파이썬 라이브러리

수업 목표

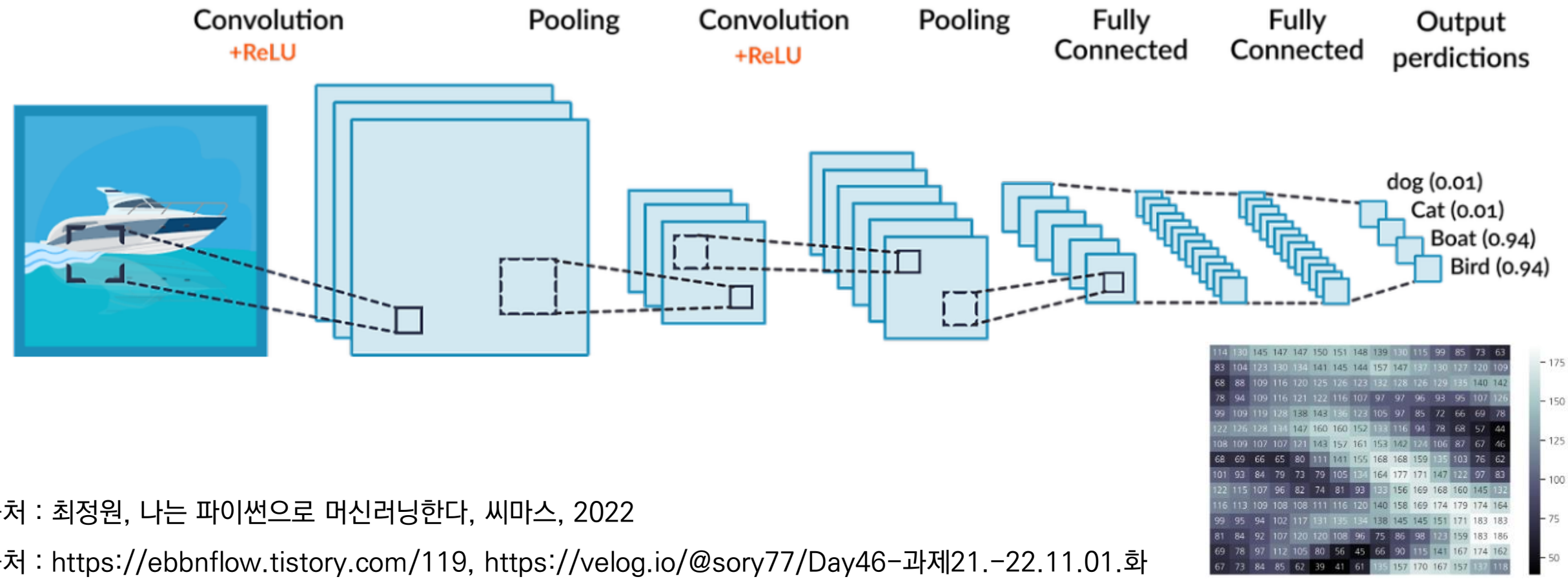
- 배열 연산을 위한 Numpy의 명령어를 설명할 수 있다.
- 데이터 분석을 위한 Pandas의 명령어를 설명할 수 있다.
- 데이터 시각화를 위한 Matplotlib의 명령어를 설명할 수 있다.

1. Numpy

1.1 Numpy란?

❖ Numpy

- 머신러닝에서 이미지, 영상, 텍스트, 소리 등의 모든 데이터는 실수 형태(수치 데이터)의 배열로 표현
- Numpy눈 수치 데이터의 배열을 구성하고, 배열을 연산 처리하는데 필요한 라이브러리



출처 : 최정원, 나는 파이썬으로 머신러닝한다, 씨마스, 2022

출처 : https://ebbnflow.tistory.com/119, https://velog.io/@sory77/Day46-과제21.-22.11.01.화

1.2 Numpy 사용하기

❖ Numpy에서 사용하는 데이터 구조 : 개수/형태에 따른 네 가지 유형

- 스칼라(Scalar) : 한 개의 숫자로 이루어진 데이터
  - 예) 24

- 벡터(Vector) : 스칼라(숫자)의 집합. 여러 개의 숫자가 특정한 순서대로 모여 있는 1차원 배열
  - 예) [1,2,3] → 행 벡터 $\begin{bmatrix}1\\2\\3\end{bmatrix}$ → 열 벡터

- 행렬(Matrix) : 벡터의 집합. 1개 이상의 행과 열로 구성된 2차원 배열
  - 예) $\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}$ → 2개의 행, 3개의 열

- 텐서(Tensor) : 행렬의 집합. 3차원 이상의 배열
  - 예) $\begin{bmatrix}\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} & \begin{bmatrix}5 & 6\\7 & 8\end{bmatrix}\\ \begin{bmatrix}9 & 0\\1 & 2\end{bmatrix} & \begin{bmatrix}3 & 4\\5 & 6\end{bmatrix}\end{bmatrix}$

출처 : 최정원, 나는 파이썬으로 머신러닝한다, 씨마스, 2022

1.2 Numpy 사용하기


```python
#Numpy 라이브러리를 불러오고, np라는 별명으로 사용
import numpy as np

#행 벡터 생성
row = np.array([1,2,3])

#열 벡터 생성
col_1 = np.array([[1],
                  [2],
                  [3]])

col_2 = np.array([[1,2,3]])

#행렬 생성
matrix_1 = np.array([[1,2,3],
                     [4,5,6],
                     [7,8,9]])

matrix_2 = np.array([[1,2,3],[4,5,6],[7,8,9]])
```

```python
#출력
print("행 벡터")
print(row)

print("₩n열 벡터1")
print(col_1)

print("₩n열 벡터2")
print(col_2)

print("₩n행렬1")
print(matrix_1)

print("₩n행렬2")
print(matrix_2)
```

```
행 벡터
[1 2 3]

열 벡터1
[[1]
 [2]
 [3]]

열 벡터2
[[1 2 3]]

행렬1
[[1 2 3]
 [4 5 6]
 [7 8 9]]

행렬2
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

출처 : 최정원, 나는 파이썬으로 머신러닝한다, 씨마스, 2022

1.2 Numpy 사용하기

❖ 배열 채우기

- arrange( ) : 배열의 크기(범위)를 지정하면, 지정한 범위의 정숫값을 갖는 배열 생성
- zeros( ) : 배열의 데이터를 0으로 채움
- ones( ) : 배열의 데이터를 1로 채움
- random.rand( ) : 배열의 데이터를 임의의 값으로 채움

```
[8]   1 number = np.arange(1,6)
      2 print(number)
```

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

```
[9]   1 number_zero = np.zeros((2,3))
      2 print(number_zero)
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
```

```
[10]  1 number_one = np.ones((2,3))
      2 print(number_one)
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
```

```
[11]  1 number_rand = np.random.rand(2,3)
      2 print(number_rand)
```

```
[[0.82559081 0.2721674  0.84155651]
 [0.6884881  0.5021458  0.7814772 ]]
```

출처 : 최정원, 나는 파이썬으로 머신러닝한다, 씨마스, 2022

1.2 Numpy 사용하기

❖ 배열의 사칙연산

```
1 a = np.array([[10,20,30],[40,50,60]])
2 b = np.array([[1,2,3],[4,5,6]])
3
4 print("행렬의 덧셈")
5 print(a+b)
6
7 print("₩n행렬의 뺄셈")
8 print(a-b)
9
10 print("₩n행렬의 곱셈")
11 print(a*b)
12
13 print("₩n행렬의 나눗셈")
14 print(a/b)
```

```
행렬의 덧셈
[[11 22 33]
 [44 55 66]]

행렬의 뺄셈
[[ 9 18 27]
 [36 45 54]]

행렬의 곱셈
[[ 10  40  90]
 [160 250 360]]

