

기온 데이터 분석

공공데이터 분석

기온 공공 데이터 살펴보기

■ 기상자료 개방포털:

- 링크: <https://data.kma.go.kr>
- 강수량, 황사 발생 일수 등 기상과 관련된 다양한 데이터



기후통계분석 > 기온분석

■ 조건별 통계

- 원하는 날짜 입력 후 기온 정보 조회
 - 서울, 2010월 1월 1일 ~2022년 6월 30일
- CSV 파일 다운로드 가능함



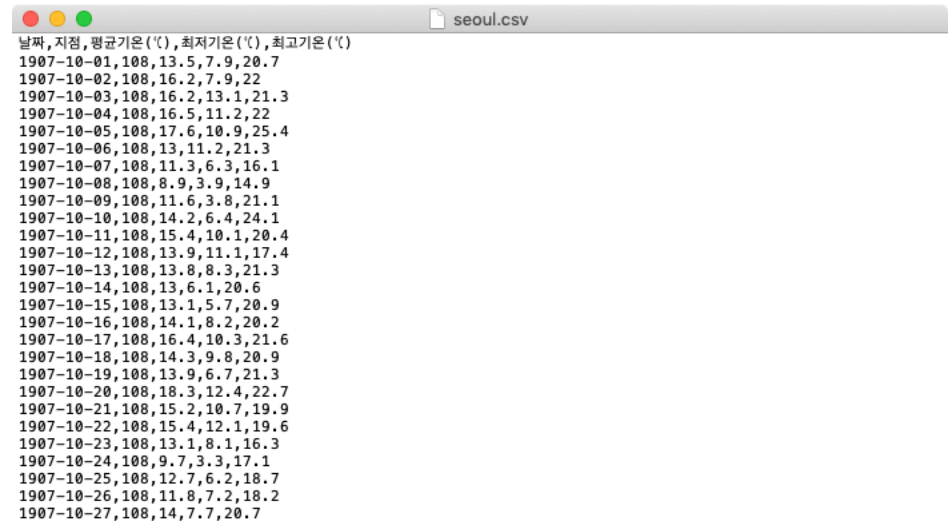
CSV 파일

■ CSV (Comma-Separated Values)

- 각각의 데이터 값을 콤마(,)로 구분하는 파일 형식
- 예전 스프레드시트나 데이터베이스 소프트웨어에서 많이 활용
- 정부에서 운영하는 공공데이터포털에서 제공하는 파일 형식
- 엑셀 및 메모장에서 열기 가능함



날짜	지점	평균기온(°C)	최저기온(°C)	최고기온(°C)
1907.10.1	108	13.5	7.9	20.7
1907.10.2	108	16.2	7.9	22
1907.10.3	108	16.2	13.1	21.3
1907.10.4	108	16.5	11.2	22
1907.10.5	108	17.6	10.9	25.4
1907.10.6	108	13	11.2	21.3
1907.10.7	108	11.3	6.3	16.1
1907.10.8	108	8.9	3.9	14.9
1907.10.9	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
1907.10.14	108	13	6.1	20.6
1907.10.15	108	13.1	5.7	20.9
1907.10.16	108	14.1	8.2	20.2
1907.10.17	108	16.4	10.3	21.6
1907.10.18	108	14.3	9.8	20.9
1907.10.19	108	13.9	6.7	21.3
1907.10.20	108	18.3	12.4	22.7
1907.10.21	108	15.2	10.7	19.9
1907.10.22	108	15.4	12.1	19.6
1907.10.23	108	13.1	8.1	16.3
1907.10.24	108	9.7	3.3	17.1
1907.10.25	108	12.7	6.2	18.7
1907.10.26	108	11.8	7.2	18.2
1907.10.27	108	14	7.7	20.7
1907.10.28	108	13.8	8.4	19.6
1907.10.29	108	14.3	10.7	20
1907.10.30	108	12.2	6.7	20.1
1907.10.31	108	13.4	5.2	20.3
1907.11.1	108	16.1	11.7	21.3
1907.11.2	108	14.2	9.1	21.1
1907.11.3	108	6.4	2.3	11.1



```
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
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
1907-10-14,108,13,6.1,20.6
1907-10-15,108,13.1,5.7,20.9
1907-10-16,108,14.1,8.2,20.2
1907-10-17,108,16.4,10.3,21.6
1907-10-18,108,14.3,9.8,20.9
1907-10-19,108,13.9,6.7,21.3
1907-10-20,108,18.3,12.4,22.7
1907-10-21,108,15.2,10.7,19.9
1907-10-22,108,15.4,12.1,19.6
1907-10-23,108,13.1,8.1,16.3
1907-10-24,108,9.7,3.3,17.1
1907-10-25,108,12.7,6.2,18.7
1907-10-26,108,11.8,7.2,18.2
1907-10-27,108,14,7.7,20.7
```

CSV 파일 다운로드

- 기상자료개방포털(<https://data.kma.go.kr>)

- 1) 기후통계분석 -> 통계분석 ->기온분석
- 2) 기간: 1904년 1월 1일부터 2022년 6월 30일까지
- 3) 지역: 대구(143)

지역/지점명으로 선택

- 전체
- 지역
 - 서울특별시
 - 부산광역시
 - 대구광역시
 - 대구 (143)
 - 대구(기) (176)
 - 인천광역시
 - 광주광역시
 - 대전광역시
 - 울산광역시
 - 경기도
 - 강원도
 - 충청북도
 - 충청남도
 - 전라북도
 - 전라남도
 - 경상북도
 - 경상남도
 - 제주도
 - 세종특별자치시

선택완료

기온분석

평년값

통계분석

기상현상일수

응용기상분석

데이터 개방
오픈 API

기온분석

그래프

분포도

자료 설정

지점별로 기온의 시계열 분석을 확인합니다.
일, 월, 연의 평균기온, 최저기온, 최고기온을 각각 조회할 수 있습니다.

