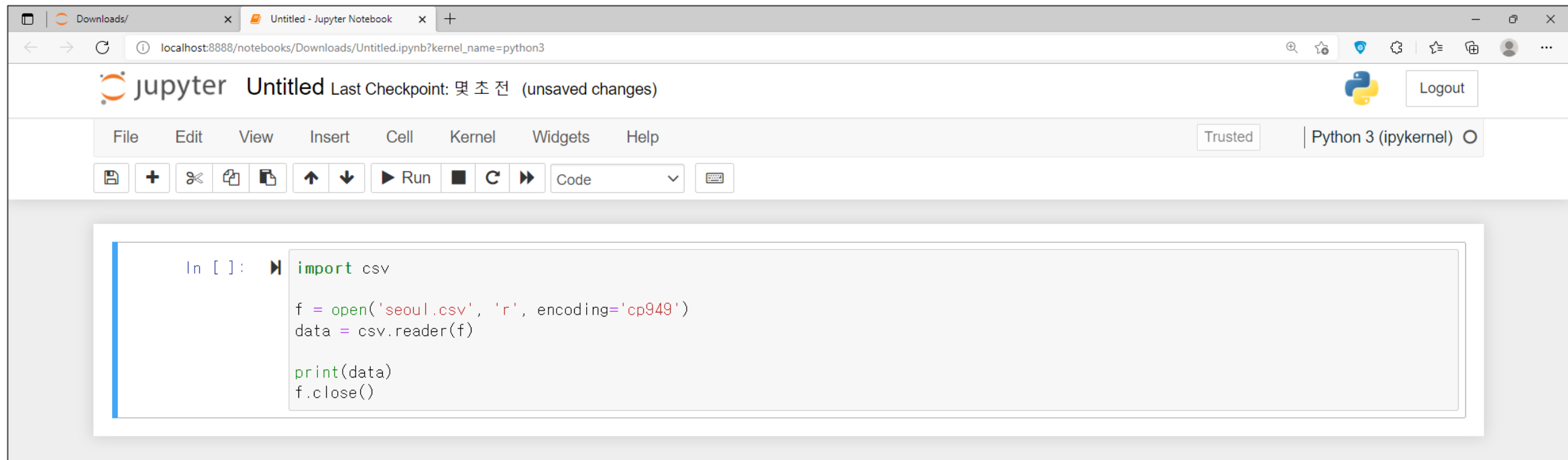
- 브라우저에서 Downloads 폴더를 선택한 후 오른쪽에 있는 [New] – [Python 3] 버튼을 클릭하여 새 파이썬 노트북을 생성합니다.





### ❖ ① CSV 파일에서 데이터 읽어오기 (3/5)

- 노트북의 빈 셀 (cell) 에 다음과 같이 코드를 작성합니다.