행렬의 나눗셈
[[10. 10. 10.]
 [10. 10. 10.]]
```

```
행렬 a
[[10 20 30]
 [40 50 60]]

행렬 b
[[1 2 3]
 [4 5 6]]
```

출처 : 최정원, 나는 파이썬으로 머신러닝한다, 씨마스, 2022

1.2 Numpy 사용하기

❖ 행렬 함수

- sum( ) : 합계
- average( ) : 평균
- median( ) : 중앙값
- max( ) : 최댓값
- min( ) : 최솟값
- var( ) : 분산
- std( ) : 표준편차

출처 : 최정원, 나는 파이썬으로 머신러닝한다, 씨마스, 2022

1.2 Numpy 사용하기

❖ sum( ) : 합계

❖ average( ) : 평균

```
1 a = np.array([[10,20,30],[40,50,60]])
2 b = np.array([[1,2,3],[4,5,6]])
3
4 print("행렬a")
5 print(a)
6
7 print("₩n행렬b")
8 print(b)
9
10 #합(sum)
11 print("₩n행렬의 a의 합")
12 print(np.sum(a))
13
14 print("₩n행렬의 b의 합")
15 print(np.sum(b))
16
17 print("₩n행렬 a,b에서 행의 원소 합계")
18 print(np.sum(a, axis=0))
19
20 print("₩n행렬 a,b에서 열의 원소 합계")
21 print(np.sum(a, axis=1))
22
23 #평균(average)
24 print("₩n행렬 a의 평균")
25 print(np.average(a))
26
27 print("₩n행렬 a의 행별 원소 평균")
28 print(np.average(a, axis=0))
29
30 print("₩n행렬 a의 열별 원소 평균")
31 print(np.average(a, axis=1))
```

```
행렬a
[[10 20 30]
 [40 50 60]]

행렬b
[[1 2 3]
 [4 5 6]]

행렬의 a의 합
210

행렬의 b의 합
21

행렬 a,b에서 행의 원소 합계
[50 70 90]

행렬 a,b에서 열의 원소 합계
[ 60 150]

행렬 a의 평균
35.0

행렬 a의 행별 원소 평균
[25. 35. 45.]

행렬 a의 열별 원소 평균
[20. 50.]
```

출처 : 최정원, 나는 파이썬으로 머신러닝한다, 씨마스, 2022

1.2 Numpy 사용하기

❖ 중앙값(median)

```
#중앙값(median)
c = np.array([[100,200,300,400,500],
              [600,700,800,900,1000],
              [1,2,3,4,5],
              [6,7,8,9,0]])

print("행렬c")
print(c)

print("₩n행렬 c의 중앙값")
print(np.median(c))

print("₩n행렬 c의 행별 중앙값")
print(np.median(c, axis=0))

print("₩n행렬 a의 열별 중앙값")
print(np.median(c, axis=1))
```

```
행렬c
[[ 100  200  300  400  500]
 [ 600  700  800  900 1000]
 [   1    2    3    4    5]
 [   6    7    8    9    0]]

행렬 c의 중앙값
54.5

행렬 c의 행별 중앙값
[ 53.  103.5 154.  204.5 252.5]

행렬 a의 열별 중앙값
[300. 800.   3.   7.]
```

출처 : 최정원, 나는 파이썬으로 머신러닝한다, 씨마스, 2022

1.2 Numpy 사용하기

- max( ) : 최댓값
- min( ) : 최솟값

```
#최댓값(max), 최솟값(min)
c = np.array([[100,200,300,400,500],
              [600,700,800,900,1000],
              [1,2,3,4,5],
              [6,7,8,9,0]])

print("행렬c")
print(c)

print("₩n행렬 c의 최댓값, 최솟값")
print(np.max(c), np.min(c))

print("₩n행렬 c의 행별 최댓값")
print(np.max(c, axis=0))

print("₩n행렬 a의 열별 최댓값")
print(np.max(c, axis=1))

print("₩n행렬 c의 최댓값의 인덱스")
print(np.argmax(c))

print("₩n행렬 c의 행별 최댓값 인덱스")
print(np.argmax(c, axis=0))

print("₩n행렬 a의 열별 최댓값 인덱스")
print(np.argmax(c, axis=1))
```

```
행렬c
[[ 100  200  300  400  500]
 [ 600  700  800  900 1000]
 [   1    2    3    4    5]
 [   6    7    8    9    0]]

행렬 c의 최댓값, 최솟값
1000 0

행렬 c의 행별 최댓값
[ 600  700  800  900 1000]

행렬 a의 열별 최댓값
[ 500 1000    5    9]

행렬 c의 최댓값의 인덱스
9

행렬 c의 행별 최댓값
[1 1 1 1 1]

행렬 a의 열별 최댓값
[4 4 4 3]
```

출처 : 최정원, 나는 파이썬으로 머신러닝한다, 씨마스, 2022

1.2 Numpy 사용하기

- var( ) : 분산
- std( ) : 표준편차

```
1 #분산(var)), 표준편차(std)
2 c = np.array([[100,200,300,400,500],
3                [600,700,800,900,1000],
4                [1,2,3,4,5],
5                [6,7,8,9,0]])
6
7 print("행렬c")
8 print(c)
9
10 print("₩n행렬 c의 분산, 표준편차")
11 print(np.var(c), np.std(c))
```

```
행렬c
[[ 100  200  300  400  500]
 [ 600  700  800  900 1000]
 [   1    2    3    4    5]
 [   6    7    8    9    0]]

행렬 c의 분산, 표준편차
115646.6875 340.06865115738026
```

출처 : 최정원, 나는 파이썬으로 머신러닝한다, 씨마스, 2022

1.2 Numpy 사용하기

- any( ) : 특정 조건을 만족하는 원소가 한 개라도 있으면 TRUE
- all( ) : 특정 조건을 모든 원소가 만족하면 TRUE
- where( ) : 특정 조건을 만족하는 경우와 그렇지 않은 경우 다른 결과를 출력

```
1 #any() : 특정 조건을 만족하는 원소가 1개라도 있는가
2 a = np.array([[10,20,30],[40,50,60]])
3 print(np.any(a>40))
4
5 #all() : 특정 조건을 모든 원소가 만족하는가?
6 print(np.all(a>40))
7
8 #where() : 특정 조건을 만족하는 것과 그렇지 않은 것의 다른 결과
9 print(np.where(a>40, a, 0))
```

