

# 오픈소스 **SW** 기여

## 중간보고서

32182775 위성준  
32183698 이현기

## - 상세설계

### 1. 전체적인 시스템 구조

ui 스케치를 통해 구성한 데모이다.

```

import gradio as gr
import random
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd

```

Month  
May

Day  
15

Weather element  
☒ Temperature
☒ Precipitation
☒ Wind
☒ Humidity
☒ Air pressure

Location  
☒ Washington D.C
☒ Seoul

Clear Submit

지역	10(금)	11(토)
뉴욕	맑음 2~9	소나기 2~8
LA	구름 조금 9~13	흐리고 비 11~14
베이징	맑음 2~22	맑음 4~23

## - Gradio components

Month, Day -> **Dropdown**

Weather elements (temperature, wind, humidity, air pressure)

-> **CheckboxGroup**

Location (Washington, Seoul) -> **Radio**

Precipitation -> **Checkbox**

Month  
Select Months

Day  
Select Day

Weather element  
Choose weather element

☐ temperature
☐ wind
☐ humidity
☐ air\_pressure

Location  
Choose location

☐ Washington
☐ Seoul

☐ precipitation?

클리어 제출하기

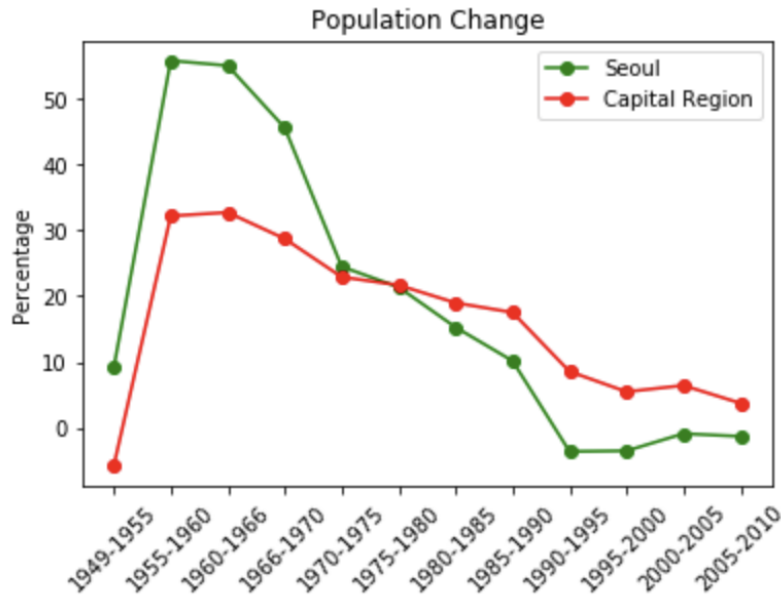
- **Matplotlib components**

선 그래프 (기온, 풍속, 기압) -> matplotlib.pyplot의 **plot method** 이용.

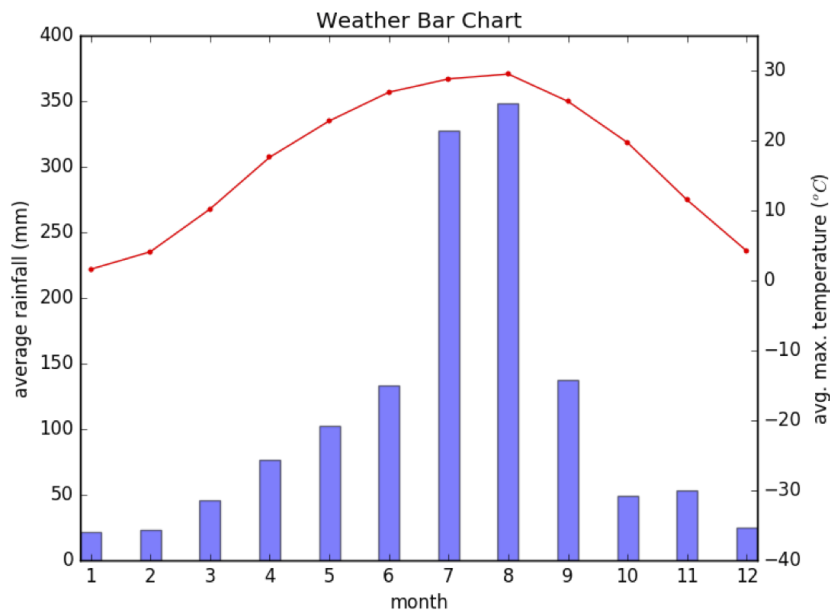
막대 그래프 (습도) -> matplotlib.pyplot의 **bar method** 이용.