The screenshot shows a Jupyter Notebook interface in a web browser. The address bar shows the URL: localhost:8888/notebooks/Downloads/Untitled.ipynb?kernel\_name=python3. The notebook title is "Untitled" and it indicates "Last Checkpoint: 몇 초 전 (unsaved changes)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar contains icons for saving, adding cells, deleting, copying, pasting, and running code. The code cell contains the following Python code:

```
In [ ]: import csv

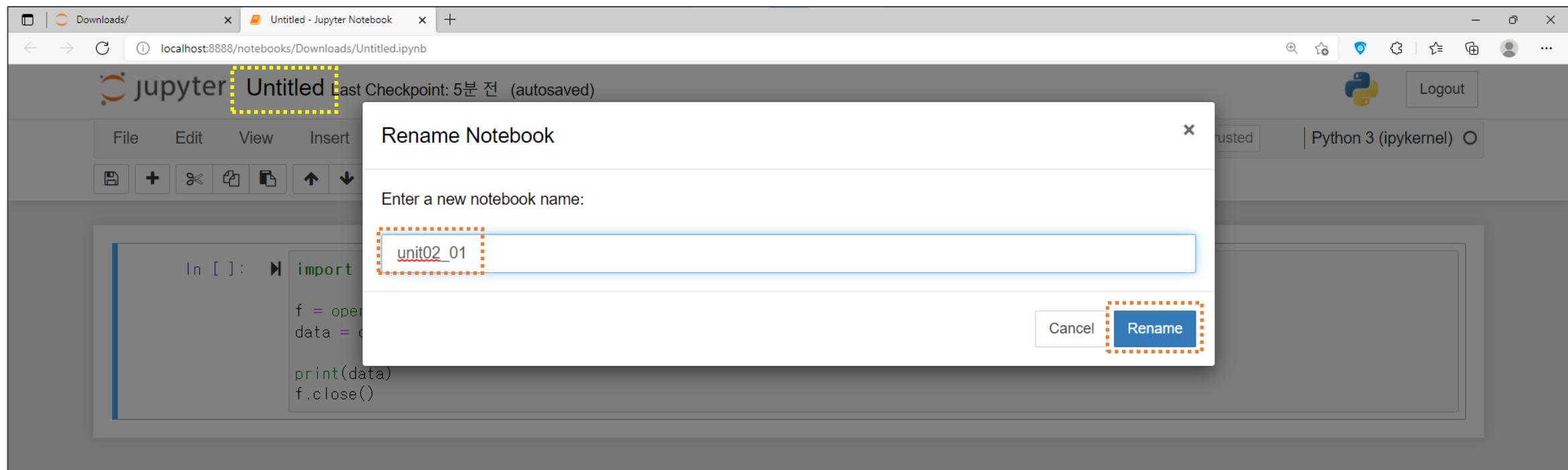
f = open('seoul.csv', 'r', encoding='cp949')
data = csv.reader(f)

print(data)
f.close()
```

만약 윈도우가 아닌 다른 운영체제 (macOS, 리눅스 등)를 사용하고 있다면,  
encoding='cp949'를 반드시 입력해야 합니다.

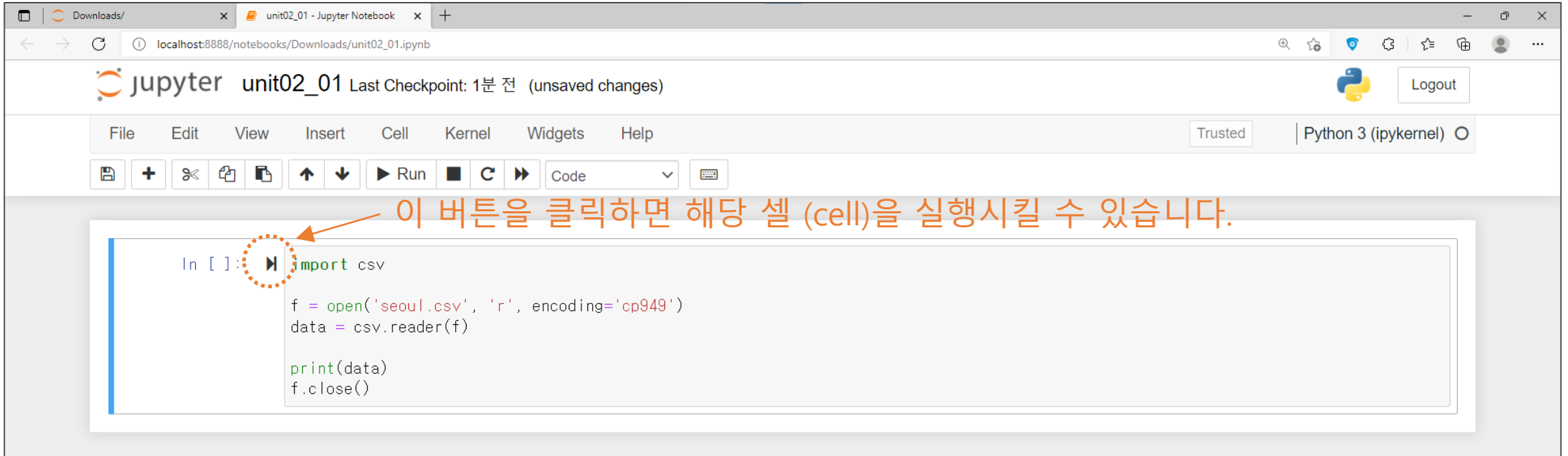
### ❖ ① CSV 파일에서 데이터 읽어오기 (4/5)