* '지역/지점'의 '지역'은 전국 및 광역 단위의 평균 제공(1973년~) (전국 및 광역별 평균에 사용된 지점은 전국 평균산출에 사용되는 45개 지점이며, 제주도는 제주시와 서귀포시 자료임)

자료구분: 일

자료형태: 기본

기간: 19041001 ~ 20200608

지역/지점: 대구

검색

CSV 다운로드

Excel 다운로드

기온분석 기본 대구(143) 일자료 기간: 19041001 ~ 20200608

최저기온

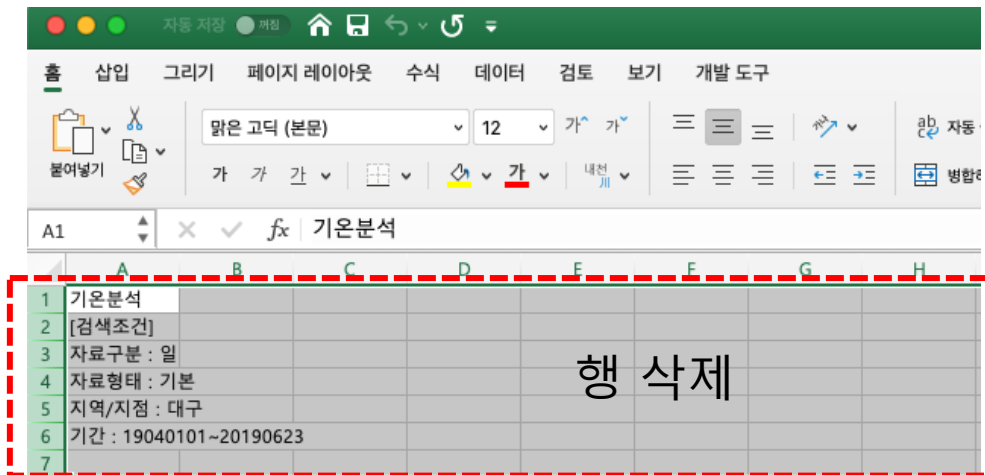
평균기온

최고기온

jupyter notebook 시작
폴더에 저장

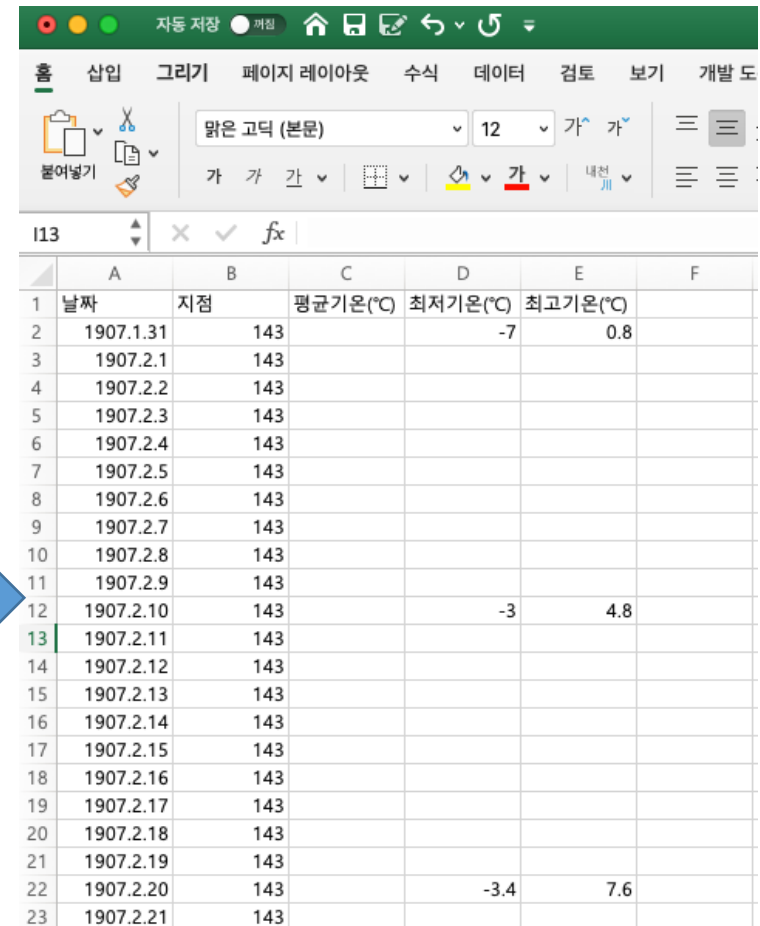
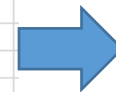
엑셀에서 불필요한 행 삭제

