<날씨 정보 시각화>

20181288 윤성호

이번 프로젝트에서는 총 8개의 기능을 담고있는 날씨 정보 시각화 프로그램을 만들었다. 수업시간에 배운 여러가지 내용들을 활용하여 만들었으며, 어느 부분을 이용하여 만들었는지는 아래에서 자세히 설명하도록 하겠다. 또한, 코드에 대한 설명은 주석으로 자세히 달아놓았으므로 보고서에서는 코드에 대한 설명보다 어떤 기능을 구현하였는지와 작동하는 방식에 설명하는데에 중점을 두도록 하겠다.

먼저, 프로그램은 크게 한 개의 함수와, 두개의 클래스로 이루어져 있다. checkRunningTime함수, UTIL클래스, Weather 클래스가 그것들이다. 먼저, checkRunningTime 함수는 함수의 실행시간을 출력해주는 것으로, Weather 클래스의 모든 매서드들에 적용하여 각 매서드의 실행시간을 출력하도록 하였다. 그리고 UTIL 클래스에는 Weather 클래스의 매서드들을 구현하는데 필요한 함수들을 저장해놓은 클래스로, 모든 매서드를 정적 매서드로 구현하였다. UTIL 클래스에서 작성한 함수들의 역할은 주석으로 자세히 달아놓았다.

그리고, 날씨 정보 시각화를 구현한 Weather 클래스이다. 이 클래스에는 총 8개의 매서드가 있으며 각각의 매서드는 아래와 같다. 이 매서드들은 각각 아래에서 설명하도록 하겠다.

* today\_temp : 오늘의 날씨 출력
* future\_temp : 내일/모레/글피 날씨 출력
* past\_temp : 과거의 날씨 출력
* pastToday\_temp : 지정한 날짜의 과거 모든 날씨 출력
* annual\_average\_temp : 연평균 기온 추이 출력
* max\_temp : 구간 최고기온 정보 출력
* min\_temp : 구간 최저 기온 정보 출력
* recommend\_clothes : 내일의 예상 기온에 따른 옷차림 추천

<이용한 라이브러리(dependencies)>

날씨 정보를 시각화 하는 과정을 구현하기 위해서는 여러가지 라이브러리를 불러와서 활용해야 한다. 아래 사진은 프로젝트를 진행하면서 필요한 라이브러리들이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 1]

* import requests : get 방식으로 웹사이트에 접근을 하며 통신을 하기 위해 필요한 라이브러리이다. 기상청 날씨누리에 접근하기 위해 임포트하였다.
* From bs4 import BeautifulSoup : 웹사이트에 get 방식으로 접근한 이후, 웹사이트의 내용을 정적 크롤링하기 위해 임포트하였다.
* From selenium import webDriver : 웹사이트에서 javascript에 의해 같은 URL상에 클릭이나 스크롤을 이용해 동적으로 변화한다면, BeautifulSoup만을 이용한다면 크롤링할 수 없는 정보가 존재한다. 이러한 정보들을 가져오기 위해서는 동적 크롤링이 필요하고, 이를 위해 임포트하였다. 주피터 노트북에 기본적으로 설치되어 있지 않기 때문에 터미널에서 pip install selenium을 통해 설치해주었다.
* from selenium.webdriver.chrome.service import Service : 다운받은 크롬드라이버에 접근하는데 도와주는 라이브러리이다. 주피터 노트북에 기본적으로 설치되어 있지 않기 때문에 터미널에서 pip install webdriver-manager을 통해 설치해주었다.
* from webdriver\_manager.chrome import ChromeDriverManager : 버전 변경에 상관없이 현재 OS에 설치된 크롬을 이용할 수 있게 해주는 라이브러리이다. 이 또한 pip install webdriver-manager을 통해 설치해주었다.
* Import re : 웹사이트 크롤링이 완료되었으면 정제되지 않은 긴 문자열형태의 자료를 받아오게 된다. 그 중 필요한 부분만을 가져오려면 정규표현식을 이용하는 것이 편리하기 때문에 임포트하였다.
* Import numpy as np : 리스트의 합이나 평균을 구하는 일이 잦은 프로그램인만큼, np.ndarray을 이용하게 되면 매우 효율적으로 계산할 수 있다고 생각하였고, 그렇기에 임포트하였다.
* Import matplotlib.pyplot as plt : 날씨 정보를 시각화 하기 위해서 그래프를 그려주어야 하는데, 그를 도와주는 라이브러리이다.
* From copy import deepcopy : 리스트를 얕은 복사하게 된다면 정보의 손실이 날 우려가 있기 때문에 중요한 정보는 deepcopy를 이용해 복사하였다.
* From datetime import datetime, timedelta : 프로그램 특성상 날짜를 다루는 일이 매우 빈번하다. 그렇기에 datetime을 임포트 하였으며, 날짜 간 연산도 자주 이루어지므로 timedelta 또한 임포트하였다.
* Import time : checkRunningTime 함수를 작성하는데도 필요하고, 날짜를 다루는 과정에서도 쓰이기 때문에 임포트하였다.