```
True
False
[[ 0  0  0]
 [ 0 50 60]]
```

출처 : 최정원, 나는 파이썬으로 머신러닝한다, 씨마스, 2022

2. 데이터 시각화(Pandas)

Pandas(판다스)란 데이터를 표 형태의 데이터프레임에 넣어서 쉽게 이용할 수 있게 해주는 파이썬의 공학용 라이브러리. 즉, **파이썬의 엑셀**과 같음

Q) 엑셀로 하면 안될까?

A) 엑셀로도 가능하지만 대용량 처리를 할 때 속도가 아주 느림
똑같은 작업을 반복해야 할 때 Pandas로 코딩을 해 놓으면 생산성이 훨씬 높음
Pandas를 사용하면 머신러닝을 위한 데이터 처리에서 활용할 수 있음

Q) SQL로 하면 안될까?

A) 대부분의 프로젝트에서 보안이나 안전상의 이유로 직접 DBMS에 작업하기 어려움
일반적으로 해당 부서에서 원하는 데이터를 엑셀로 넘겨 주는 경우가 대부분이고, 각종 정보 공개 사이트에서도 엑셀 형태로 제공되는 경우가 많음

<출처> https://funfunfuhaha.tistory.com/18

Pandas 자료구조

pandas는 크게 세가지의 자료구조를 지원.
Series(1차원 자료구조), DataFrame(2차원 자료구조), Panel(3차원 자료구조)

Series

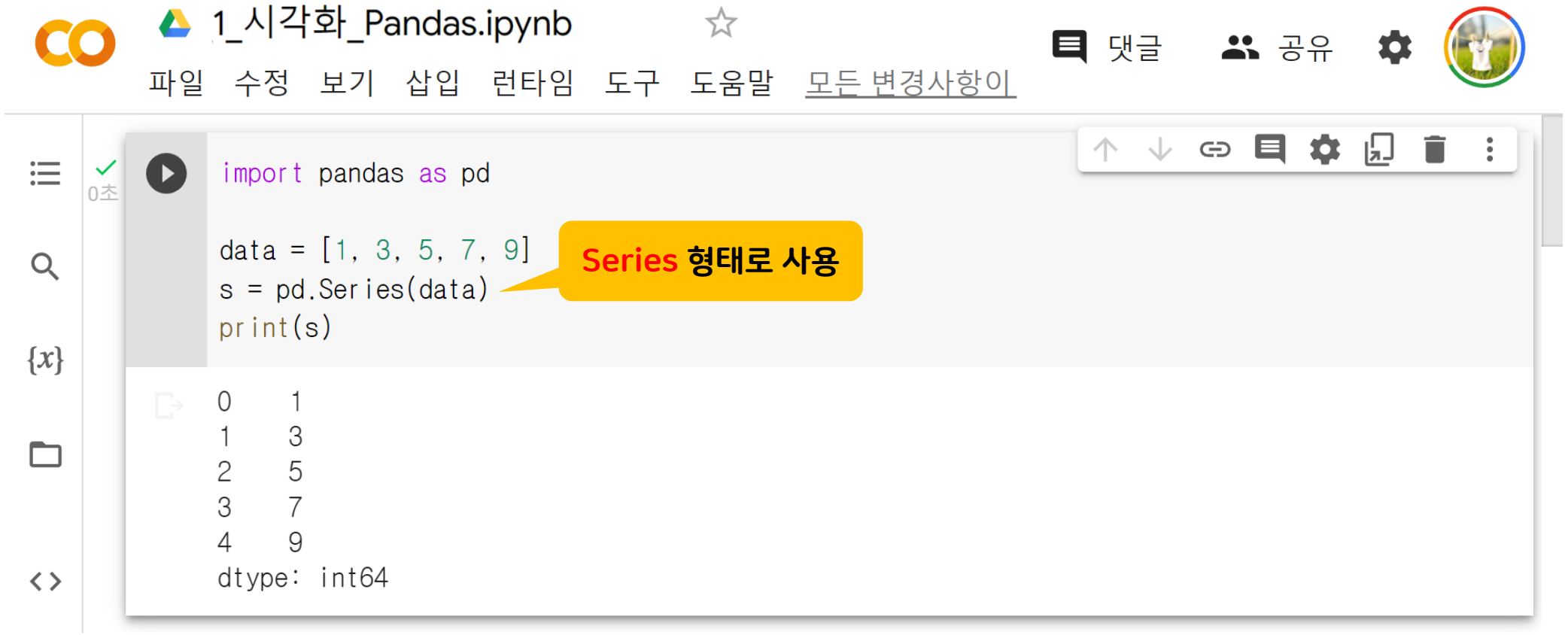
Data Frame

Panel

<출처> http://pythonstudy.xyz/python/article/408-pandas-데이타-분석

Pandas 자료구조(1차원 자료구조 Series)

1차원 자료구조인 Series는 배열/리스트와 같은 일련의 시퀀스 데이터 사용
별도의 인덱스 레이블을 지정하지 않으면 자동으로 0부터 시작되는 디폴트 정수 인덱스 사용



<출처> http://pythonstudy.xyz/python/article/408-pandas-데이타-분석

Pandas 자료구조(2차원 자료구조 DataFrame)

2차원 자료구조인 DataFrame는 행/열이 있는 테이블 데이터(Tabular Data)를 사용

<출처> http://pythonstudy.xyz/python/article/408-pandas-데이타-분석

Pandas 자료구조(2차원 자료구조 DataFrame)

2차원 자료구조인 DataFrame는 행/열이 있는 테이블 데이터(Tabular Data)를 사용

```
[6]  print(df['year']) #년도만 출력
```

```
0    2016
1    2017
2    2018
Name: year, dtype: int64
```

```
[7]  print(df.year)   #이렇게 사용도 가능
```

```
0    2016
1    2017
2    2018
Name: year, dtype: int64
```

```
[9]  sum = df['GDP rate'].sum()   #GDP rate의 합
     avg = df['GDP rate'].mean()  #GDP rate의 평균
     print(sum, avg)
```

```
8.9 2.966666666666667
```

```
[11] df[df.year>2016] #년도가 2016보다 큰 것만 출력
```

	year	GDP rate	GDP
1	2017	3.1	1.73M
2	2018	3.0	1.83M

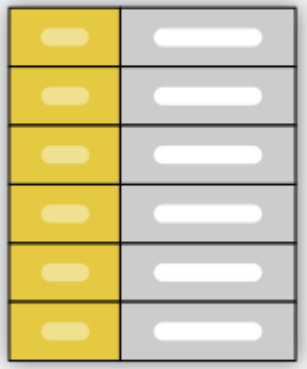
```
[10] df.describe() #기본 통계치를 모두 표시
```

	year	GDP rate
count	3.0	3.000000
mean	2017.0	2.966667
std	1.0	0.152753
min	2016.0	2.800000
25%	2016.5	2.900000
50%	2017.0	3.000000
75%	2017.5	3.050000
max	2018.0	3.100000

<출처> http://pythonstudy.xyz/python/article/408-pandas-데이타-분석

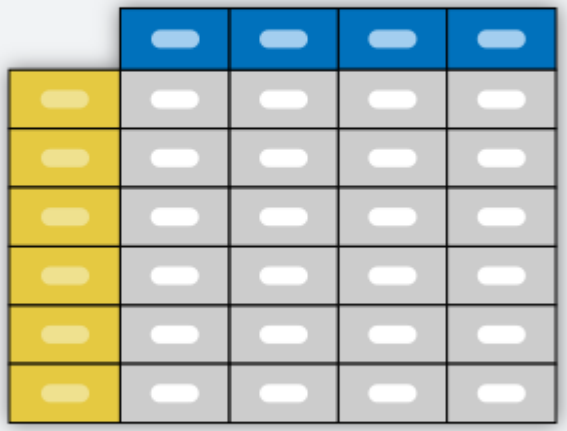
Pandas 에서 사용되는 대표적인 데이터 오브젝트

시리즈 (Series)



Series 는 1차원 배열의 형태를 갖는다.
인덱스(노란색)라는 한 가지 기준에
의하여 데이터가 저장된다.

데이터프레임 (DataFrame)



DataFrame 은 2차원 배열의 형태를 갖는다.
인덱스(노란색)와 컬럼(파란색)이라는 두 가지
기준에 의하여 표 형태처럼 데이터가 저장된다.

컬럼명을 이용하여 데이터 선택하기 (컬럼 선택)

df.A 또는 df['A']

	A	B	C	D
2013-01-01	0.469112	-0.282863	-1.509059	-1.135632
2013-01-02	1.212112	-0.173215	0.119209	-1.044236
2013-01-03	-0.861849	-2.104569	-0.494929	1.071804
2013-01-04	0.721555	-0.706771	-1.039575	0.271860
2013-01-05	-0.424972	0.567020	0.276232	-1.087401
2013-01-06	-0.673690	0.113648	-1.478427	0.524988

<출처> https://dandyrilla.github.io/2017-08-12/pandas-10min/

2.1 선 그래프(Line Graph)

Pandas – 선 그래프

기상청 날씨누리

날씨
바다
영상·일기도
태풍
지진·화산
테마날씨
황사
관측·기후

육상

도시별관측
지역별상세관측
과거관측

요소별자료

관측·기후 > 육상 > 과거관측 > 요소별자료

지점 서울(유) 선택 년도 2021년 선택 요소 평균기온 선택

[일평균기온(℃)] 108 서울 / 2021년

일자	1월	2월	3월	4월	5월	6월	7월	8월	9월	10월	11월	12월
1일	-4.2	5.0	4.7	17.7	10.2	20.2	26.3	27.1	21.4	21.1	12.9	-1.3
2일	-5.0	-5.6	2.3	17.8	12.4	23.2	27.1	26.5	23.5	20.7	11.5	1.3
3일	-5.6	-3.2	4.4	14.9	14.1	19.1	22.6	28.0	24.3	22.3	12.1	1.9
30일	-1.6		10.8	11.3	18.4	24.3	30.5	24.2	20.3	13.8	5.7	-3.9
31일	3.2		14.5		19.9		29.8	20.5		14.1		-6.7
평균	-2.4	2.7	9.0	14.2	17.1	22.8	28.1	25.9	22.6	15.6	8.2	0.6

<출처> https://www.weather.go.kr/w/obs-climate/land/past-obs/obs-by-element.do?stn=108&yy=2021&obs=07

Pandas – 선 그래프

일자	1월	2월	3월	4월	5월	6월	7월	8월	9월	10월	11월	12월
평균	-2.4	2.7	9.0	14.2	17.1	22.8	28.1	25.9	22.6	15.6	8.2	0.6

2_시각화_그래프.ipynb