- 다운로드한 CSV 파일을 엑셀 프로그램으로 열고,
- 데이터 분석에 불필요한 1~7행 삭제 후 csv 파일 형태로 저장



행 삭제

	A	B	C	D	E	F	G	H
1	기온분석							
2	[검색조건]							
3	자료구분 : 일							
4	자료형태 : 기본							
5	지역/지점 : 대구							
6	기간 : 19040101~20190623							
7								
8	날짜	지점	평균기온(°C)	최저기온(°C)	최고기온(°C)			
9	1907.1.31	143		-7	0.8			
10	1907.2.1	143						
11	1907.2.2	143						
12	1907.2.3	143						
13	1907.2.4	143						
14	1907.2.5	143						
15	1907.2.6	143						
16	1907.2.7	143						
17	1907.2.8	143						
18	1907.2.9	143						
19	1907.2.10	143		-3	4.8			
20	1907.2.11	143						
21	1907.2.12	143						
22	1907.2.13	143						
23	1907.2.14	143						



	A	B	C	D	E	F
1	날짜	지점	평균기온(°C)	최저기온(°C)	최고기온(°C)	
2	1907.1.31	143		-7	0.8	
3	1907.2.1	143				
4	1907.2.2	143				
5	1907.2.3	143				
6	1907.2.4	143				
7	1907.2.5	143				
8	1907.2.6	143				
9	1907.2.7	143				
10	1907.2.8	143				
11	1907.2.9	143				
12	1907.2.10	143		-3	4.8	
13	1907.2.11	143				
14	1907.2.12	143				
15	1907.2.13	143				
16	1907.2.14	143				
17	1907.2.15	143				
18	1907.2.16	143				
19	1907.2.17	143				
20	1907.2.18	143				
21	1907.2.19	143				
22	1907.2.20	143		-3.4	7.6	
23	1907.2.21	143				

CSV 파일 저장 및 확인

- daegu.csv 파일은 jupyter notebook이 실행되는 폴더에 복사함
- jupyter notebook 실행 경로 확인: `%pwd` 명령어
 - pwd: print working directory

```
In 3 1 %pwd
```

```
Out 3      '/Users/changsu/PycharmProjects/BigDataClass/PublicData/day1'
```

- daegu.csv 파일 복사 후 파일 목록 확인: `%ls` 명령어

```
In 4 1 %ls
```

```
daegu.csv
```

```
day1_temperature.ipynb
```

CSV 파일에서 데이터 읽어 오기

■ CSV 파일 함수

`csv.reader(csvfile, delimiter)`: CSV 파일에서 데이터를 읽어오는 함수
`csv.writer(csvfile, delimiter)`: CSV 파일에 데이터를 저장하는 함수

■ 순서

- CSV 모듈을 불러옴 (`import csv`)
- 파일 `open()`
- csv reader 객체 생성 및 파일 읽어 오기
- data 출력
- 파일 `close()`

```
import csv

f = open('daegu.csv', 'r', encoding='euc_kr')
data = csv.reader(f, delimiter=',') # 구분자 설정
print(data) # 데이터 출력
f.close() # 파일 닫음
```

```
<_csv.reader object at 0x11146c7b8>
```


대구 기온 분석

- csv 파일에서 한 라인씩 데이터 출력

```
import csv
f = open('daegu.csv', 'r', encoding='euc_kr')
data = csv.reader(f, delimiter=',')
for row in data:
    print(row)
f.close()
```

```
['날짜', '지점', '평균기온(°C)', '최저기온(°C)', '최고기온(°C)'] ['1907.1.31',
'143', '', '-7', '0.8']
['1907.2.1', '143', '', '', '']
['1907.2.2', '143', '', '', '']
['1907.2.3', '143', '', '', '']
['1907.2.4', '143', '', '', '']
['1907.2.5', '143', '', '', '']
['1907.2.6', '143', '', '', '']
['1907.2.7', '143', '', '', '']
['1907.2.8', '143', '', '', '']
['1907.2.9', '143', '', '', '']
['1907.2.10', '143', '', '-3', '4.8']
```

리스트 형태로 출력됨
(문자열 데이터)

데이터 분석: 서울

- 서울의 기온 데이터 누락: 1950년 9월

```
In [5]: import csv
f = open('seoul.csv', 'r', encoding='cp949')
data = csv.reader(f, delimiter=',')
for row in data:
    print(row)
f.close()
```

```
['1950-08-21', '108', '26.4', '21.9', '32.9']
['1950-08-22', '108', '25.2', '21.4', '32.5']
['1950-08-23', '108', '23.3', '18.9', '28.9']
['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']
['1950-09-01', '108', '', '', '']
['1950-09-02', '108', '', '', '']
['1950-09-03', '108', '', '', '']
['1950-09-04', '108', '', '', '']
['1950-09-05', '108', '', '', '']
['1950-09-06', '108', '', '', '']
['1950-09-07', '108', '', '', '']
['1950-09-08', '108', '', '', '']
['1950-09-09', '108', '', '', '']
```