< 초기화면 >

활용단원 : OOP, 예외처리

텍스트이(가) 표시된 사진

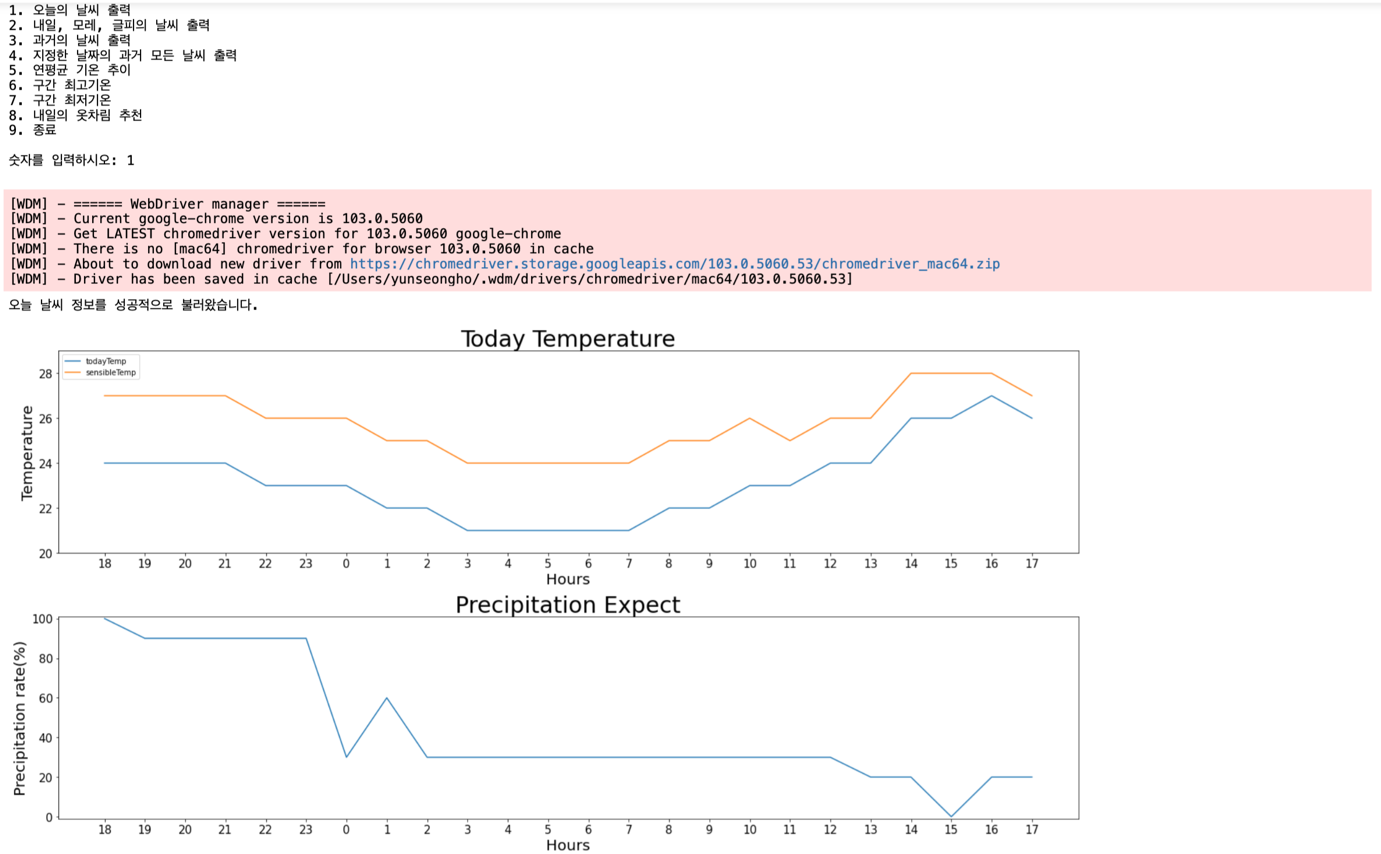
자동 생성된 설명

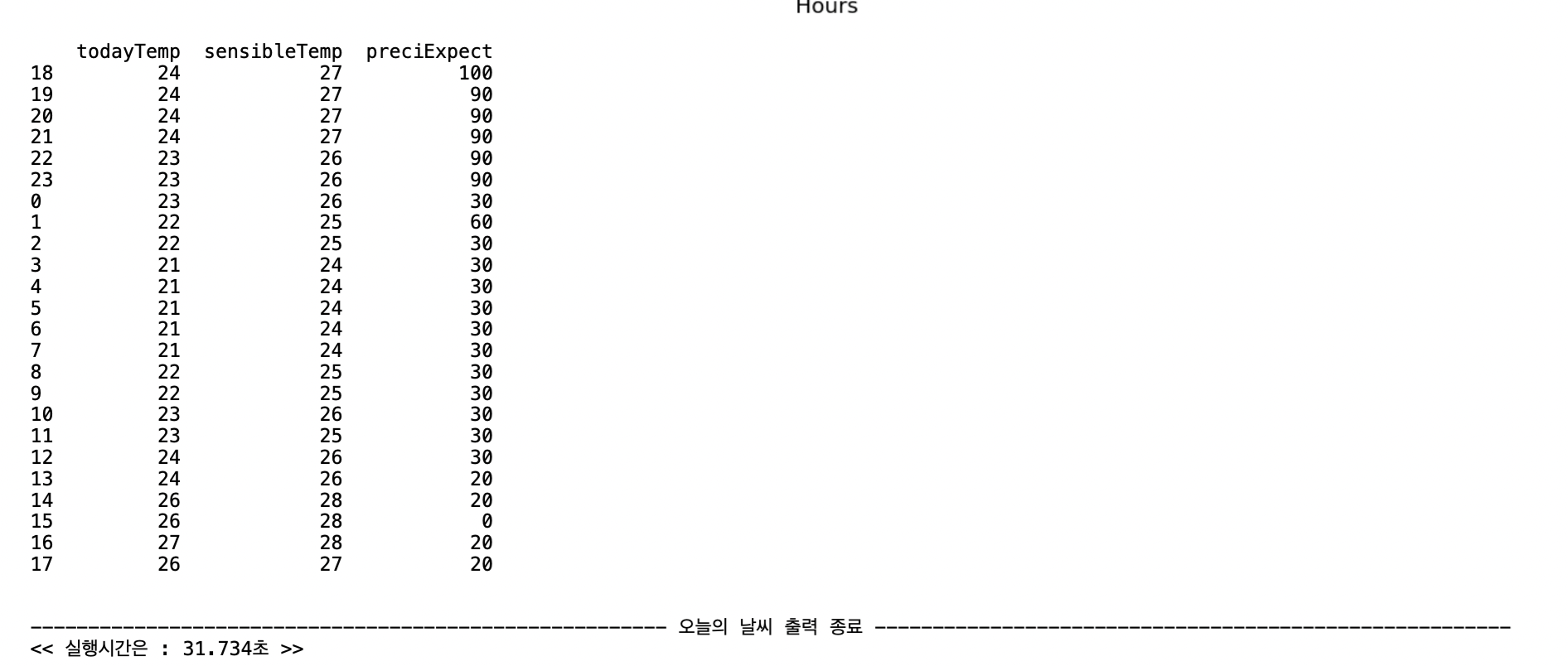
[사진 2]