```
[1]  import pandas as pd
     import matplotlib.pyplot as plt

     df = pd.DataFrame()
     df['month'] = ['2021-'+str(x).zfill(2) for x in range(1,13)] ## 월(month)
     df['temperature'] = [-2.4, 2.7, 9, 14.2, 17.1, 22.8, 28.1, 25.9, 22.6, 15.6, 8.2, 0.6] ## 평균기온
```

<출처> https://www.weather.go.kr/w/obs-climate/land/past-obs/obs-by-element.do?stn=108&yy=2021&obs=07

Pandas – 선 그래프

기상청 날씨누리

날씨

바다

영상·일기도

지점 서울(유) 선택 년도 2021년 선택 요소 평균기온 선택

일자	1월	2월	3월	4월	5월	6월	7월	8월	9월	10월	11월	12월
평균	-2.4	2.7	9.0	14.2	17.1	22.8	28.1	25.9	22.6	15.6	8.2	0.6

2_시각화_그래프.ipynb

```
[1]  import pandas as pd
     import matplotlib.pyplot as plt

     df = pd.DataFrame()
     df['month'] = ['2021-'+str(x).zfill(2) for x in range(1,13)] ## 월(month)
     df['temperature'] = [-2.4, 2.7, 9, 14.2, 17.1, 22.8, 28.1, 25.9, 22.6, 15.6, 8.2, 0.6] ## 평균기온
```

Matplotlib 라이브러리를 사용하여
그래프를 그릴 수 있음

<출처> https://www.weather.go.kr/w/obs-climate/land/past-obs/obs-by-element.do?stn=108&yy=2021&obs=07

Pandas – 선 그래프

일자	1월	2월	3월	4월	5월	6월	7월	8월	9월	10월	11월	12월
평균	-2.4	2.7	9.0	14.2	17.1	22.8	28.1	25.9	22.6	15.6	8.2	0.6

2_시각화_그래프.ipynb

```
[1]  import pandas as pd
     import matplotlib.pyplot as plt

     df = pd.DataFrame()
     df['month'] = ['2024-'+str(x).zfill(2) for x in range(1,13)] ## 월(month)
     df['temperature'] = [-2.4, 2.7, 9, 14.2, 17.1, 22.8, 28.1, 25.9, 22.6, 15.6, 8.2, 0.6] ## 평균기온
```

파이썬 문자열 앞 0으로 채우기 zfill()

<출처> https://zetawiki.com/wiki/파이썬_문자열_앞_0으로_채우기_zfill()

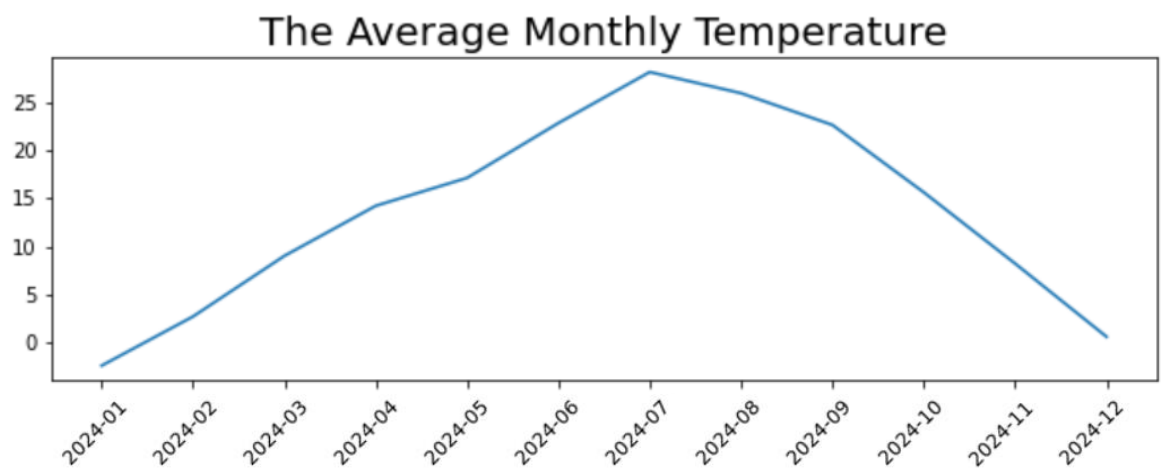
Pandas – 선 그래프

2_시각화_그래프.ipynb

댓글 공유

```
fig = plt.figure(figsize=(10,3)) ## 캔버스 생성
fig.set_facecolor('white') ## 캔버스 색상 설정
ax = fig.add_subplot() ## 그림 뼈대(프레임) 생성

ax.plot(df['month'],df['temperature']) ## 선그래프 생성

plt.xticks(rotation=45) ## x축 눈금 라벨 설정 - 40도 회전
plt.title('The Average Monthly Temperature',fontsize=20) ## 제목
plt.show()
```

ax.plot(x축에 들어갈 데이터, y축에 들어갈 데이터)



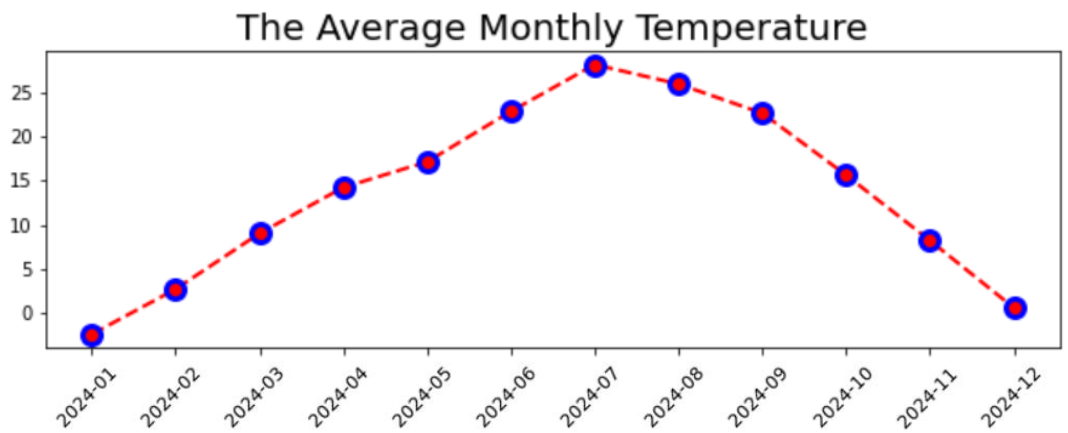
<출처> https://www.weather.go.kr/w/obs-climate/land/past-obs/obs-by-element.do?stn=108&yy=2021&obs=07

Pandas – 선 그래프

2_시각화_그래프.ipynb

댓글 공유