데이터 누락
(1950년 ~1953년)

데이터 헤더 저장하기

■ 데이터 헤더

- 데이터 파일에서 여러 가지 값들이 어떤 의미를 갖는지 표시한 행
- 데이터의 첫 번째 줄에 위치

```
['날짜', '지점', '평균기온(°C)', '최저기온(°C)', '최고기온(°C)']  
[0]      [1]          [2]              [3]              [4]
```

• 헤더 저장

– `next()` 함수 사용

– 첫 번째 데이터 행을 읽어오면서 데이터의 탐색 위치를 다음 행으로 이동시키는 명령어

```
import csv  
f = open('daegu.csv', encoding='euc_kr')  
data = csv.reader(f)  
header = next(data)  
print(header)  
f.close()
```

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

데이터 헤더 및 데이터 출력

■ 헤더 및 5개의 데이터만 출력하기

```
import csv
f = open('daegu.csv', encoding='euc_kr')
data = csv.reader(f)
header = next(data)
print(header)

i = 1
for row in data:
    print(row) # csv 데이터를 한 줄씩 출력함
    if(i>5):
        break # 5개의 데이터 출력이면 break
    i += 1
f.close()
```

```
['날짜', '지점', '평균기온(°C)', '최저기온(°C)', '최고기온(°C)']
['1907.1.31', '143', '', '-7', '0.8']
['1907.2.1', '143', '', '', '']
['1907.2.2', '143', '', '', '']
['1907.2.3', '143', '', '', '']
['1907.2.4', '143', '', '', '']
```

데이터 누락

대구가 가장 더웠던 날은?

- 대구 기온이 가장 높았던 날의 날짜와 기온 구하기
 - 1. csv 데이터를 읽어옴
 - 2. 순차적으로 최고 기온을 확인 (비교)
 - 3. 최고 기온이 가장 높았던 날짜의 데이터를 저장
 - 문자열 형태의 데이터를 실수 형태로 변환
 - 지금까지의 최고 기온 값보다 현재 행(row)의 최고 기온이 더 높으면
 - 최고 기온 날짜 업데이트
 - 최고 기온 값 업데이트
 - 4. 최종 저장된 데이터 출력

대구 최고 기온 날짜와 최고 온도 구하기

```
import csv
f = open('daegu.csv', encoding='euc_kr')
data = csv.reader(f)

header = next(data)
max_temp = -999 # 최고 기온을 저장할 변수 초기화
max_date = '' # 최고 기온의 날짜를 저장할 변수 초기화

for row in data:
    if row[-1] == '': # [-1]: 리스트에서 마지막 데이터가 없는 경우
        row[-1] = -999
    row[-1] = float(row[-1])

    if(max_temp < row[-1]):
        max_temp = row[-1]
        max_date = row[0] # 날짜: index[0]
f.close()

print('기상 관측 이래 대구의 최고 기온이 가장 높았던 날은 {}로 {}도  
였습니다.'.format(max_date, max_temp))
```

기상 관측 이래 대구의 최고 기온이 가장 높았던 날은 1942.8.1 로 40.0 도였습니다.

Numpy를 활용한 최고 기온 찾기

```
import csv
import numpy as np

f = open('daegu.csv', encoding='euc_kr')
data = csv.reader(f)
header = next(data)
max_temp = -999
max_date = ''
max_temp_list = list() # 최고 기온을 저장할 리스트
max_temp_date = list() # 날짜 정보를 저장할 리스트
for row in data:
    if row[-1] == '': # [-1]: 리스트에서 마지막 데이터가 없는 경우
        row[-1] = '-999' # 가장 작은 값을 입력함

    max_temp_list.append(row[-1])
    max_temp_date.append(row[0])
f.close()
```

- np.max(): array에서 최대값
- np.min(): array에서 최소값 리턴
- np.argmax(): 최고값의 index 리턴
- np.argmin(): 최소값의 index 리턴

```
max_temp_array = np.array(max_temp_list) # 리스트를 ndarray 타입으로 변경
max_temp_array = max_temp_array.astype(float) # 문자열 타입을 float로 변경
max_date_array = np.array(max_temp_date) # 리스트를 ndarray 타입으로 변경
```

```
max_temp = max_temp_array.max()
index = max_temp_array.argmax() # 최대값의 index 리턴
max_date = max_date_array[index]
print("기상 관측 이래 대구의 최고 기온이 가장 높았던 날은 ", max_date,
      '로 ', max_temp, ' 도였습니다.')
```

데이터를 리스트에 저장하기

■ 리스트에 저장하기

- 기후 데이터에서 최고 기온 데이터를 리스트에 저장
- 리스트에 저장된 데이터 개수 확인

'날짜'	'지점'	'평균기온(°C)'	'최저기온(°C)'	'최고기온(°C)'
[0]	[1]	[2]	[3]	[4]