Plot 구성 -> 기상 요소 개수에 따라 그래프 추가 (최대 plot 2개)

예시) 1번 plot X축 : 시간, Y축 : 기온, 풍속



예시) 2번 plot X축 : 시간, Y축 : 습도, 기압



※ 강수량은 일일 데이터의 양이 적기 때문에 그래프에 표시 X.

title = 2022년 기상 그래프

xlabel = 해당 date, ylabel = 해당 weather element

x축 rotation = 45

## - Pandas components

기상데이터 표 -> **DataFrame**

DataFrame 구성

index : hide, column : Location, date, time, Weather elements

데이터 전처리 후 Gradio를 이용하여 표 출력

In [33]: data

Out[33]:

	Area Abbreviation	Area Code	Area	Item Code	Item	Element Code	Element	Unit	latitude	longitude	...	Y2004	Y2005	Y2006	Y2007	Y2008	Y2009
0	AF	2	Afghanistan	2511	Wheat and products	5142	Food	1000 tonnes	33.94	67.71	...	3249.0	3486.0	3704.0	4164.0	4252.0	4538.0
1	AF	2	Afghanistan	2805	Rice (Milled Equivalent)	5142	Food	1000 tonnes	33.94	67.71	...	419.0	445.0	546.0	455.0	490.0	415.0
2	AF	2	Afghanistan	2513	Barley and products	5521	Feed	1000 tonnes	33.94	67.71	...	58.0	236.0	262.0	263.0	230.0	379.0
3	AF	2	Afghanistan	2513	Barley and products	5142	Food	1000 tonnes	33.94	67.71	...	185.0	43.0	44.0	48.0	62.0	55.0
4	AF	2	Afghanistan	2514	Maize and products	5521	Feed	1000 tonnes	33.94	67.71	...	120.0	208.0	233.0	249.0	247.0	195.0
5	AF	2	Afghanistan	2514	Maize and products	5142	Food	1000 tonnes	33.94	67.71	...	231.0	67.0	82.0	67.0	69.0	71.0
6	AF	2	Afghanistan	2517	Millet and products	5142	Food	1000 tonnes	33.94	67.71	...	15.0	21.0	11.0	19.0	21.0	18.0
7	AF	2	Afghanistan	2520	Cereals, Other	5142	Food	1000 tonnes	33.94	67.71	...	2.0	1.0	1.0	0.0	0.0	0.0
8	AF	2	Afghanistan	2531	Potatoes and products	5142	Food	1000 tonnes	33.94	67.71	...	276.0	294.0	294.0	260.0	242.0	250.0
9	AF	2	Afghanistan	2536	Sugar cane	5521	Feed	1000 tonnes	33.94	67.71	...	50.0	29.0	61.0	65.0	54.0	114.0
10	AF	2	Afghanistan	2537	Sugar beet	5521	Feed	1000 tonnes	33.94	67.71	...	0.0	0.0	0.0	0.0	0.0	0.0

## - 사용자 정의 함수

### 1. 기상 데이터를 찾아주는 함수

**dataSearch(Month, Day, Weather\_elements, Location, Precipitation)**

pandas로 기상데이터.csv 파일 오픈

date에 맞는 형식으로 문자열 생성

매개변수와 매칭되는 data들만 DataFrame 형태로 추출

**return DataFrame**

### 2. 그래프와 표를 출력하는 함수

**showOutput(Month, Day, Weather\_elements, Location, Precipitation)**

영문 month를 숫자로 매핑

weatherTable = dataSearch(매핑된 지역변수들)

weatherTable 기반으로 matplotlib.pyplot 사용해서 weatherPlot 생성

**return [DataFrame, figure]**

위 컴포넌트들을 다 구성했으면 `gradio.interface(showOutput, Gradio components, Matplotlib·Pandas components, examples)`로 `gradio demo`를 구성한다. `examples`는 입력 예시를 보여주는 테이블을 넣어 사용에 도움을 주는 역할을 한다. `Demo`를 다 구성했으면 `gradio.interface.launch()`로 실행한다.