- "Untitled"라고 적혀 있는 노트북의 이름을 클릭하여 "unit02\_01"로 변경한 후 Rename버튼을 클릭하여 저장합니다.



### ❖ ① CSV 파일에서 데이터 읽어오기 (5/5)

- 작성한 코드를 실행하겠습니다.



The screenshot shows a Jupyter Notebook window titled 'unit02\_01'. The toolbar includes buttons for File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A 'Run' button (play icon) is highlighted with an orange circle and an arrow pointing to it. The code cell contains the following Python code:

```
In [ ]: import csv  
  
f = open('seoul.csv', 'r', encoding='cp949')  
data = csv.reader(f)  
  
print(data)  
f.close()
```

이 버튼을 클릭하면 해당 셀 (cell)을 실행시킬 수 있습니다.

※ 단축키를 통해서도 실행할 수 있습니다.

[Ctrl] + [Enter]: Run Cells

[Shift] + [Enter]: Run Cells and Select Cell

[Alt] + [Enter]: Run Cells and Insert Cell

### ❖ ② 데이터 출력하기 (1/6)

- 셀 (cell)을 하나 추가하여 아래의 코드 내용을 작성하고 실행합니다.

```
In [ ]: ▶ import csv

f = open('seoul.csv', 'r', encoding='cp949')
data = csv.reader(f)

for row in data:
    print(row)

f.close()
```

```
['날짜', '지점', '평균기온(℃)', '최저기온(℃)', '최고기온(℃)']
['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']
['1907-10-04', '108', '16.5', '11.2', '22']
['1907-10-05', '108', '17.6', '10.9', '25.4']
['1907-10-06', '108', '13', '11.2', '21.3']
['1907-10-07', '108', '11.3', '6.3', '16.1']
['1907-10-08', '108', '8.9', '3.9', '14.9']
['1907-10-09', '108', '11.6', '3.8', '21.1']
['1907-10-10', '108', '14.2', '6.4', '24.1']
['1907-10-11', '108', '15.4', '10.1', '20.4']
['1907-10-12', '108', '13.9', '11.1', '17.4']
['1907-10-13', '108', '13.8', '8.3', '21.3']
```

### ❖ ② 데이터 출력하기 (2/6)

- 출력된 결과화면을 살펴봅니다.

```
['1952-01-02', '108', '', '', '']  
['1952-01-03', '108', '', '', '']  
['1952-01-04', '108', '', '', '']  
['1952-01-05', '108', '', '', '']  
['1952-01-06', '108', '', '', '']  
['1952-01-07', '108', '', '', '']  
['1952-01-08', '108', '', '', '']  
['1952-01-09', '108', '', '', '']  
['1952-01-10', '108', '', '', '']  
['1952-01-11', '108', '', '', '']  
['1952-01-12', '108', '', '', '']  
['1952-01-13', '108', '', '', '']  
['1952-01-14', '108', '', '', '']  
['1952-01-15', '108', '', '', '']  
['1952-01-16', '108', '', '', '']  
['1952-01-17', '108', '', '', '']  
['1952-01-18', '108', '', '', '']  
['1952-01-19', '108', '', '', '']  
['1952-01-20', '108', '', '', '']  
['1952-01-21', '108', '', '', '']
```

누락된 데이터가 있습니다.