[사진 2]은 날씨 정보 시각화 프로그램을 실행시킨 초기화면이다. Weather이라는 이름의 클래스에 필요한 매서드들을 모두 작성하였고, 이 매서드를 활용하여 날씨 정보를 얻기 위해서는 Weather 객체를 생성해야 한다. 객체를 생성하면 생성자로 위의 정보를 출력하도록 하였으며, 숫자도 하나 입력받도록 하였다. 이 과정은 while(true)문에 의해 무한루프를 돌도록 설정되었으며, “9”를 눌러 무한루프를 빠져나오도록 했다. 숫자는 1~9까지의 정수만을 입력받는것이 가능하며, 다른 입력이 주어지면 try, except문에 의해 예외처리 되게 되고 다시 새로운 숫자를 입력하도록 하게 한다. 1~9번이 입력되면, 해당하는 기능이 실행되도록 코드를 작성하였다.

<1번 매서드 : 오늘의 날씨 출력>

활용단원 : OOP, 정규표현식, numpy, matplotlib, 실행시간체크, 예외처리, pandas





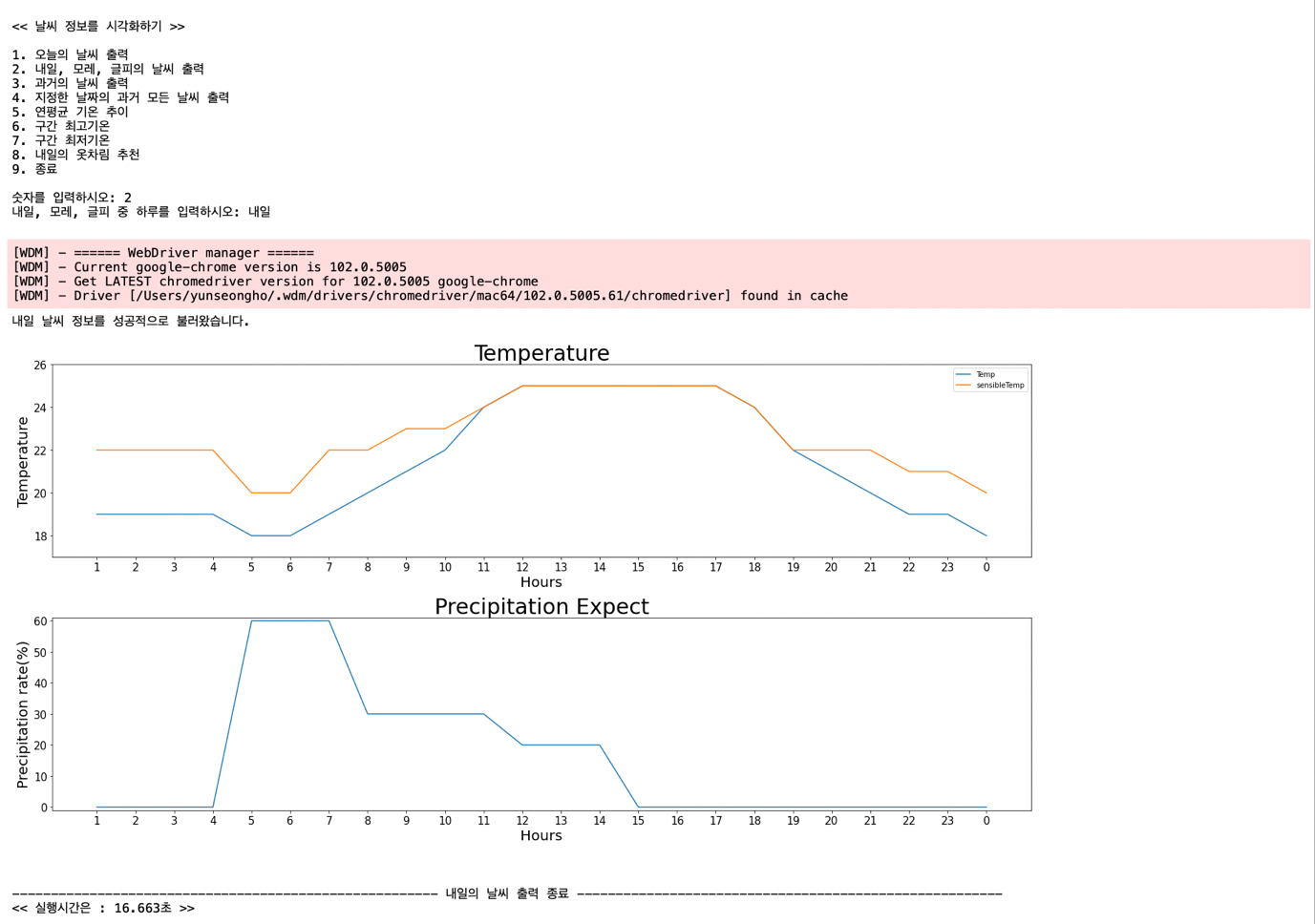
[사진 3]