## 2. Demo 코드

현재 Demo코드를 pseudo 코드 대신에 현재까지 작성한 코드를 보여준다.

```
import gradio as gr
import pandas as pd
import matplotlib.pyplot as plt

# search weather data that is in csv file
def dataSearch(month, day, weather_elements, location, precipitation):
    if location=='Seoul':
        df = pd.read_csv('서울데이터파일.csv')
    elif location=='Washington':
        df = pd.read_csv('워싱턴 데이터 파일.csv')
    if precipitation:
        weather_elements.append('precipitation')
    if day in ['1','2','3','4','5','6','7','8','9']:
        today = '2022-'+month+'-0'+day
    else:
        today = '2022-'+month+'-'+day

    df1 = df[df.date == today]
    columns = ['location', 'date', 'time'] + weather_elements
    df2 = df1.loc[:, columns]

    return df2

# show weather data in plot using matplotlib
def showOutput(month, day, weather_elements, location, precipitation):
    if month=='January':
        month = '01'
    elif month=='February':
        month = '02'
    elif month=='March':
        month = '03'
    elif month=='April':
        month = '04'
    elif month=='May':
        month = '05'
    elif month=='June':
        month = '06'
    elif month=='July':
        month = '07'
    elif month=='August':
```

```

    month = '08'
elif month=='September':
    month = '09'
elif month=='October':
    month = '10'
elif month=='November':
    month = '11'
elif month=='December':
    month = '12'

weatherTable = dataSearch(month, day, weather_elements, location, precipitation)

xname = '2022'+month+day
plt.title("2022년 기상 그래프")
plt.xlabel(xname)
plt.ylabel("날씨 요소")
plt.legend(loc = "upper left")
plt.xticks(size=10, rotation=45)

weatherPlot = plt.figure(figsize=(10, 10))

y_value=[0]*len(weather_elements)

for i in range(len(weather_elements)):
    y_value[i] = weatherTable[weather_elements[i]]

x_value = weatherTable['time']

if 'humidity' in weather_elements:
    humidity_index = weather_elements.index('humidity')
    if weather_elements[humidity_index] != weather_elements[-1]:
        temp = weather_elements[humidity_index]
        weather_elements[humidity_index] = weather_elements[-1]
        weather_elements[-1] = temp

if len(weather_elements) == 1:
    plt.title("2022년 기상 그래프")
    plt.xlabel(xname)
    plt.ylabel(weather_elements)
    plt.legend(loc = "upper left")
    plt.xticks(size=10, rotation=45)

```

```

plt.bar(x_value, y_value[-1], color='blue')

elif len(weather_elements) == 2:
    ax1 = weatherPlot.add_subplot(1,1,1)
    ax1.plot(x_value, y_value[0], color='red', marker = "o")
    ax1_sub = ax1.twinx()
    ax1_sub.bar(x_value, y_value[-1], color='blue')

elif len(weather_elements) == 3:
    ax1 = weatherPlot.add_subplot(2, 1, 1)
    ax2 = weatherPlot.add_subplot(2, 1, 2)
    ax1.plot(x_value, y_value[0], color='red', marker = "o")
    ax1_sub = ax1.twinx()
    ax1_sub.plot(x_value, y_value[1], color='skyblue', marker = "o")
    ax2.bar(x_value, y_value[-1], color='blue')

elif len(weather_elements) == 4:
    ax1 = weatherPlot.add_subplot(2, 1, 1)
    ax2 = weatherPlot.add_subplot(2, 1, 2)
    ax1.plot(x_value, y_value[0], color='red', marker = "o")
    ax1_sub = ax1.twinx()
    ax1_sub.plot(x_value, y_value[1], color='skyblue', marker = "o")
    ax2.bar(x_value, y_value[-1], color='blue')
    ax2_sub = ax2.twinx()
    ax2_sub.plot(x_value, y_value[2], color='gray', marker = "o")

else:
    if len(weather_elements) == 1:
        plt.plot(x_value, y_value[0], color='red', marker='o')

    elif len(weather_elements) == 2:
        ax1 = weatherPlot.add_subplot(1,1,1)
        ax1.plot(x_value, y_value[0], color='red', marker='o')
        ax1_sub = ax1.twinx()
        ax1_sub.plot(x_value, y_value[1], color='skyblue', marker='o')

    elif len(weather_elements) == 3:
        ax1 = weatherPlot.add_subplot(2, 1, 1)
        ax2 = weatherPlot.add_subplot(2, 1, 2)
        ax1.plot(x_value, y_value[0], color='red', marker = "o")
        ax1_sub = ax1.twinx()