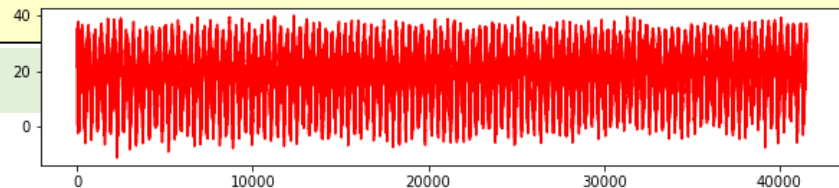
```
import csv
f = open('daegu.csv', encoding='euc_kr')
data = csv.reader(f)

header = next(data)
result = []

for row in data:
    if row[4] != '': # 최고 기온 데이터 값이 존재하면 리스트에 저장
        result.append(float(row[4]))

print(len(result))
f.close()
plt.figure(figsize=(10, 2)) # 그래프 크기 조절(가로 10인치, 세로 2인치)
plt.plot(result, 'r') # result 리스트에 저장된 값을 빨간색 그래프로 그리기
plt.show() # 그래프 그리기
```

41525



히스토그램: hist() 함수

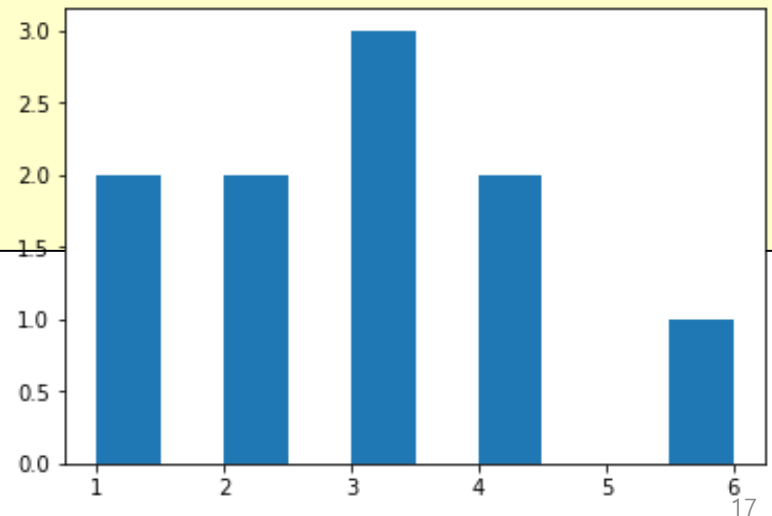
- 주사위 시뮬레이션
 - 주사위를 n회 굴려서 나온 결과를 기록함
 - 주사위의 눈이 나온 횟수를 히스토그램으로 그림
- bins: 가로축의 구간 개수 설정 (구간의 경계값 리스트)

```
import random
import matplotlib.pyplot as plt

dice = []
for i in range(10):
    dice.append(random.randint(1, 6))
print(dice)

#plt.hist(dice, bins=6)
plt.hist(dice)
plt.show()
```

```
[6, 2, 4, 6, 3, 1, 4, 3, 4, 5]
```



기온 데이터를 히스토그램으로 표현

- 한글 폰트 사용시 레이블의 '-' 기호 깨지는 현상 해결

- plt.rc('axes', unicode_minus=False)
- plt.rcParams['axes.unicode_minus'] = False

https://matplotlib.org/stable/gallery/text_labels_and_annotations/unicode_minus.html

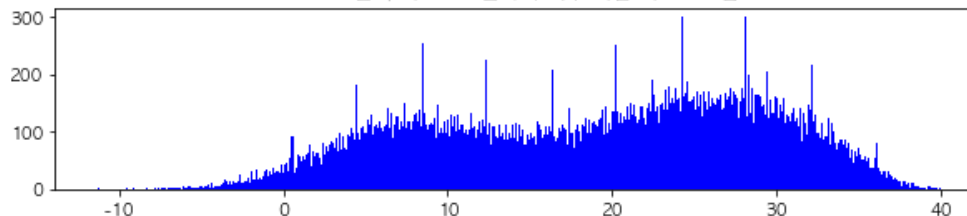
```
import csv
import matplotlib.pyplot as plt
f = open('daegu.csv', encoding='euc_kr')
data = csv.reader(f)
next(data)
result = []

for row in data :
    if row[-1] != '' : # 최고 기온을 리스트에 저장
        result.append(float(row[-1]))
f.close()
plt.figure(figsize=(10, 2))
plt.hist(result, bins=500, color='blue') # result에 저장된 값을 히스토그램으로 그림
plt.rc('font', family='AppleGothic') # "Malgun Gothic" For windows

plt.rcParams['axes.unicode_minus'] = False # 레이블에 '-' 부호가 깨지는 현상 방지
plt.title("1907년 부터 2022년까지 대구 기온 히스토그램")
plt.show()
```

1907년 부터 2022년까지 대구 기온 히스토그램

bins=500



기온 히스토그램 (8월)