```
fig = plt.figure(figsize=(10,3)) ## 캔버스 생성
fig.set_facecolor('white') ## 캔버스 색상 설정
ax = fig.add_subplot() ## 그림 뼈대(프레임) 생성

## 선그래프 꾸미기
#ax.plot(df['month'],df['temperature'])
ax.plot(df['month'],df['temperature'], color='#ff0000',
        linestyle='--',linewidth=2,
        marker='o',markersize=10,
        markeredgecolor='blue', markeredgewidth=3,)

plt.xticks(rotation=45) ## x축 눈금 라벨 설정 - 40도 회전
plt.title('The Average Monthly Temperature',fontsize=20) ## 제목
plt.show()
```

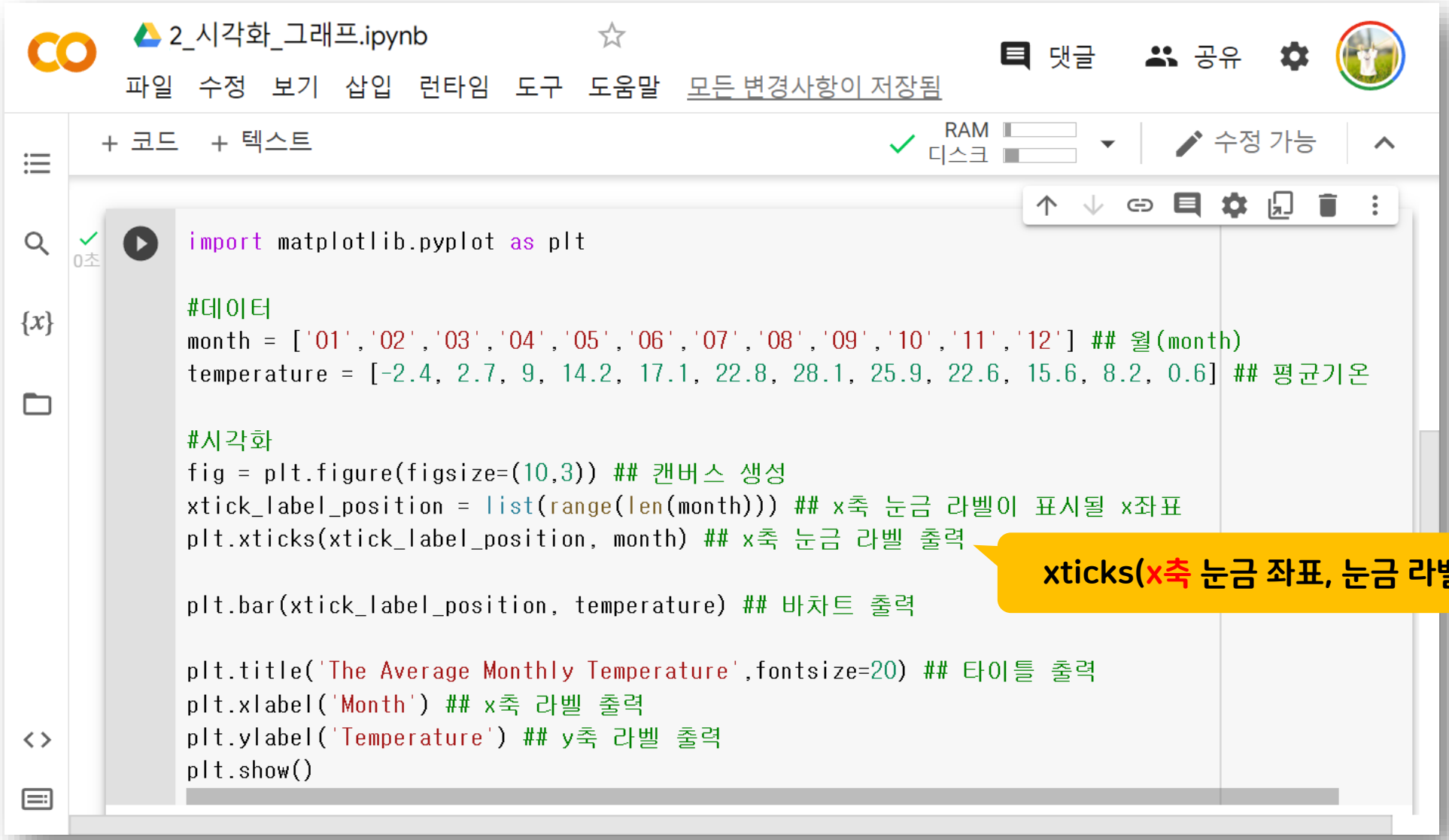