```

```

ax1_sub.plot(x_value, y_value[1], color='skyblue', marker = "o")
ax2.plot(x_value, y_value[2], color='gray', marker = "o")

return [weatherTable, weatherPlot]

# make gradio interface
output = gr.Plot()
output2 = gr.Dataframe()

demo = gr.Interface(
    fn=showOutput,
    inputs=[
        gr.Dropdown(["January", "February", "March", "April", "May", "June",
                    "July", "August", "September", "October", "November", "December"],
                    label="Month", info="Select Months"),
        gr.Dropdown(["1", "2", "3", "4", "5", "6", "7", "8", "9", "10",
                    "11", "12", "13", "14", "15", "16", "17", "18", "19", "20",
                    "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31"], label="Day",
                    info="Select Day"),
        gr.CheckboxGroup(["temperature", "wind", "humidity", "air_pressure"],
                        label="Weather element", info="Choose weather element"),
        gr.Radio(["Washington", "Seoul"],
                label="Location", info="Choose location"),
        gr.Checkbox(label="precipitation?"),
    ],
    outputs=[output2, output]
)

if __name__ == "__main__":
    demo.launch(debug=True)

```

#### - 추진 전략

Demo 제작을 빠르게 진행하면서 Demo에 대한 구체적인 설명이 포함된 문서들을 작성한다.

Demo에 대해 설명 시 Demo를 직접 사용해보는 animated gif나 영상을 만들어 gradio 측에서 쉽게 이해할 수 있게 한다. 문서 같은 경우에는 영어 형태로 번역하여 github에 pull request를 보내고, gradio 측과 소통을 하며 요구에 맞는 형태로 수정 및 보완을 진행한다. 이때에 gradio 측의 답변이 늦어질 수 있으므로 최대한 빠르게 pull request 한다.

Demo 제작에 사용한 코드(ipynb파일) 및 문서는 github에 올려 모든 사람들이 볼 수 있게 한다.



## - 테스트

### 1. 단위 테스트

#### - **dataSearch** 함수 테스트

input 데이터에 알맞은 csv파일로 읽어온 데이터프레임을 알맞게 가공하여 return 하는지를 확인한다.

- **Test data**

**month** = "01"

**day** = "11"

**weather\_elements** = ["temperature"]

**location** = "Seoul"

**precipitation** = False

**month** = "12"

**day** = "30"

**weather\_elements** = ["temperature", "air\_pressure"]

**location** = "Washington"

**precipitation** = True

- 테스트 결과

#### 1번 Test

Seoul의 날씨데이터는 1시간 간격으로 측정되어 있다.

	location	date	time	temperature
219	Seoul	2022-01-11	12:00:00 AM	-9.3
220	Seoul	2022-01-11	1:00:00 AM	-8.5
221	Seoul	2022-01-11	2:00:00 AM	-7.6
222	Seoul	2022-01-11	3:00:00 AM	-7.0
223	Seoul	2022-01-11	4:00:00 AM	-5.9
224	Seoul	2022-01-11	5:00:00 AM	-5.4
225	Seoul	2022-01-11	6:00:00 AM	-4.9
226	Seoul	2022-01-11	7:00:00 AM	-4.8
227	Seoul	2022-01-11	8:00:00 AM	-5.5
228	Seoul	2022-01-11	9:00:00 AM	-6.7
229	Seoul	2022-01-11	10:00:00 AM	-7.5
230	Seoul	2022-01-11	11:00:00 AM	-8.3
231	Seoul	2022-01-11	12:00:00 PM	-9.0
232	Seoul	2022-01-11	1:00:00 PM	-9.6
233	Seoul	2022-01-11	2:00:00 PM	-10.0
234	Seoul	2022-01-11	3:00:00 PM	-10.2
235	Seoul	2022-01-11	4:00:00 PM	-10.7
236	Seoul	2022-01-11	5:00:00 PM	-11.0
237	Seoul	2022-01-11	6:00:00 PM	-11.3
238	Seoul	2022-01-11	7:00:00 PM	-11.3
239	Seoul	2022-01-11	8:00:00 PM	-11.1
240	Seoul	2022-01-11	9:00:00 PM	-11.0
241	Seoul	2022-01-11	10:00:00 PM	-11.2
242	Seoul	2022-01-11	11:00:00 PM	-11.2

