

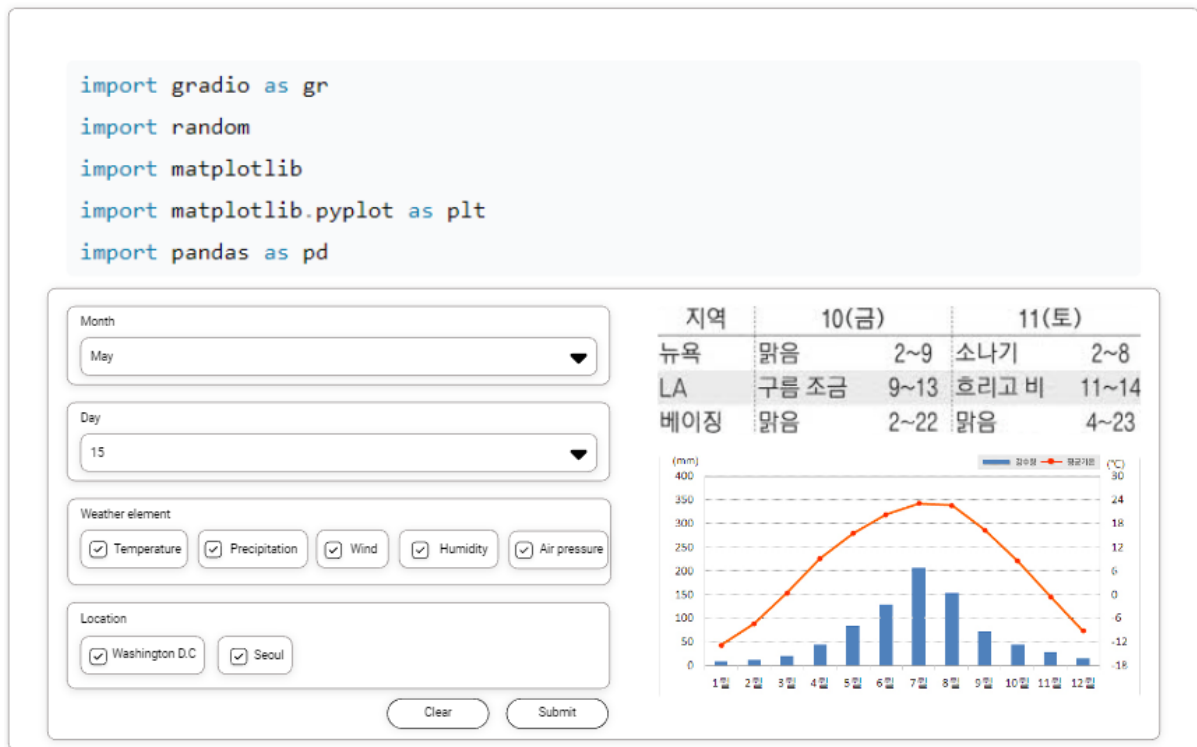
오픈소스 **SW** 기여

Detail Design

32182775 위성준
32183698 이현기

1) 전체적인 시스템 구조

ui 스케치를 통해 구성한 데모이다.



- Gradio components
 - Month, Day -> Dropdown
 - Weather elements (temperature, wind, humidity, air pressure) -> CheckboxGroup
 - Location (Washington D.C, Seoul) -> Radio
 - Precipitation -> Checkbox

The sketch shows the layout of the UI components:

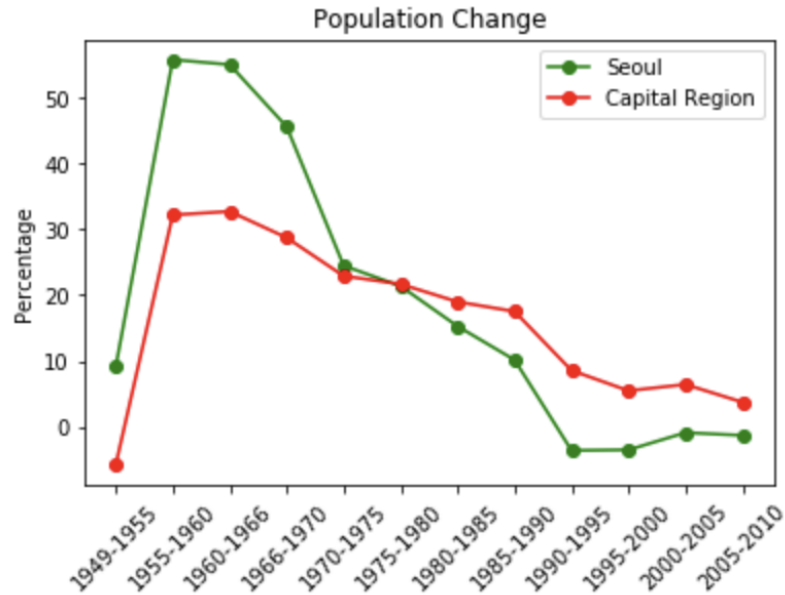
- Month:** Select Months dropdown.
- Day:** Select Day dropdown.
- Weather element:** Choose weather element section with checkboxes for temperature, wind, humidity, and air_pressure.
- Location:** Choose location section with radio buttons for Washington and Seoul.
- precipitation?** A checkbox at the bottom.
- Buttons:** 클리어 (Clear) and 제출하기 (Submit).