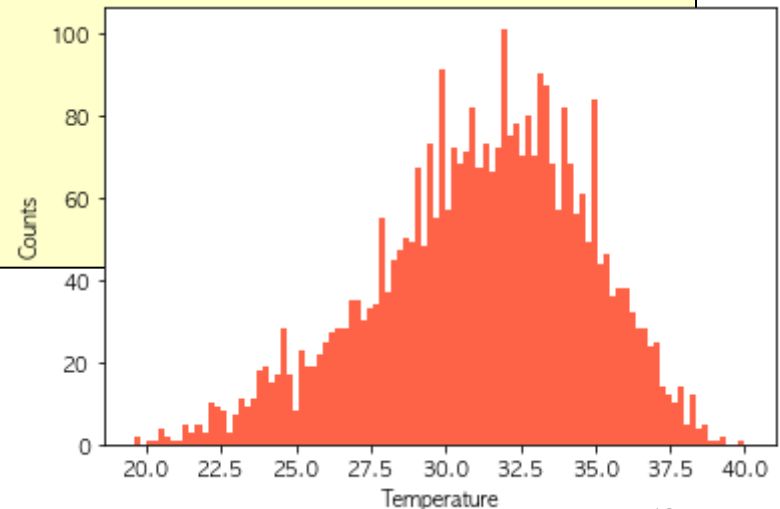
```
import csv
import matplotlib.pyplot as plt

f = open('daegu.csv', encoding='euc_kr')
data = csv.reader(f)
next(data)
aug = []

for row in data :
    month = row[0].split('.')[1]
    if row[-1] != '' :
        if month == '8':
            aug.append(float(row[-1]))

f.close()
plt.hist(aug, bins = 100, color = 'tomato')
plt.xlabel("Temperature") # x축 레이블
plt.ylabel("Counts")      # y축 레이블
plt.show()
```

날짜 정보: 1907.8.1
에서 ('.') 을 기준으로 분리함
[0]: 1907
[1]: 8
[2]: 1



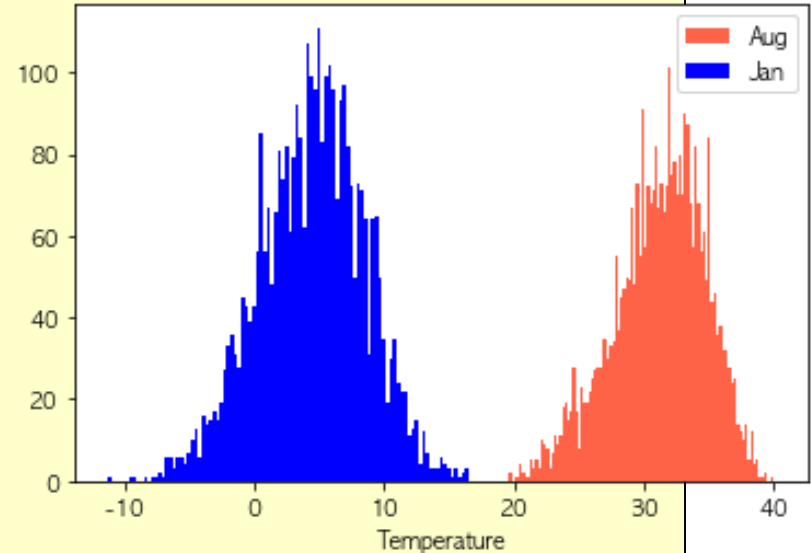
1월과 8월의 기온 데이터 히스토그램

```
import csv
import matplotlib.pyplot as plt

f = open('daegu.csv', encoding='euc_kr')
data = csv.reader(f)
next(data)
aug = []
jan = []

for row in data :
    month = row[0].split('.')[1]
    if row[-1] != '' :
        if month == '8':
            aug.append(float(row[-1]))
        if month == '1':
            jan.append(float(row[-1]))

f.close()
plt.hist(aug, bins = 100, color = 'tomato', label='Aug')
plt.hist(jan, bins = 100, color = 'b', label='Jan')
plt.xlabel("Temperature")
plt.rc('axes', unicode_minus=False)
plt.legend()
plt.show()
```



특정 날짜의 최고 기온 찾기

```
import csv
import matplotlib.pyplot as plt

def draw_graph_on_date(month, day):
    f = open('daegu.csv', encoding='euc_kr')
    data = csv.reader(f)
    next(data)
    result = []
    for row in data:
        if row[-1] != '':
            date_string = row[0].split('.')
            if date_string[1] == month and date_string[2] == day:
                result.append(float(row[-1]))

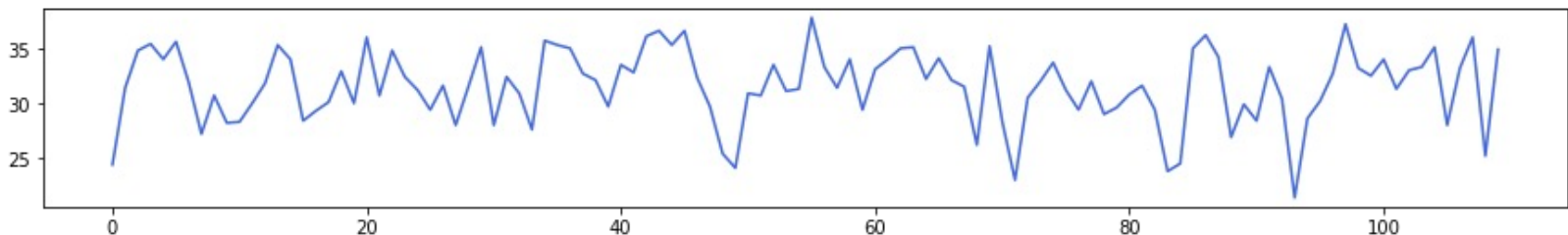
    f.close()
    plt.figure(figsize=(15,2))
    plt.plot(result, 'royalblue')
    plt.show()

month, date = input('날짜(월 일)를 입력하세요: ').split()

draw_graph_on_date(month, date)
```