첫번째 매서드는 기상청 날씨누리에서 현재시간부터 24시간 이후까지의 시간까지의 기온, 체감온도, 강수확률을 모두 불러와서, 그래프로 나타내주었다. 초기화면에서 1을 누르면 실행된다.기상청 사이트에는 기온의 그래프는 나와있지만, 체감온도와 강수확률에 대한 그래프는 나와있지 않아서 전체적인 경향을 확인하기가 어려웠다. 또한, 강수확률의 경우, 현재 시간으로부터 4~5시간이후까지는 정보가 제공되지 않는다. 이 시간대에 대한 정보는 강수량 정보에서 확인이 가능하며, 비가 오고있다면 강수량 칸에 mm단위의 강수량정보가 제공되고, 어떤 경우에는 빈칸인경우도 있다. 이러한 빈칸에 대한 강수확률값 처리에 대한 고민이 있었으며, 이는 강수량이 빈칸이 아니면 비가 오고있다는 말이므로 강수확률을 100으로 주었고, 빈칸이라면 강수확률을 0으로 주어서 처리하였다. 근접한 미래에 대한 강수량정보가 제공되는 만큼 정확하다고 믿고서 할당한 값들이다. [사진 3]의 그래프는 현재 보고서를 작성중인 2022년 6월 23일 17시 56분에 1번 매서드를 실행시킨 결과이다. 첫번째 그래프는 가로축은 시간, 세로축은 온도이며, 두번째 그래프는 가로축은 시간, 세로축은 강수확률이다. 현재 가로축은 18부터 17로 설정되어 있는데, 이는 2022년 6월 23일 18시 ~ 2022년 6월 24일 17시 까지의 날씨 정보를 그래프로 나타낸 것이다. 가로축에 나타나는 시간은 프로그램 실행시간에 따라 동적으로 달라지도록 프로그램을 작성하였다. 아래 강수확률 그래프의 경우, 현재 비가 오고 있기 때문에 23시의 강수확률을 100이 할당되었고, 그 이후 몇시간 동안은 비소식이 없기 때문에 강수확률을 0으로 설정하였다. 그 이후 시간에 대해서는 강수확률정보가 제공되므로 제공된 강수확률 정보를 토대로 그래프를 그렸다. 이러한 데이터를 pandas를 활용해서 표의 형태로도 나타내보았다.

오늘의 날씨 기능은 BeautifulSoup 라이브러리를 통해 정보를 크롤링해서 진행할 계획을 세웠었다. 그러나 오늘의 날씨는 스크롤에 의해 같은 URL임에도 불구하고 다른 정보를 화면에 출력한다. 이는 javascript를 이용하여 동적으로 웹사이트가 만들어지고 있는 것이므로 BeautifulSoup 라이브러리를 통해 크롤링은 불가능하다. 따라서, selenium 라이브러리와 크롬 드라이버를 이용해 원격으로 웹사이트에 접근해 정보를 끌어와야 하는 동적 크롤링 과정이 필요하다는 것을 구글링을 통해 새롭게 확인할 수 있게 되었다. 오늘의 날씨 정보를 표시해주었다면 함수를 종료한다는 메세지를 출력한 후 함수를 종료한다. checkRunningTime 함수에 의해 실행시간 또한 표시되게 하였다.

<2번 매서드 : 내일/모레/글피의 날씨 출력>

활용단원 : OOP, 정규표현식, numpy, matplotlib, 실행시간체크, 예외처리



[사진 4]

두번째 매서드는 기상청 날씨누리 웹사이트에서 제공하는 내일, 모레, 글피 중 하루의 날씨정보를 불러와서 그래프로 나타낸 것이다. 초기화면에서 2를 누르면 실행된다. 첫번째 매서드와 전체적으로 유사하지만 다른점은 내일, 모레, 글피 중 하루를 입력해야 하는 부분이 있는 것이다. 이 과정에서 내일, 모레, 글피 가 아닌 다른 입력이 주어진다면 입력오류임을 알려주는 문구를 출력해주고 매서드를 종료한다. 입력이 정상적으로 주어졌다면 기상청 웹사이트에 selenium 라이브러리와 크롬 드라이버를 이용해 접근하여 정보를 동적 크롤링해온 후, 크롤링 해온 정보중 입력한 날짜의 정보만을 뽑아낸다. Html tag를 활용하여 정보를 뽑아왔기 때문에 크롤링을 통해 받아오는 기온, 체감온도, 강수확률 정보는 현재시간을 기준으로 글피까지 정보가 모두 리스트에 들어 있다. 이들을 잘 뽑아오기 위해 내일, 모레, 글피의 날씨정보가 저장되어 있는 리스트의 인덱스값을 indexes라는 딕셔너리에 저장해놓았다. 이를 통해 원하는 날씨 정보를 정제하는 과정에서 indexes의 값만을 확인하여 뽑아온다면, 원하는 날의 기온, 체감온도, 강수확률을 얻을 수 있다. 그래프를 그리는 과저은 첫번째 매서드와 같다. 두번째 매서드 또한 그래프를 그린 이후에는 함수를 종료한다는 메세지를 출력한 후 종료한다. checkRunningTime 함수에 의해 실행시간 또한 표시되게 하였다. [사진 4]는 내일/모레/글피 중 내일을 입력하여 기상청에서 제공하는 내일의 기온, 체감온도, 강수확률을 그래프로 나타낸 것이다.

<3번 매서드 : 과거의 날씨 출력>

활용단원 : OOP, 정규표현식, numpy, matplotlib, 실행시간체크, 예외처리

테이블이(가) 표시된 사진

자동 생성된 설명

[사진 5]

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 6]