## 2번 Test

Washington의 날씨데이터는 6시간 간격으로 측정되어 Seoul의 데이터에 비해 양이 적다.

	location	date	time	temperature	air_pressure	precipitation
1435	Washington	2022-12-30	12:00:00 AM	6.7	1015.1	0.0
1436	Washington	2022-12-30	6:00:00 AM	0.0	1014.8	0.0
1437	Washington	2022-12-30	12:00:00 PM	-1.7	1014.8	0.0
1438	Washington	2022-12-30	6:00:00 PM	15.0	1012.1	0.0

### - showOutput 함수 테스트

input 데이터를 형식에 맞게 매핑하고 dataSearch함수를 통해 가져온 날씨 데이터를 그래프와 데이터프레임의 형태로 잘 나타내는지 확인한다.

#### • Test data

month = "January"

day = "11"

weather\_elements = ["temperature", "wind"]

location = "Seoul"

precipitation = False

month = "October"

day = "1"

weather\_elements = ["air\_pressure", "humidity"]

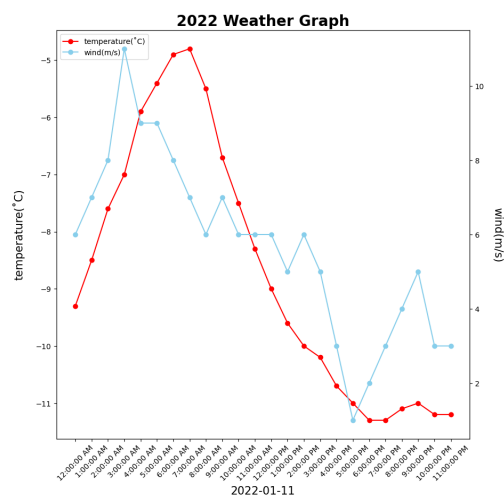
location = "Washington"

precipitation = True

#### • 테스트 결과

### 1번 Test

	location	date	time	temperature(°C)	wind(m/s)
219	Seoul	2022-01-11	12:00:00 AM	-9.3	6.0
220	Seoul	2022-01-11	1:00:00 AM	-8.5	7.0
221	Seoul	2022-01-11	2:00:00 AM	-7.6	8.0
222	Seoul	2022-01-11	3:00:00 AM	-7.0	11.0
223	Seoul	2022-01-11	4:00:00 AM	-5.9	9.0
224	Seoul	2022-01-11	5:00:00 AM	-5.4	9.0
225	Seoul	2022-01-11	6:00:00 AM	-4.9	8.0
226	Seoul	2022-01-11	7:00:00 AM	-4.8	7.0
227	Seoul	2022-01-11	8:00:00 AM	-5.5	6.0
228	Seoul	2022-01-11	9:00:00 AM	-6.7	7.0
229	Seoul	2022-01-11	10:00:00 AM	-7.5	6.0
230	Seoul	2022-01-11	11:00:00 AM	-8.3	6.0
231	Seoul	2022-01-11	12:00:00 PM	-9.0	6.0
232	Seoul	2022-01-11	1:00:00 PM	-9.6	5.0
233	Seoul	2022-01-11	2:00:00 PM	-10.0	6.0
234	Seoul	2022-01-11	3:00:00 PM	-10.2	5.0
235	Seoul	2022-01-11	4:00:00 PM	-10.7	3.0
236	Seoul	2022-01-11	5:00:00 PM	-11.0	1.0
237	Seoul	2022-01-11	6:00:00 PM	-11.3	2.0
238	Seoul	2022-01-11	7:00:00 PM	-11.3	3.0
239	Seoul	2022-01-11	8:00:00 PM	-11.1	4.0
240	Seoul	2022-01-11	9:00:00 PM	-11.0	5.0
241	Seoul	2022-01-11	10:00:00 PM	-11.2	3.0
242	Seoul	2022-01-11	11:00:00 PM	-11.2	3.0