날짜 정보를 분리하여
해당 날짜에 해당되는
데이터만 리스트에 저장

입력된 문자열을
공백으로 분리해서 각
변수에 입력



운영체제 플랫폼 구별 및 한글 폰트 설정

▪ platform 모듈

- system() 함수 사용

- Windows: 'Windows', Mac: 'Darwin', Linux: 'Linux'

```
import platform
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm

system_name = platform.system()
if system_name == 'Windows':
    # Windows 운영체제
    print('Windows OS')
    plt.rc('font', family='Malgun Gothic')
elif system_name == 'Darwin': # Mac OS
    print('Mac OS')
    plt.rc('font', family='AppleGothic')
elif system_name == 'Linux': # Linux
    print('Linux OS')
    path = '/usr/share/fonts/truetype/nanum/NanumMyeongjo.ttf'
    font_name = fm.FontProperties(fname=path, size=12)
    plt.rc('font', family=font_name)
else:
    print("Not support")
```

운영체제 별
설치된 한글폰트
이름 적용

2000년 이후 특정일의 최저, 최고 기온 찾기 #1

```
import csv
import matplotlib.pyplot as plt
import platform

def draw_lowhigh_graph(year, month, day):
    f = open('daegu.csv', encoding='euc_kr')
    data = csv.reader(f)
    next(data)
    high_temp = [] # 최고 기온을 저장할 리스트
    low_temp = [] # 최저 기온을 저장할 리스트
    x_year = [] # x축 연도를 저장할 리스트
    for row in data:
        if row[-1] != '':
            date_string = row[0].split('.')
            if int(date_string[0]) >= year: # 문자열 값을 int 형으로 변환해서 비교
                if int(date_string[1]) == month and int(date_string[2]) == day:
                    high_temp.append(float(row[-1]))
                    low_temp.append(float(row[-2]))
                    x_year.append(date_string[0]) # 연도 저장

    f.close()
```

입력된 year 이후부터 특정
날짜(month, day)의 기온
분석

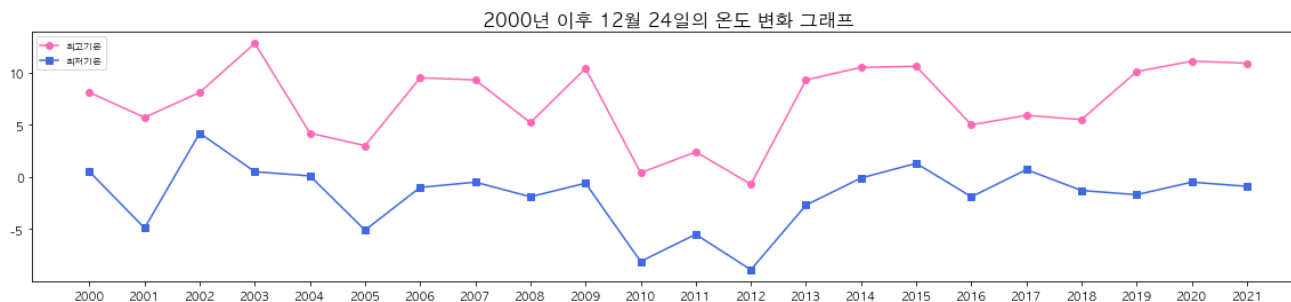
2000년 이후 특정일의 최저, 최고 기온 찾기 #2

```
plt.figure(figsize=(20, 4))
plt.plot(x_year, high_temp, 'hotpink', marker='o', label='최고기온')
plt.plot(x_year, low_temp, 'royalblue', marker='s', label='최저기온')

if platform.system() == 'Windows':
    font_name = fm.FontProperties(fname='c:\\Windows\\Fonts\\malgun.ttf').get_name()
    plt.rc('font', family=font_name)
    # plt.rc('font', family='Malgun Gothic', size=8) # 간단히 맑은 고딕으로 설정
else:
    # MacOS
    plt.rc('font', family='AppleGothic', size=8) # 한글 폰트 사용 For Mac OS

plt.rcParams['axes.unicode_minus'] = False
plt.title("{0}년 이후 {1}월 {2}일의 온도 변화 그래프".format(year, month, day),
          size=16)
plt.legend(loc=2)
plt.show() # 그래프 나타내기
```

draw_lowhigh_graph(2000, 12, 24)



Pandas를 활용한 기온 데이터 #1

❑ Pandas의 read_csv() 함수 호출

```
import pandas as pd

weather_df = pd.read_csv('daegu.csv', encoding='euc_kr')
print(weather_df.columns)
print(weather_df['날짜'].dtype) # '날짜' 컬럼은 object 타입
```

```
Index(['날짜', '지점', '평균기온(℃)', '최저기온(℃)', '최고기온(℃)'], dtype='object')
object
```

❑ DataFrame의 column 이름 변경: 특수 문자 제거

```
weather_df.columns = ['날짜', '지점', '평균기온', '최저기온', '최고기온']
print(weather_df.columns)
```

```
Index(['날짜', '지점', '평균기온', '최저기온', '최고기온'], dtype='object')
```