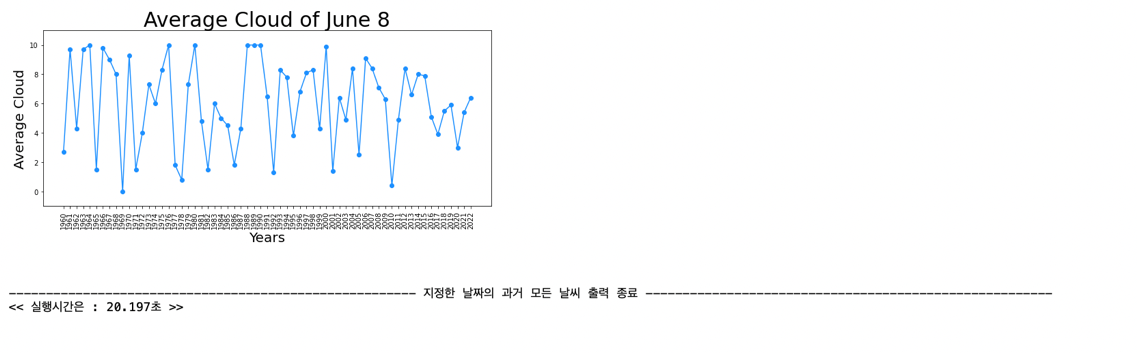
세번째 매서드는 1960년 1월 1일부터 어제 날짜중 하루를 입력하면, 해당 날짜의 평균기온, 최고기온, 최저기온, 평균운량, 일강수량을 출력한다. 초기화면에서 3을 누르면 실행된다. 만약, 잘못된 입력이 주어지면 “날짜 입력 오류. 날짜 범위(1960년 1월 1일 ~ 어제)만 입력이 가능합니다” 라는 에러 메세지를 출력하고 매서드를 종료한다. 이는 [사진 6]에서 확인할 수 있다. [사진 5]에서는 제대로 입력이 주어진 경우에 프로그램이 어떻게 작동하는 지 나타는 것이다. 해당 날짜의 평균기온, 최고기온, 최저기온, 평균운량, 일강수량을 모두 출력해주고, 그 이후 함수를 종료한다. 이 때, 날씨 정보들을 기상청 날씨누리에서 받아올 때 BeuatifulSoup 라이브러리를 활용하였다. 이는 첫번째, 두번째 매서드와는 달리 javascript를 통해 동적으로 웹사이트가 생성되는 것이 아니라, 화면에 서버로부터 이미 모든 정보를 받아온 상태이기 때문에 웹사이트의 정보를 그대로 긁어오는 정적 크롤링이 가능하다. Html tag을 이용해 크롤링해왔기 때문에 크롤링된 문자열은 원하는 정보만 들어있는 것이 아니다. 먼저, 해당 날짜의 정보가 들어있는 사이트에 접근하기 위해 URL을 분석하여, 입력된 날짜를 URL에 잘 삽입시켜서 웹사이트에 접근하였다. 웹사이트에는 달을 기준으로 정보가 되어 있기 때문에, 예를 들어, 2018년 3월 2일의 정보를 알고 싶다면 2018년 3월의 날씨 정보들이 있는 웹사이트에 접근해야 한다는 의미이다. 이후 크롤링을 진행하면 3월은 31일까지 있기 때문에 같은 형식의 데이터가 31개가 크롤링 되어 리스트에 저장되어 있을 것이다. 이 리스트는 1일부터 31일까지의 모든 날씨 정보가 저장되어 있으므로, 3월 2일의 날씨는 두번째(1번 인덱스)에 들어있을 것이다. 이러한 방식을 통해 과거의 날짜의 날씨정보를 크롤링해올 수 있다.

<4번 매서드 : 지정한 날짜의 과거 모든 날씨 출력>

활용단원 : OOP, 정규표현식, numpy, matplotlib, 실행시간체크, 예외처리

텍스트이(가) 표시된 사진

자동 생성된 설명



[사진 7]

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 8]

네번째 매서드는 월일을 입력하면 1960년부터 가장 최근년도까지의 입력한 월일의 날씨정보를 받아와서 평균기온, 최고기온, 최저기온, 평균운량을 그래프로 나타내주는 기능이다. 초기화면에서 4를 누르면 실행된다. 만약, [사진 8]과 같이 잘못된 입력이 주어지면 “날짜 오류. 날짜 범위(1월 1일 ~ 12월 31일)만 입력이 가능합니다.”라는 에러 메세지를 출력하고 함수를 종료한다. 만약, 입력이 제대로 주어졌다면 기상청 날씨누리의 모든 년도의 해당 날짜의 날씨 정보를 크롤링하여 리스트에 저장해준다. 예를 들어, [사진 7]과 같이 6월 8일을 입력하면 1960년 6월 8일부터 2022년 6월 8일의 날씨 정보를 BeautifulSoup를 이용하여 정적 크롤링을 통해 정보를 긁어온다. 이는 1960년 6월달의 날씨 정보가 저장되어 있는 URL에 접근하여, 크롤링을 통해 저장한 날씨정보를 리스트에 저장하고, 이 리스트에는 6월 1일부터 6월 30일까지의 날씨 정보가 저장이 되어 있을 것이다. 따라서, 6월 8일의 정보는 리스트의 8번째(7번 인덱스)에 들어있다고 생각할 수 있고, 이를 출력하면 된다. [사진 7]에 의하면 6월 8일의 평균기온은 2013년도에 가장 높았다. 실제로 2013년에 폭염 관련 뉴스가 굉장히 많이 나왔음을 검색을 통해 확인하였다. 이 기능을 구현할 때는 2월 29일이 문제가 되었었다. 2월 29일은 윤년에만 있는 날짜이기 때문에 모든 년도를 탐색하게 되면 에러가 떴다. 이는 조건문을 활용하여 2월 29일이 입력된 후 반복문을 돌 때 윤년이 아닌 날짜들은 무시하고 지나가도록 프로그램을 작성하여 처리하였다.