6.25전쟁 (1950년 6월 25일 – 1953년 7월 27일)

### ❖ ② 데이터 출력하기 (3/6)

- 출력된 결과화면을 살펴봅니다.

```
['2017-10-01', '108', '18.2', '15.5', '21.2']  
['2017-10-02', '108', '22', '15.6', '29.4']  
['2017-10-03', '108', '17.6', '13.4', '23.6']  
['2017-10-04', '108', '16.7', '10.7', '24.3']  
['2017-10-05', '108', '18.7', '13.9', '23.4']  
['2017-10-06', '108', '18.9', '16.2', '23.3']  
['2017-10-07', '108', '21.9', '16.9', '28.8']  
['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']  
['2017-10-17', '108', '16.2', '9.2', '23.9']  
['2017-10-18', '108', '16.5', '14.2', '19.1']  
['2017-10-19', '108', '17', '11.9', '23.2']  
['2017-10-20', '108', '17', '11.1', '24.2']
```

2017년 10월12일의 최고기온 데이터도 누락되어 있습니다.

### ❖ with open as f

- f.close() 필요 없음

```
In [9]: import csv

with open('seoul.csv', 'r', encoding='cp949') as f:
    data = csv.reader(f)
    for row in data:
        print(row)
```

```
['날짜', '지점', '평균기온(℃)', '최저기온(℃)', '최고기온(℃)']
['₩t1907-10-01', '108', '13.5', '7.9', '20.7']
['₩t1907-10-02', '108', '16.2', '7.9', '22']
['₩t1907-10-03', '108', '16.2', '13.1', '21.3']
['₩t1907-10-04', '108', '16.5', '11.2', '22']
['₩t1907-10-05', '108', '17.6', '10.9', '25.4']
['₩t1907-10-06', '108', '13', '11.2', '21.3']
['₩t1907-10-07', '108', '11.3', '6.3', '16.1']
['₩t1907-10-08', '108', '8.9', '3.9', '14.9']
['₩t1907-10-09', '108', '11.6', '3.8', '21.1']
```

### ❖ 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 합니다.

- NA: Not Available
- NaN: Not a Number

- 결측치가 있는지 어떻게 확인할 수 있나요?

◆ `pandas.DataFrame.info()`, `pandas.DataFrame.isna()`와 같은 함수들을 이용할 수 있습니다.

◆ 빈 문자열 ("")을 확인하기 위해서는 사용자가 별도로 관련 기능을 구현해야 합니다.

- 만약 결측치가 있다는 것이 확인되면, 결측치를 어떻게 처리해야 할까요?

#### ◆ ① 결측치 대체

- 해당 결측치를 평균 값이나 바로 앞 (또는 뒤) 데이터 값으로 대체하는 등 여러 방법들이 존재합니다.

#### ◆ ② 결측치 제거

- 결측치가 존재하는 행 (Row) 또는 열 (Column)을 제거합니다.



### ❖ ② 데이터 출력하기 (5/6)

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

```
In [14]: ▶ import csv

f = open('seoul.csv', 'r', encoding='cp949')
data = csv.reader(f)

for row in data:
    if '' in row:
        print(row)
    #     break

f.close()
```

### ❖ ② 데이터 출력하기 (6/6)

- 실행하여 보면 아래와 같이 결측치를 포함하는 데이터들만 출력됩니다.

```
[ '1953-11-14', '108', '', '', '' ]  
[ '1953-11-15', '108', '', '', '' ]  
[ '1953-11-16', '108', '', '', '' ]  
[ '1953-11-17', '108', '', '', '' ]  
[ '1953-11-18', '108', '', '', '' ]  
[ '1953-11-19', '108', '', '', '' ]  
[ '1953-11-20', '108', '', '', '' ]  
[ '1953-11-21', '108', '', '', '' ]  
[ '1953-11-22', '108', '', '', '' ]  
[ '1953-11-23', '108', '', '', '' ]  
[ '1953-11-24', '108', '', '', '' ]  
[ '1953-11-25', '108', '', '', '' ]  
[ '1953-11-26', '108', '', '', '' ]  
[ '1953-11-27', '108', '', '', '' ]  
[ '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', '' ]
```