```
ax.plot(df['month'],df['temperature'], color='#ff0000',
        linestyle='--',linewidth=2,
        marker='o',markersize=10,
        markeredgecolor='blue', markeredgewidth=3,)
```

<출처> https://zephyrus1111.tistory.com/21

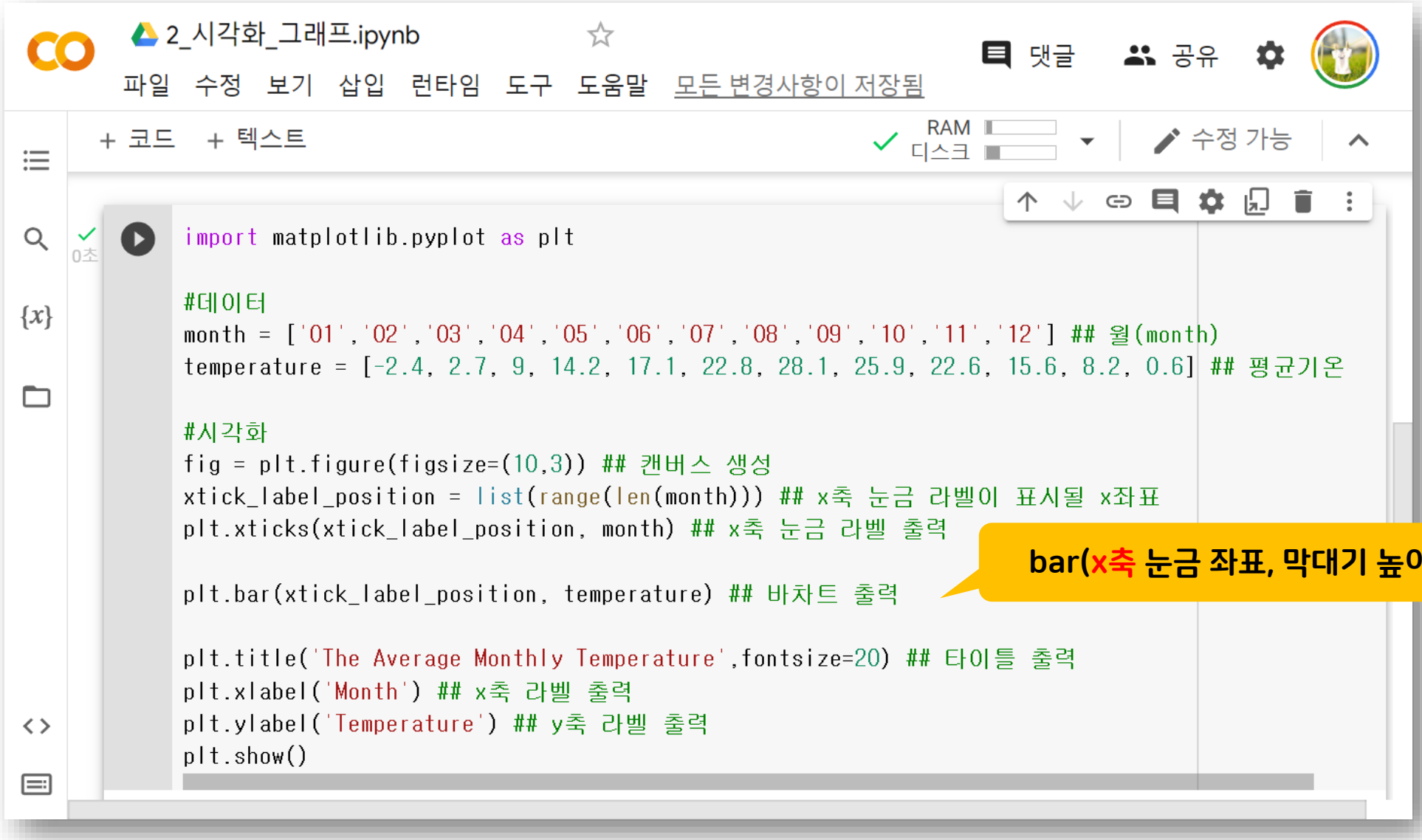
2.2 막대 그래프(Bar Chart)

Pandas – 막대 그래프



<출처> https://zephyrus1111.tistory.com/8

Pandas – 막대 그래프



<출처> https://zephyrus1111.tistory.com/8

Pandas – 막대 그래프

2_시각화_그래프.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

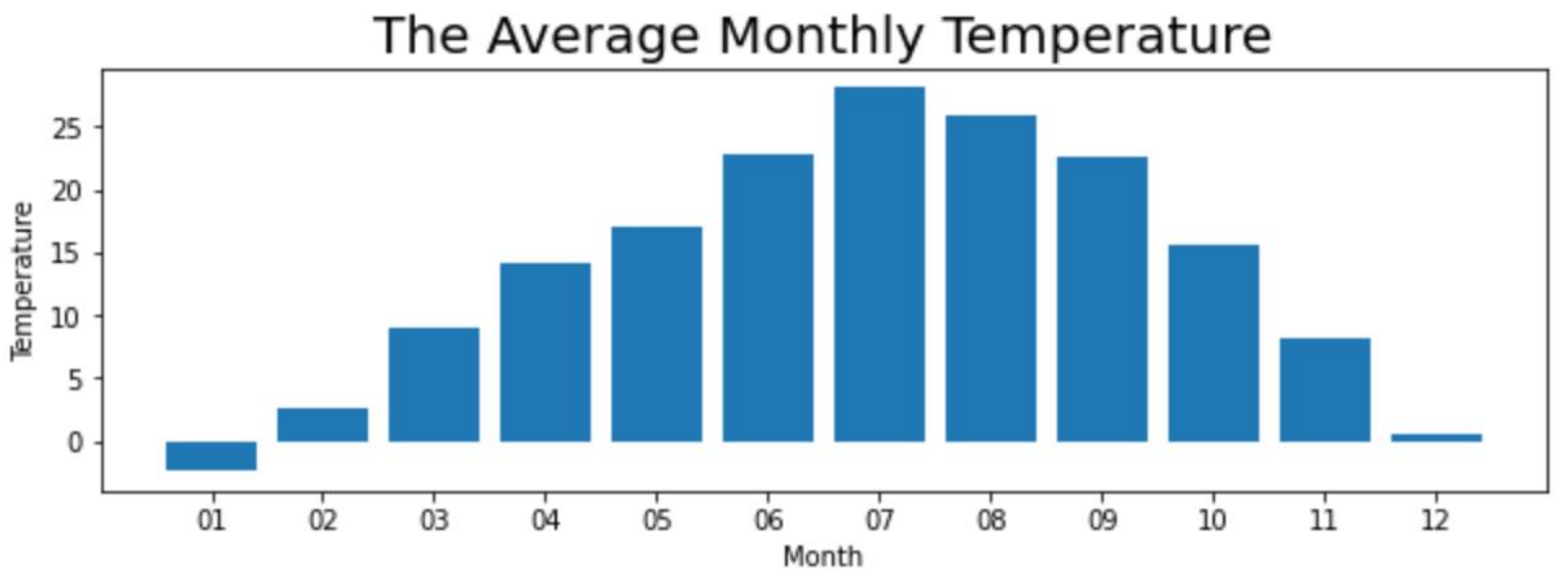
댓글 공유

+ 코드 + 텍스트

RAM 디스크

수정 가능

```
plt.title('The Average Monthly Temperature',fontsize=20) ## 타이틀 출력
plt.xlabel('Month') ## x축 라벨 출력
plt.ylabel('Temperature') ## y축 라벨 출력
plt.show()
```



<출처> https://zephyrus1111.tistory.com/8

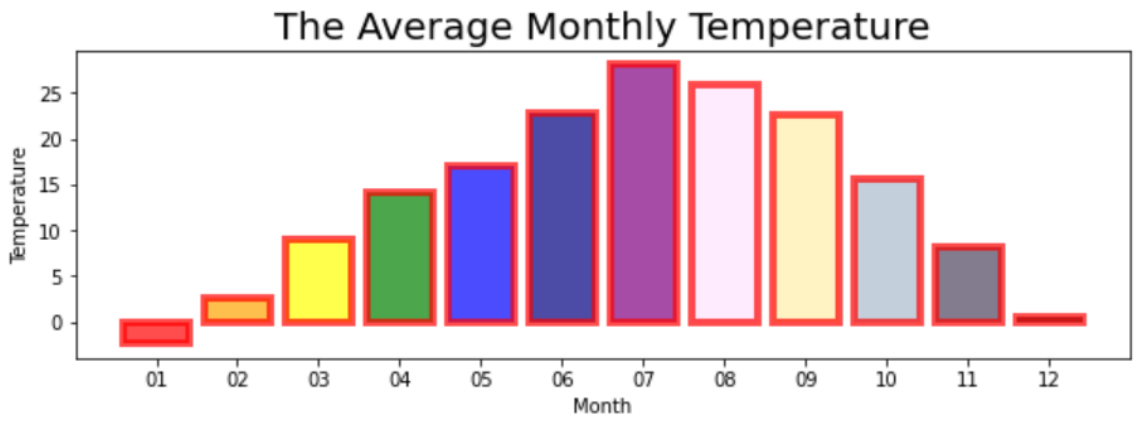
Pandas – 막대 그래프

2_시각화_그래프.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

```
xtick_label_position = list(range(len(month))) ## x축 눈금 라벨이 표시될 x좌표
plt.xticks(xtick_label_position, month) ## x축 눈금 라벨 출력