Pandas를 활용한 기온 데이터 #2

- ‘날짜’ 컬럼의 데이터 타입을 datetime 타입으로 변경
 - `to_datetime(df['컬럼명'], format='%Y-%m-%d')`

```
weather_df['날짜'] = pd.to_datetime(weather_df['날짜'], format='%Y-%m-%d')
print(weather_df['날짜'].dtype)
```

```
datetime64[ns]
```

- 누락값 개수 구하기

```
print(weather_df.head(5))
num_rows = weather_df.shape[0] # shape(row, col), shape[0]: row의 개수
num_missing = num_rows - weather_df.count() # coun(): 정상값의 개수
print(num_missing)
```

	날짜	지점	평균기온	최저기온	최고기온
0	1907-01-31	143	NaN	-7.0	0.8
1	1907-02-01	143	NaN	NaN	NaN
2	1907-02-02	143	NaN	NaN	NaN
3	1907-02-03	143	NaN	NaN	NaN
4	1907-02-04	143	NaN	NaN	NaN
날짜	0				
지점	0				
평균기온		703			
최저기온		629			
최고기온		630			

Pandas를 활용한 기온 데이터 #3

■ 누락값(NaN) 처리

- `dropna(axis)`: 누락값 제거
 - `axis=0`: NaN이 포함된 행 제거, `axis=1`: NaN이 포함된 열 제거
- `fillna(0)`: 누락값을 0으로 변경
- `fillna(method='ffill')`: 이전 값으로 변경(forward fill)
- `fillna(method='bfill')`: 이후 값으로 변경(backward fill)
- `interpolate()`: 누락값 양쪽의 값으로 중간값 계산

```
weather_df = weather_df.dropna(axis=0)
print(weather_df.count())
print(weather_df.head(5))
```

```
날짜      41450
지점      41450
평균기온   41450
최저기온   41450
최고기온   41450
dtype: int64
```

	날짜	지점	평균기온	최저기온	최고기온
701	1909-01-01	143	-4.1	-8.0	0.1
702	1909-01-02	143	-0.8	-6.7	6.1
703	1909-01-03	143	1.0	-2.4	3.6
704	1909-01-04	143	4.2	0.0	9.2
705	1909-01-05	143	3.4	-0.4	9.0

Pandas를 활용한 기온 데이터 #4

- 누락값을 제거한 최종 데이터를 csv파일로 저장
 - index = False: 인덱스 항목 저장 안함
 - encoding='utf-8' (euc_kr이 아닌 utf-8로 저장)

```
weather_df.to_csv('daegu_utf8.csv', index=False, mode='w',  
                  encoding='utf-8')
```

날짜	지점	평균기온	최저기온	최고기온
1909-01-01	143	-4.1	-8.0	0.1
1909-01-02	143	-0.8	-6.7	6.1
1909-01-03	143	1.0	-2.4	3.6
1909-01-04	143	4.2	0.0	9.2
1909-01-05	143	3.4	-0.4	9.0
1909-01-06	143	2.3	-1.2	7.4
1909-01-07	143	0.8	-2.8	6.4
1909-01-08	143	-0.5	-4.4	4.9
1909-01-09	143	-1.0	-4.4	5.3
1909-01-10	143	1.1	-5.3	8.4
1909-01-11	143	0.7	-3.6	6.1
1909-01-12	143	2.1	-2.2	7.1
1909-01-13	143	0.7	-3.7	3.9
1909-01-14	143	-1.0	-4.7	4.9
1909-01-15	143	-0.5	-5.7	5.3
1909-01-16	143	0.8	-4.8	6.4
1909-01-17	143	2.4	-3.4	9.3
1909-01-18	143	4.5	0.7	6.6
1909-01-19	143	2.2	0.3	5.0

daegu_utf8.csv 파일 내용

Pandas를 활용한 기온 데이터 #5

- 특정 연도와 달의 최고,최저 기온 평균값 계산
 - 해당 연도와 달의 DataFrame 가져오기

```
year_df = weather_df[weather_df['날짜'].dt.year == 2021]
month_df = year_df[year_df['날짜'].dt.month == 7]
print(month_df.head())
```

- datetime 객체 접근
 - dt.year, dt.month, dt.day

	날짜	지점	평균기온	최저기온	최고기온
41790	2021-07-01	143	25.1	19.6	32.4
41791	2021-07-02	143	24.0	20.3	28.8
41792	2021-07-03	143	22.7	21.4	24.8
41793	2021-07-04	143	27.8	21.8	32.4
41794	2021-07-05	143	23.8	20.6	27.2

Pandas를 활용한 기온 데이터 #6

- 특정 연도와 달의 최저 기온 및 최고 기온의 평균 계산

```
max_temp_mean = round(month_df['최고기온'].mean(), 1)
min_temp_mean = round(month_df['최저기온'].mean(), 1)

print('2021년 7월 최저기온 평균:{},
      최고기온 평균 :{}'.format(min_temp_mean, max_temp_mean))
```

2021년 7월 최저기온 평균:22.9, 최고기온 평균 :32.1



Questions?