### ❖ ③ 헤더 저장하기 (1/2)

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

#### ◆ `next()` 함수

- 첫 번째 데이터 행을 읽어오면 데이터의 탐색 위치를 다음 행으로 이동시킵니다.

```
In [15]: ▶ import csv

f = open('seoul.csv', 'r', encoding='cp949')
data = csv.reader(f)

header = next(data)
print(header)

f.close()

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

### ❖ ③ 헤더 저장하기 (2/2)

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

```
In [16]: ▶ import csv

f = open('seoul.csv', 'r', encoding='cp949')
data = csv.reader(f)

header = next(data)
print(header)

for row in data:
    print(row)

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

○○님, 제가 방금 메일로  
seoul.csv 파일을 보냈어요.  
서울에 기온 데이터가 기록되어  
있어요. 언제 가장 더웠는지 그리고  
그때 온도가 몇 도였는지 확인해서  
오늘 퇴근하기 전까지 알려주세요.



팀장

가장 덥다?  
→ "최고 기온이 가장 높다"



나

네, 팀장님 알겠습니다.

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

## ❖ ② 문제 해결 방법 구상하기

- 문제를 해결하기 위한 절차 (= 알고리즘, Algorithm)는?
  - ◆ Step 1) 데이터를 읽어온다.
  - ◆ Step 2) 순차적으로 최고 기온을 확인한다.
  - ◆ Step 3) 최고 기온이 가장 높았던 날짜의 데이터를 저장한다.
  - ◆ Step 4) 최종 저장된 데이터를 출력한다.

## ❖ ③ 파이썬 코드로 구현하기 (1/6)

- Step 1) 데이터를 읽어온다.

- ◆ 주피터 노트북을 열고 [File] – [New Notebook] – [Python 3] 버튼을 클릭하여 새 파일을 만듭니다.

- ◆ 파일의 이름을 unit03\_01로 수정하고 다음과 같은 코드를 작성합니다.

In [3]: ▶ `import csv`

```
f = open('seoul.csv', 'r', encoding='cp949')  
data = csv.reader(f)  
header = next(data)
```

```
print(header)  
for row in data:  
    print(row)  
    break
```

```
f.close()
```

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

Windows 운영체제를 사용하고 있다면,  
encoding='cp949'는 생략할 수 있습니다.

`f = open('seoul.csv')` 라고 간략하게 작성할 수 있습니다.

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



## ❖ ③ 파이썬 코드로 구현하기 (2/6)

- Step 1) 데이터를 읽어온다.

- ◆ 최고 기온 값을 찾기 위해서는 기온이 높고 낮다는 대소 관계를 비교해야 합니다.

- ◆ 따라서 자료형을 문자열 (String)에서 실수 (Float)로 변환해 주어야 합니다. → float( ) 함수를 사용하면 됩니다.

```
In [5]: ▶ import csv

f = open('seoul.csv', 'r', encoding='cp949')
data = csv.reader(f)
header = next(data)

print(header)
for row in data:
    row[4] = float(row[4])    # row[4]와 row[-1]은 같습니다.
    print(row)
    break

f.close()
```

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

작은 따옴표 (")가 사라졌습니다.  
더 이상 자료형이 문자열이 아닙니다.

## ❖ ③ 파이썬 코드로 구현하기 (3/6)

- Step 1) 데이터를 읽어온다.

- ◆ 서울 기온 데이터에는 빈 문자열 (")로 표현되는 결측치 (Missing Value)가 있었습니다.

- ◆ break 명령어를 주석 처리하고 코드를 실행하면 아래와 같이 오류가 발생합니다.

```
['1950-08-24', '108', '21.8', '15.2', 29.6]  
['1950-08-25', '108', '22', '16', 28.7]  
['1950-08-26', '108', '21.6', '17.1', 28.6]  
['1950-08-27', '108', '20.4', '15.5', 26.4]  
['1950-08-28', '108', '21.2', '16.4', 28.4]  
['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]
```

```
-----  
ValueError                                Traceback (most recent call last)  
~\AppData\Local\Temp\ipykernel_8928\1401348546.py in <module>  
      7 print(header)  
      8 for row in data:  
----> 9     row[4] = float(row[4])      # row[4]와 row[-1]은 같습니다.  
     10     print(row)  
     11
```

ValueError: could not convert string to float: ''

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

## ❖ ③ 파이썬 코드로 구현하기 (4/6)

- Step 1) 데이터를 읽어온다.