## 바차트 꾸미기
#plt.bar(xtick_label_position, temperature, color='#ff0000')
colors = ['red','orange','yellow','green','blue','navy','purple','#fee4fe','#ffeeaa','#aabbcc','#4e445e','#000000']
plt.bar(xtick_label_position, temperature, color=colors, edgecolor='red', alpha=0.7, linewidth=4)

plt.title('The Average Monthly Temperature',fontsize=20) ## 타이틀 출력
plt.xlabel('Month') ## x축 라벨 출력
plt.ylabel('Temperature') ## y축 라벨 출력
plt.show()
```



<출처> https://zephyrus1111.tistory.com/9

Pandas – 막대 그래프

2_시각화_그래프.ipynb

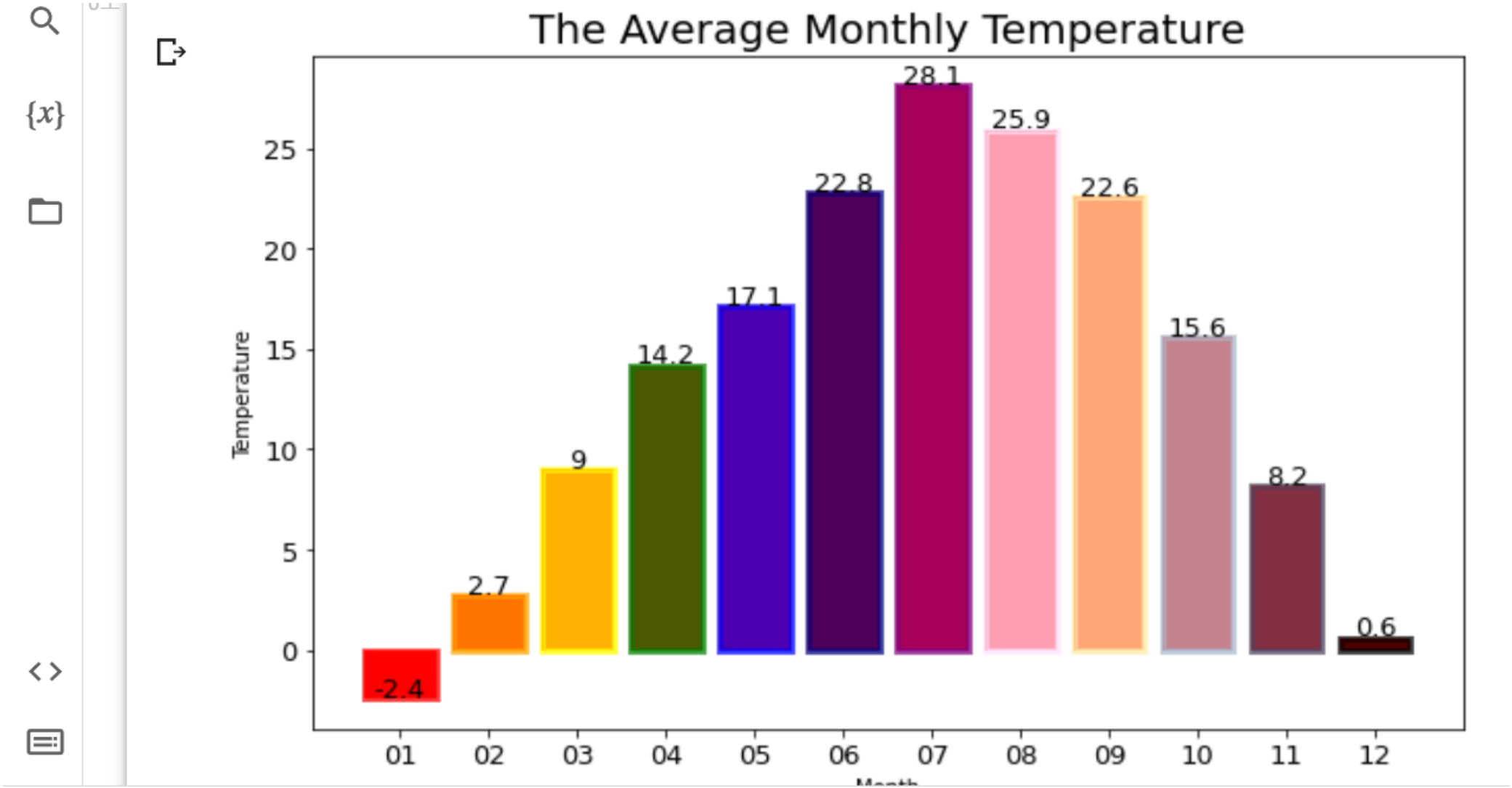
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

```
#글자 넣기
tick_size = 13 ## 눈금 폰트 사이즈
axis_label_size = 15 ## x축, y축 폰트 사이즈
ax = fig.add_subplot() ## Axes 추가
xtick_label_position = list(range(len(month))) ## x축 눈금 라벨이 표시될 x좌표
plt.yticks(fontsize=tick_size)
plt.xticks(xtick_label_position, month, fontsize=tick_size) ## x축 눈금 라벨 출력

bars = plt.bar(xtick_label_position, temperature, color=colors, ₩
               edgecolor=colors, alpha=0.7, linewidth=3) ## 바차트 출력

for i, b in enumerate(bars):
    ax.text(b.get_x()+b.get_width()*(1/2),b.get_height()+0.1, ₩
            temperature[i],ha='center',fontsize=13)