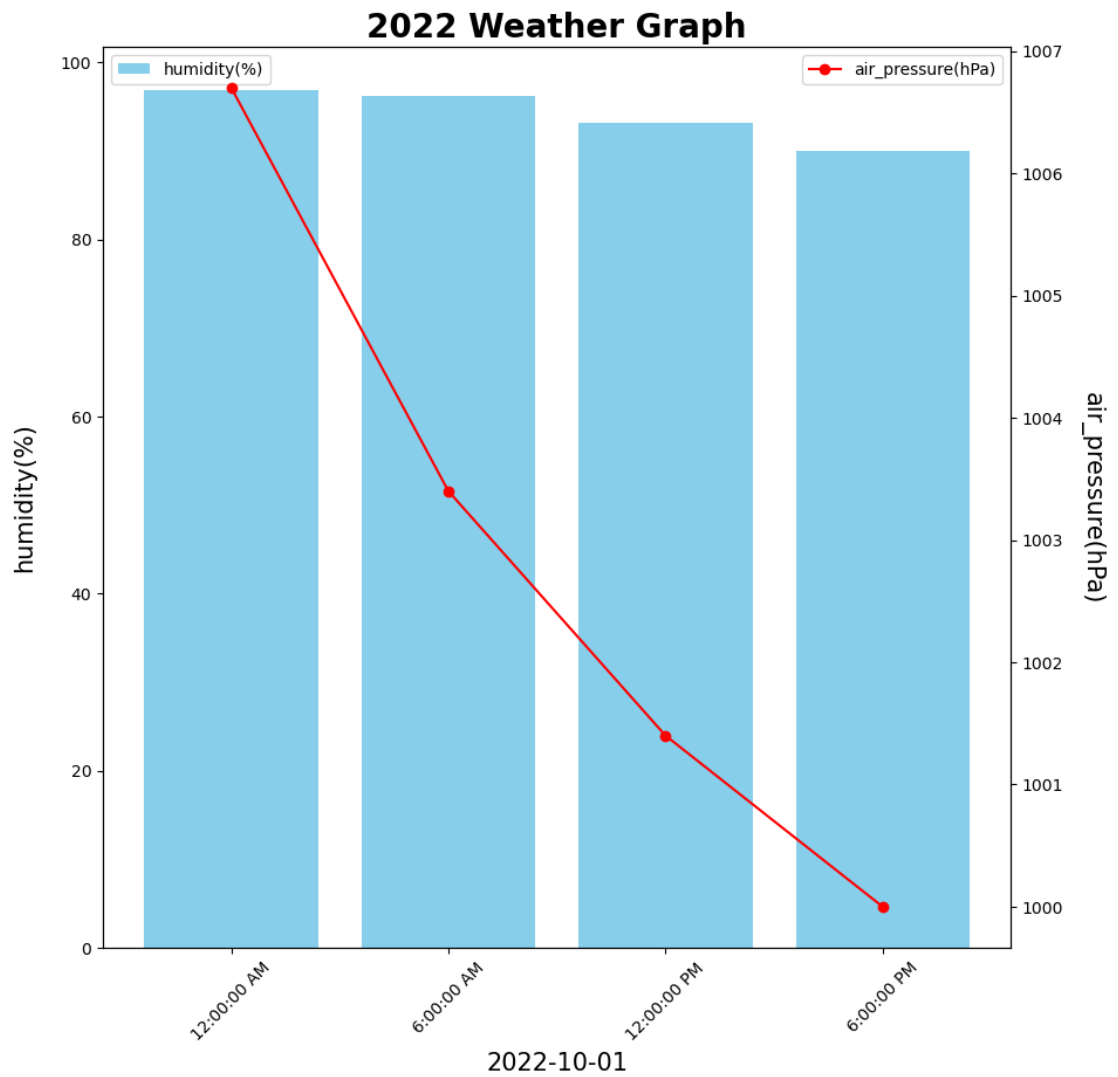


## 2번 Test

	location	date	time	air_pressure(hPa)	humidity(%)
1077	Washington	2022-10-01	12:00:00 AM	1006.7	96.9
1078	Washington	2022-10-01	6:00:00 AM	1003.4	96.2
1079	Washington	2022-10-01	12:00:00 PM	1001.4	93.1
1080	Washington	2022-10-01	6:00:00 PM	1000.0	90.0

	precipitation(mm)
1077	1
1078	16
1079	4
1080	1



## 2. 통합 테스트

### - Gradio Demo 테스트

Gradio 인터페이스 구성을 한 후 Gradio의 데모의 ui를 이용해 테스트 데이터를 입력하고 결과값이 잘 출력되는지를 확인한다.