◆ 결측치를 다른 값으로 대체하는 전략을 사용하겠습니다.

◆ 최고 기온을 찾는 문제이므로 최고 기온으로 나오기 어려운 값인 -999으로 결측치를 대체합니다.

```
In [6]: ▶ import csv

f = open('seoul.csv', 'r', encoding='cp949')
data = csv.reader(f)
header = next(data)

print(header)
for row in data:
    if row[4] == '':      # 만약 최고기온 데이터가 빈 문자열이라면
        row[4] = -999    # -999를 대입

    row[4] = float(row[4])    # row[4]와 row[-1]은 같습니다.
    print(row)

f.close()
```

## ❖ ③ 파이썬 코드로 구현하기 (5/6)

- Step 1) 데이터를 읽어온다.

◆ 코드를 다시 실행시켜보면, 오류 없이 수행됩니다.

```
[ '2021-12-04', '108', '1.4', '2.1', 11.0]  
['2021-12-05', '108', '2.9', '-2.3', 9.6]  
['2021-12-06', '108', '5.4', '-0.7', 12.1]  
['2021-12-07', '108', '6.8', '2.6', 13.3]  
['2021-12-08', '108', '6.7', '1.9', 13.3]  
['2021-12-09', '108', '6.5', '2.7', 9.7]  
['2021-12-10', '108', '7', '5.8', 8.2]  
['2021-12-11', '108', '6', '4.4', 9.8]  
['2021-12-12', '108', '1.2', '-3.8', 5.1]  
['2021-12-13', '108', '-2.2', '-5.9', 1.9]  
['2021-12-14', '108', '3.7', '-0.7', 8.2]  
['2021-12-15', '108', '7.2', '5.4', 10.0]  
['2021-12-16', '108', '6.4', '3.3', 9.8]  
['2021-12-17', '108', '-5.6', '-10.1', 3.2]  
['2021-12-18', '108', '-5.7', '-11.2', -1.8]  
['2021-12-19', '108', '-0.8', '-5.8', 4.2]  
['2021-12-20', '108', '5.2', '-0.9', 11.1]  
['2021-12-21', '108', '5.2', '0.8', 8.9]  
['2021-12-22', '108', '2.2', '-2.6', 8.2]  
['2021-12-23', '108', '2.7', '-1.5', 8.3]
```

## ❖ ③ 파이썬 코드로 구현하기 (6/6)

```
In [13]: ▶ import csv

max_temp = -999      # 최고 기온 값을 저장할 변수
max_date = ''        # 최고 기온이 가장 높았던 날짜를 저장할 변수

f = open('seoul.csv', 'r', encoding='cp949')
data = csv.reader(f)
header = next(data)

# print(header)
for row in data:
    if row[4] == '':    # 만약 최고기온 데이터가 빈 문자열이라면
        row[4] = -999  # -999를 대입

    row[4] = float(row[4])    # row[4]와 row[-1]은 같습니다.

    if max_temp < row[4]:
        max_temp = row[4]
        max_date = row[0]

f.close()

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

Step 2) 순차적으로 최고 기온을 확인한다.

Step 3) 최고 기온이 가장 높았던 날짜의 데이터를 저장한다.

Step 4) 최종 저장된 데이터를 출력한다.

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 언제였을까

# THANK YOU!

## Q & A

- Name: 강환수
- Office: 동양미래대학교 2호관 706호 (02-2610-1941)
- E-mail: [hsknag@dongyang.ac.kr](mailto:hsknag@dongyang.ac.kr)
- Homepage: <https://github.com/ai7dnn/2023-DA>