<5번 매서드 : 연평균 추이 그래프 출력>

활용단원 : OOP, 정규표현식, numpy, matplotlib, 실행시간체크, 예외처리

테이블이(가) 표시된 사진

자동 생성된 설명

[사진 9]

다섯 번째 매서드는 기상청 날씨누리에 있는 모든 날씨 정보를 받아와서 각 년도마다 연평균기온을 직접 구한 후, 그래프에 나타내주는 기능이다. 초기화면에서 5를 누르면 실행된다. 1960년 1월 1일부터 2021년 12월 31일까지의 각 날짜마다 날씨 정보를 모두 받아온다. 먼저, 각 년도를 반복문으로 돌며, 1월 1일부터 12월 31일까지 모든 날짜의 평균기온을 리스트에 저장한다. 이를 뽑아오기 위해서는 해당 월의 URL에 접근하여서 각 월마다 1일부터 30~31일까지의 날짜를 모두 탐색하며 평균기온만을 걸러낸다. 이를 위해서는 크롤링한 자료에서 어느 html tag에 평균기온이 둘러 쌓여있는지 파악한 후, 그 안의 정보들만을 뽑아냈다. 이렇게 해서 1년치 자료가 리스트에 들어왔다면, 넘파이 라이브러리의 매서드중 합을구해주는 매서드를 이용해서 합을 구한후 자료의 수로 나누어 평균을 구한다. 합을 구하는 연산이 자주 일어나므로 넘파이 ndarray를 이용하여 더 효율적인 연산이 가능하도록 했다. 그리고, 그렇게 구한 탐색년도 연평균기온을 avgTemps 배열에 추가한다. 이렇게 해서 기상청 날씨누리에서 찾을 수 있는 모든 날씨정보를 찾으면 1960년부터 2021년까지이므로 총 82년이고, 이는 약 30000일로 계산되므로, 약 30000개정도의 입력자료를 활용하여 날씨정보를 시각화한 것이다. 이렇게 생성된 그래프를 살펴보면 전체적으로 우상향하는 것을 알 수 있는데, 이는 대한민국에서도 지구온난화가 진행되고 있다는 것을 확인 할 수 있는 자료로 해석할 수 있다.

<6번 매서드 : 구간 최고기온 출력>

활용단원 : OOP, 정규표현식, numpy, matplotlib, 실행시간체크, , 예외처리

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 10]

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 11]

여섯 번째 매서드는 두개의 날짜를 입력하면 두 날짜 사이의 평균기온을 모두 불러와서 분석한 후, 평균기온이 가장 높은날짜를 찾고, 그날의 평균기온이 어땠는지 출력해준다. 또한, 입력한 날짜 사이의 평균기온을 그래프로 나타내주어 한눈에 기온의 변화가 어떻게 진행되는지 알 수 있도록 하였다. 초기화면에서 6을 누르면 실행된다. 만약, [사진 11]과 같이 날짜형식이 아닌 것이 입력되거나, 유효하지 않은 날짜가 입력되면 “날짜 오류. 두 날짜 모두 날짜 범위(1960년 1월 1일 ~ 어제)만 입력이 가능합니다.” 라는 에러 메세지를 출력해주고 함수를 종료한다. 만약, 두 날짜 모두 유효하다면 두 날짜 사이의 날씨정보를 기상청 웹사이트에서 크롤링 해온다. 여기서 두 날짜의 선후관계는 상관없도록 프로그램을 작성하였다. 이 말은, 첫번째 인풋으로 2022년 5월 15일을 넣어주고 두번째 인풋으로 2022년 6월 3일을 준 것이나, 첫번째 인풋으로 2022년 6월 3일을 넣어주고 두번째 인풋으로 2022년 5월 15일을 준 것이 똑같이 처리되도록 프로그램을 작성하였다는 것을 의미한다. 이러한 날짜 처리 과정 이후, 해당 날짜 사이의 날씨정보를 불러와야 하는데, 기상청 날씨누리는 달을 기준으로 날씨 정보들을 저장하고 있기 때문에 2022년 5월 15일부터 2022년 6월 3일 사이의 날씨를 크롤맇아기 위해서는 먼저 2022년 5월 정보가 저장되어 있는 사이트에 접근해야 한다. 여기서 날씨 정보를 크롤링해오게 되면, 5월 1일부터 5월 30일 웹사이트에서 나타나는 형식은 똑같기 때문에 5월 한달 전체의 날씨 정보를 가져오게 된다. 이 중, 5월 15일부터의 정보가 필요한 것이므로 크롤링한 정보가 저장된 리스트에서 15번째 정보(14번째 인덱스)부터 리스트에 새롭게 추가해주어야 한다. 그리고, 5월 정보를 모두 받아왔다면 6월달로 넘어가서 같은 방식으로 6월 3일까지의 정보만을 뽑아온다. 그렇게 얻어온 정보들은 리스트형태로 저장이 되어있고, 이를 이용해 그래프로 나타내어 날씨 정보를 시각화했다. [사진 10]의 예시입력에서는 주어진 구간사이에서는 2022년 5월 26일에 20.9도로 가장 높은 온도를 기록했다.

<7번 매서드 : 구간 최저기온 출력>

활용단원 : OOP, 정규표현식, numpy, matplotlib, 실행시간체크, 예외처리

텍스트이(가) 표시된 사진

자동 생성된 설명

[사진 12]