- Matplotlib components

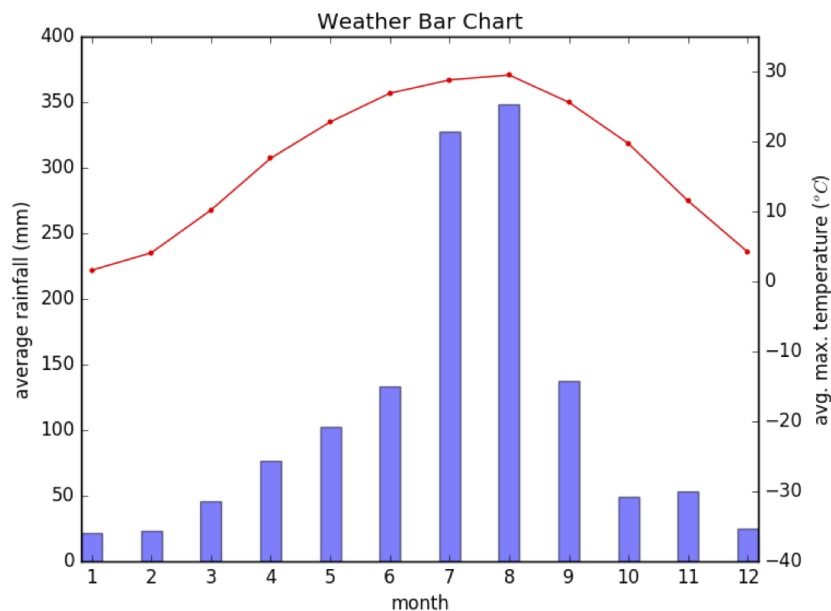
선 그래프 (기온, 풍속, 기압) -> matplotlib.pyplot의 plot method 이용.
 막대 그래프 (습도) -> matplotlib.pyplot의 bar method 이용.

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, plt]

위 컴포넌트들을 다 구성했으면 gradio.interface(showOutput, Gradio components, Matplotlib·Pandas components, examples)로 gradio demo를 구성한다. examples는 입력 예시를 보여주는 테이블을 넣어 사용에 도움을 주는 역할을 한다. Demo를 다 구성했으면 gradio.interface.launch()로 실행한다.

- Demo 상세 설계 코드

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

def dataSearch(Month, Day, Weather_elements, Location, Precipitation):
    df = pd.read_csv('날씨데이터파일.csv')

    today = Month와 Day로 파일 형식에 맞게 문자열 생성
    df1 = df[df.date == today]
    columns = ['Location', '날짜', '시간'] + Weather_elements
    df2 = df1.loc[:, columns]

    return df2

def showOutput(Month, Day, Weather_elements, Location, Precipitation):
    month = Month를 해당 월 숫자로 매핑

    weatherTable = dataSearch(month, day, elements, location)

    plt.title("2022년 기상 그래프")
    plt.xlabel("2022년 해당 날짜")
    plt.ylabel("날씨 요소")
    plt.legend(loc = "upper left")

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

    x_value = 해당 날짜에 측정된 시간들
    y_value1 = 해당 날짜의 기온
    y_value2 = 해당 날짜의 풍속
    y_value3 = 해당 날짜의 습도
    y_value4 = 해당 날짜의 기압

    선택되는 Weather_elements의 개수에 따라 그래프 개수를 다르게 설정
    최대 2개 plot 생성, humidity는 bar그래프
    ax1 = weatherPlot.add_subplot(2, 1, 1)
    ax2 = weatherPlot.add_subplot(2, 1, 2)

    ax1.plot(x_value, y_value1, color='red', marker = "o")
    ax1_sub = ax1.twinx()
```

```

ax1_sub.plot(x_value, y_value2, color='sky', marker = "o")

ax2.bar(x_value, y_value3, color='blue')
ax2_sub = ax2.twinx()
ax2_sub.plot(x_value, y_value4, color='gray', marker = "o")

return [weatherTable, weatherPlot]

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()

```

2) 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.