- 테스트 결과

## 1번 Test

### Input

Month  
Select Months

April

Day  
Select Day

4

Weather element  
Choose weather element

☐ temperature(°C) ☒ wind(m/s) ☒ humidity(%) ☐ air\_pressure(hPa)

Location  
Choose location

☐ Washington ☒ Seoul

☐ precipitation?

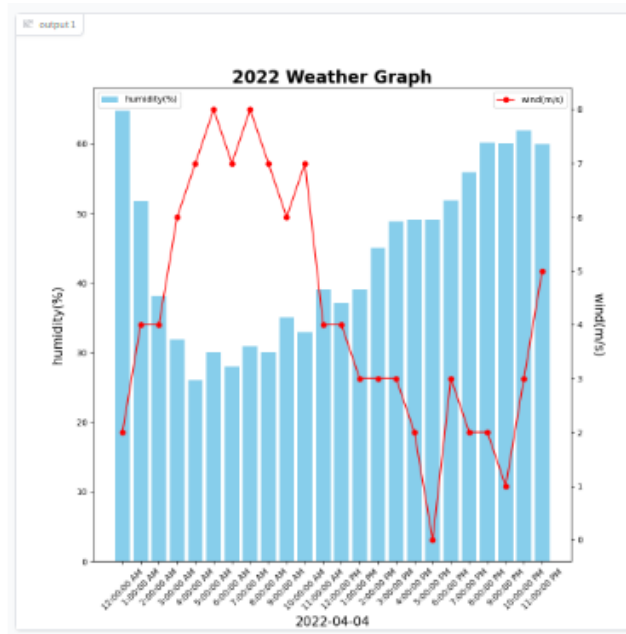
클리어

제출하기

### Output

output 3

location	date	time	wind(m/s)	humidity(%)
Seoul	2022-04-04	12:00:00 AM	2	64.8
Seoul	2022-04-04	1:00:00 AM	4	51.8
Seoul	2022-04-04	2:00:00 AM	4	38.2
Seoul	2022-04-04	3:00:00 AM	6	31.9
Seoul	2022-04-04	4:00:00 AM	7	26.1
Seoul	2022-04-04	5:00:00 AM	8	30
Seoul	2022-04-04	6:00:00 AM	7	28
Seoul	2022-04-04	7:00:00 AM	8	30.9
Seoul	2022-04-04	8:00:00 AM	7	30
Seoul	2022-04-04	9:00:00 AM	6	35.1
Seoul	2022-04-04	10:00:00 AM	7	33
Seoul	2022-04-04	11:00:00 AM	4	39.1
Seoul	2022-04-04	12:00:00 PM	4	37.1
Seoul	2022-04-04	1:00:00 PM	3	39.1
Seoul	2022-04-04	2:00:00 PM	3	45.1
Seoul	2022-04-04	3:00:00 PM	3	48.9
Seoul	2022-04-04	4:00:00 PM	2	49.1
Seoul	2022-04-04	5:00:00 PM	0	49.1
Seoul	2022-04-04	6:00:00 PM	3	51.9
Seoul	2022-04-04	7:00:00 PM	2	55.9
Seoul	2022-04-04	8:00:00 PM	2	60.2
Seoul	2022-04-04	9:00:00 PM	1	60.1
Seoul	2022-04-04	10:00:00 PM	3	61.9
Seoul	2022-04-04	11:00:00 PM	5	60



2번 Test

Input

Month  
Select Months

June

Day  
Select Day

10

Weather element  
Choose weather element

☒ temperature(°C)

☐ wind(m/s)

☐ humidity(%)

☒ air\_pressure(hPa)

Location  
Choose location

☒ Washington

☐ Seoul

☒ precipitation?

클리어

제출하기

Output

output 0

location	date	time	temperature(°C)	air_pressure(hPa)	preci
Washington	2022-06-10	12:00:00 AM	23.3	999.4	0
Washington	2022-06-10	6:00:00 AM	14.4	1001.4	0
Washington	2022-06-10	12:00:00 PM	18.9	1002.4	0
Washington	2022-06-10	6:00:00 PM	26.7	1001.7	0

output 1