텍스트이(가) 표시된 사진

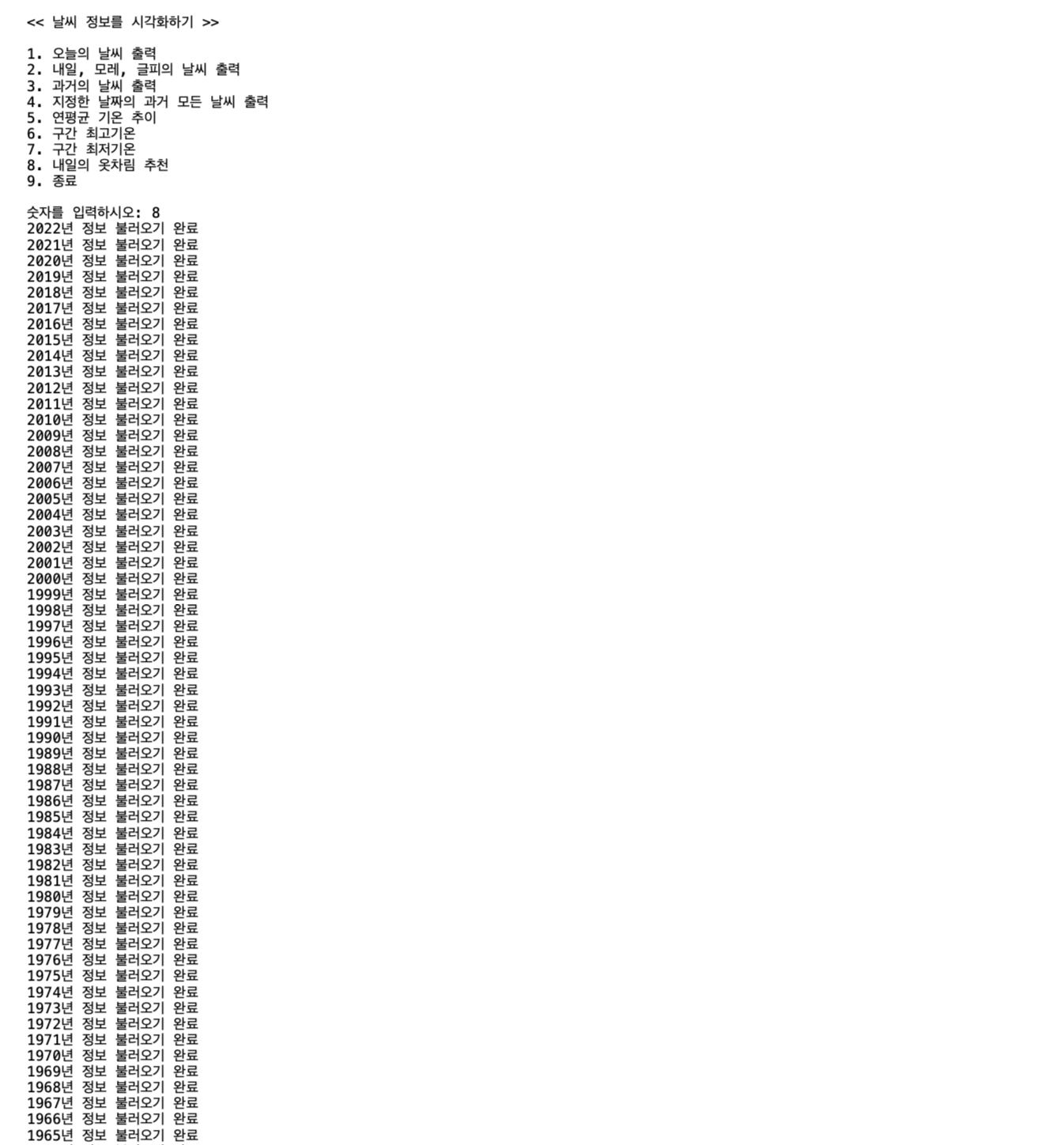
자동 생성된 설명

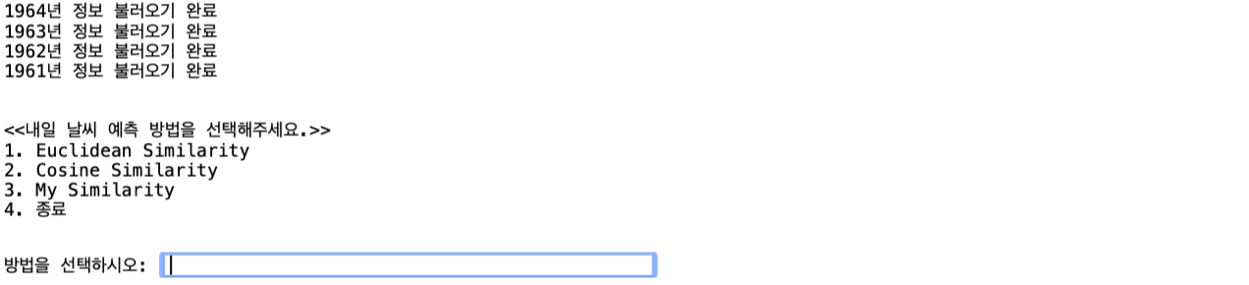
[사진 13]

일곱 번째 매서드는 두개의 날짜를 입력하면 두 날짜 사이의 평균기온을 모두 불러와서 분석한 후, 평균기온이 가장 낮은 날짜를 찾고, 그날의 평균기온이 어땠는지 출력해준다. 또한, 입력한 날짜 사이의 평균기온을 그래프로 나타내주어 한눈에 기온의 변화가 어떻게 진행되는지 알 수 있도록 하였다. 초기화면에서 7을 누르면 실행된다. 만약, [사진 13]과 같이 날짜형식이 아닌 것이 입력되거나, 유효하지 않은 날짜가 입력되면 “날짜 오류. 두 날짜 모두 날짜 범위(1960년 1월 1일 ~ 어제)만 입력이 가능합니다.” 라는 에러 메세지를 출력해주고 함수를 종료한다. 여기서 또한 여섯 번째 매서드에서와 같이 입력이 제대로 주어진다면, 날씨 정보 처리를 해주고, 그 이후 입력한 두 날짜 사이의 모든 날씨 정보를 모두 불러와서 데이터를 처리한다. 데이터를 불러오는 과정은 현재 입력 날짜의 달 정보를 보여주는 URL에 접근한 이후, 해당 달의 모든 날씨 정보를 불러온다. 그렇게 되면 우리가 원하는 날짜가 아닌 날짜의 날씨정보 또한 얻어진다. 이를 제외시켜주기 위해 반복문을 돌며 입력날짜와 탐색날짜를 확인한다. 그 이후, 모든 탐색이 완료되었다면, 최저기온과 최저 기온을 기록한 날짜를 출력해주고, 구간의 모든 날씨정보들을 그래프로 그려준다. [사진 12]의 예시입력에서는 주어진 구간사이에서는 2022년 5월 15일에 13.4도로 가장 낮은 온도를 기록했다.

<8번 매서드 : 내일의 예상 기온에 따른 옷차림 추천>

활용단원 : OOP, 정규표현식, numpy, matplotlib, 실행시간체크, 예외처리





[사진 14]

여덟번째 매서드는 먼저, 매년 오늘 날짜를 기준으로 HOW\_PAST일전 날짜 까지의 평균기온 정보를 리스트에 각각 저장한다. 이렇게 되면 1961년부터 2022년의 정보를 모두 저장하는 것이므로 크기가 HOW\_PAST인 리스트가 82개 생성될 것이다. 하나는 올해의 날씨 정보이고 나머지 81개는 과거의 날씨 정보이다. 81개의 과거 데이터를 올해의 날씨 데이터와 비교하며 가장 유사한 세 개 년도의 자료를 선택해, 선택된 년도의 내일 날짜 날씨들을 유사도에 따라 가중평균을 하여 내일의 날씨를 예측한다. 그리고 그렇게 예측한 날씨에 어울리는 옷차림을 추천해주는 프로그램이다. 현재 프로그램에서는 현재 날짜로부터 6개월정도 전까지의 날씨 정보를 저장해 분석하고 있으므로 HOW\_PAST는 180일로 설정하였다. 날씨는 카오스 이론을 따르므로 미세한 차이로도 예측이 크게 빗나갈 수 있지만, 지수적인 관계가 성립하기 때문에, 과거의 날씨 정보 데이터가 올해의 것과 유사할수록 내일날씨 예측이 더 정확할 수 있다는 것에 정당성이 생기고, 이러한 이유때문에 유사도 측정을 활용한 날씨 예측 알고리즘을 생각해냈다. 초기화면에서 8을 누르면 실행이 되며, 기상청에서 제공하는 모드 년도의 현재 날짜 기준 HOW\_PAST일 전까지의 평균기온을 URL에 접근하여 BeuatifulSoup 라이브러리를 이용해 크롤링한다. 이렇게 해서 올해 날짜 데이터는 temps에, 과거 데이터는 PAST\_TEMPS\_DICT 라는 딕셔너리에 저장하였다. 인덱스값으로 년도를 넣어주고 싶었기 때문에 리스트 자료구조대신 딕셔너리를 활용해야 겠다는 생각을 했다. 이후, 모든 날짜를 불러왔다면, 날씨를 예측할 세가지 방법 중 하나를 선택하라는 인풋창이 다시 등장한다. 유사도 측정을 하는 방식을 선택하라는 것을 의미한다. 여기서 선택지는 Euclidean Similarity, Cosine Similarity My Similarity 가 있다. 위의 [사진 14]는 옷차림 추천 프로그램을 실행하면 나오는 화면이다.

먼저, Euclidean Similarity는 올해 자료와 과거자료와의 각 날짜의 평균기온의 차이의 합을 유사도 측정에 활용하였다. 각 날짜의 날씨가 비슷하다면 차이가 크게 나지 않을 거고, 유사도 값에는 작은 값이 더해지는 효과가 날 것이다. 이렇게 측정한다면 결과 값이 작을수록 유사도가 커지는 것이고, 두자료가 유사하다는 것을 의미한다. 이를 통해 가장 유사한 세 개의 년도를 알 수 있게 되었고, 그리고, 선택한 3개의 년도의 내일 날짜에 해당하는 날씨들을 불러온 후, 그들을 유사도값을 반영하여 가중평균을 내어 내일 날씨를 예측한다. 아래 [사진 15]는 Euclidean Similarity를 이용해 내일 날씨를 예측한 후 옷차림을 추천하준 화면이다.



[사진 15]

다음으로는, Cosine Similarity이다. 이는 Euclidean Similarity를 조금 더 개선시킨 것이다. 갑작스런 태풍이 온다거나, 자연재해로 인해 평균기온이 다른 값들과 현저히 멀리 떨어지는 경우가 있는데, 이런 경우는 특이한 케이스이므로 유사도를 선형적인 차이로 계산하는 Euclidean Similairty의 경우 정확하지 않을 수 있다. 따라서, Cosine Similarity는 유사도를 두 벡터간의 각 차이로 두고, 이 각이 작을 수록 더 유사도가 크다고 해석하여서 날씨를 예측한다. 올해 날씨 데이터나 과거 날씨 데이터는 모두 일차원 리스트이다. 즉, 넘파이의 선형대수 계산툴을 이용하면 각도를 구할 수 있다. 두 날씨 자료를 각각 nowTemps, pastTemps라고 한다면, 이 두 자료를 벡터로 보고 각 차이를 계산할 수 있다. 그 때 이용할 식은 아래와 같다.

벡터의 크기와 벡터의 내적값을 이용하면 두 벡터 사이의 각을 구할 수 있다. 이 두가지 값들은 넘파이 라이브러리를 이용하면 쉽게 구할 수 있다. 이 각도가 작으면 작을수록 두 자료가 유사하다는 것을 의미한다. 이를 통해 가장 유사한 세 개의 년도를 알 수 있게 되었고, 그리고, 선택한 3개의 년도의 내일 날짜에 해당하는 날씨들을 불러온 후, 그들을 유사도값을 반영하여 가중평균을 내어 내일 날씨를 예측한다. 아래 [사진 16]는 Euclidean Similarity를 이용해 내일 날씨를 예측한 후 옷차림을 추천하준 화면이다.



[사진 16]

마지막 방법은, 이번 프로젝트를 진행하며 따로 생각해본 유사도 측정 방법이다. Cosine Similarity와 마찬가지로 Euclidean Similarity를 개선하기 위한 방법이다. 날씨의 특성을 생각해보면 카오스 이론을 따르기 때문에 미세한 자료의 차이가 나중 날씨 예측을 크게 빗나가게 할 수 있다. 이는, 날씨는 차이에 아주 민감하게 반응한다는 것을 의미하고, 차이를 더 부각시킨다면 유사도를 측정하여 날씨를 예측하는데 더 정확할 것이라는 생각이 들었다. 따라서, 각 날짜의 날씨들의 단순 차이가 아닌, 차이의 제곱을 생각해보면 날씨 예측을 더 잘 할 수 있을 것이고 이를 프로그램 상에 반영하여 구현하였다. 아래 [사진 17]는 Euclidean Similarity를 이용해 내일 날씨를 예측한 후 옷차림을 추천하준 화면이다.



[사진 17]