plt.title('The Average Monthly Temperature',fontsize=20) ## 타이틀 출력
plt.xlabel('Month') ## x축 라벨 출력
plt.ylabel('Temperature') ## y축 라벨 출력
plt.show()
```

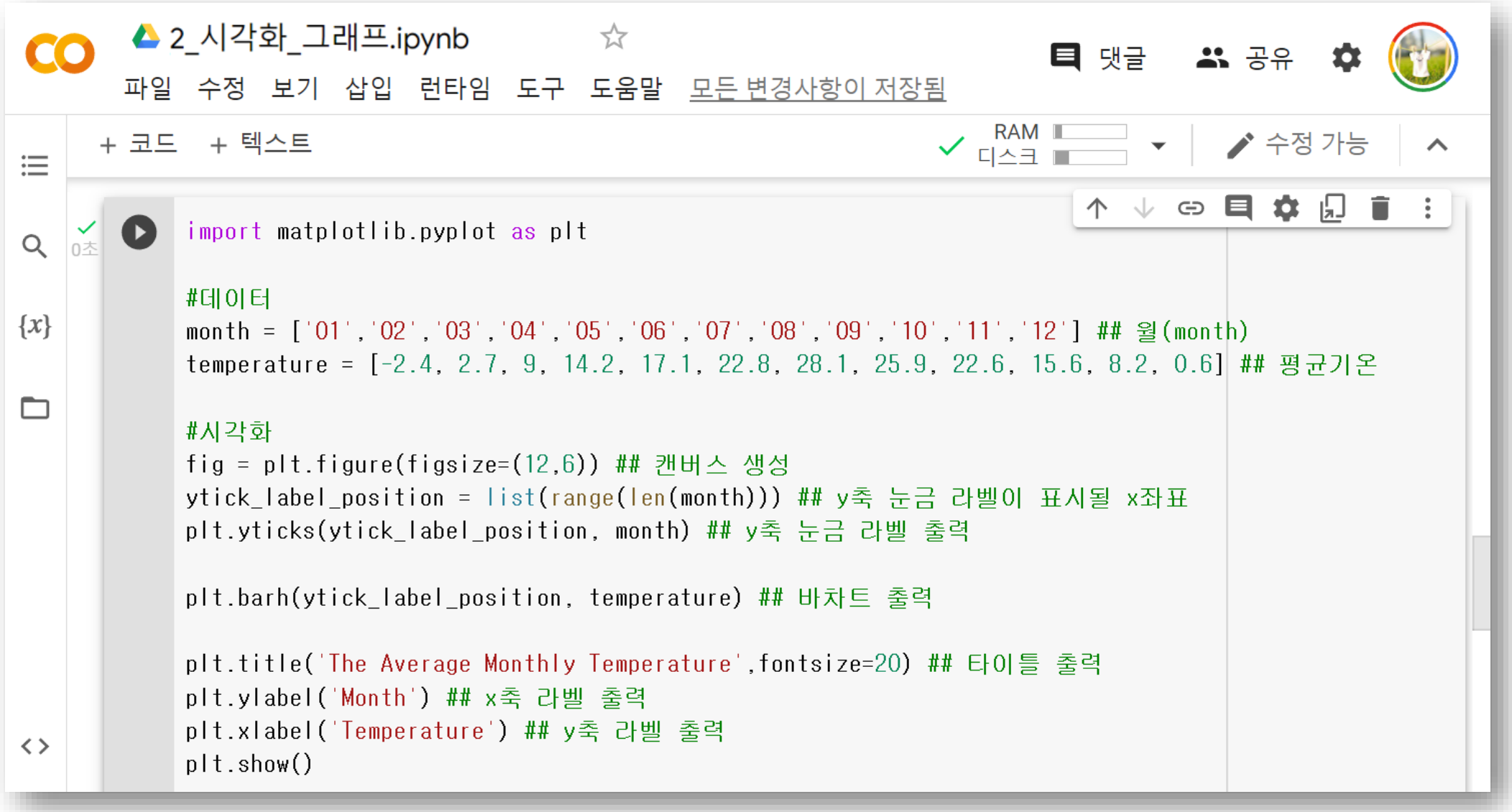
<출처> https://zephyrus1111.tistory.com/9

Pandas – 막대 그래프



<출처> https://zephyrus1111.tistory.com/9

Pandas – 막대 그래프(수평바)



<출처> https://zephyrus1111.tistory.com/8

Pandas – 막대 그래프

2_시각화_그래프.ipynb

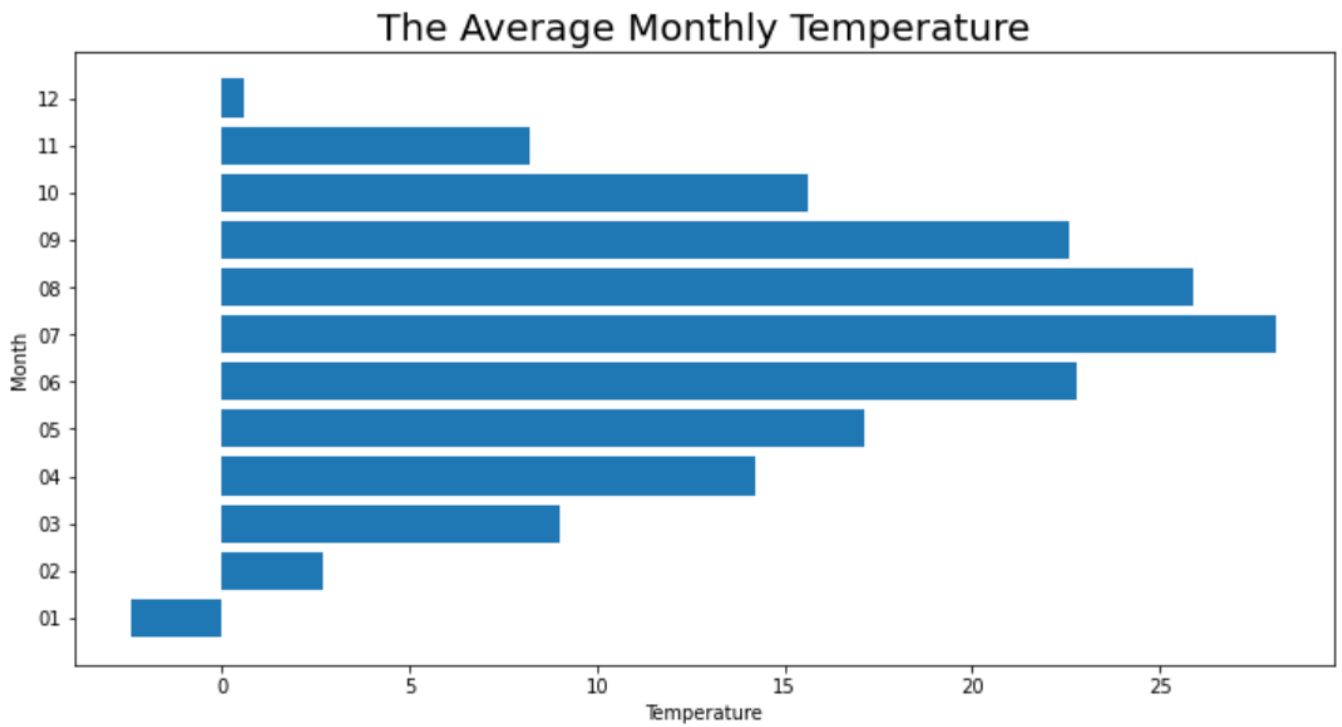
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유

+ 코드 + 텍스트

RAM 디스크 수정 가능

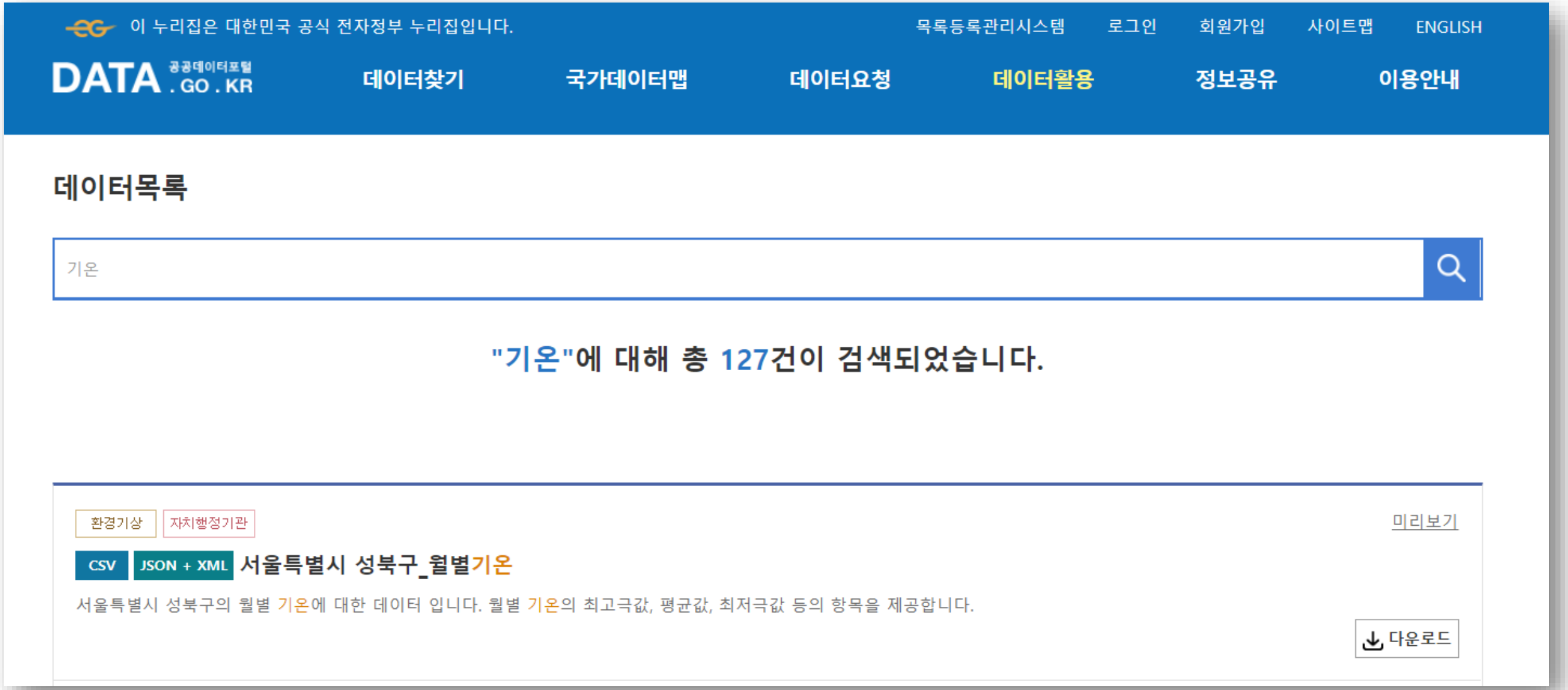
```
plt.ylabel('Month') ## x축 라벨 출력
plt.xlabel('Temperature') ## y축 라벨 출력
plt.show()
```



<출처> https://zephyrus1111.tistory.com/8

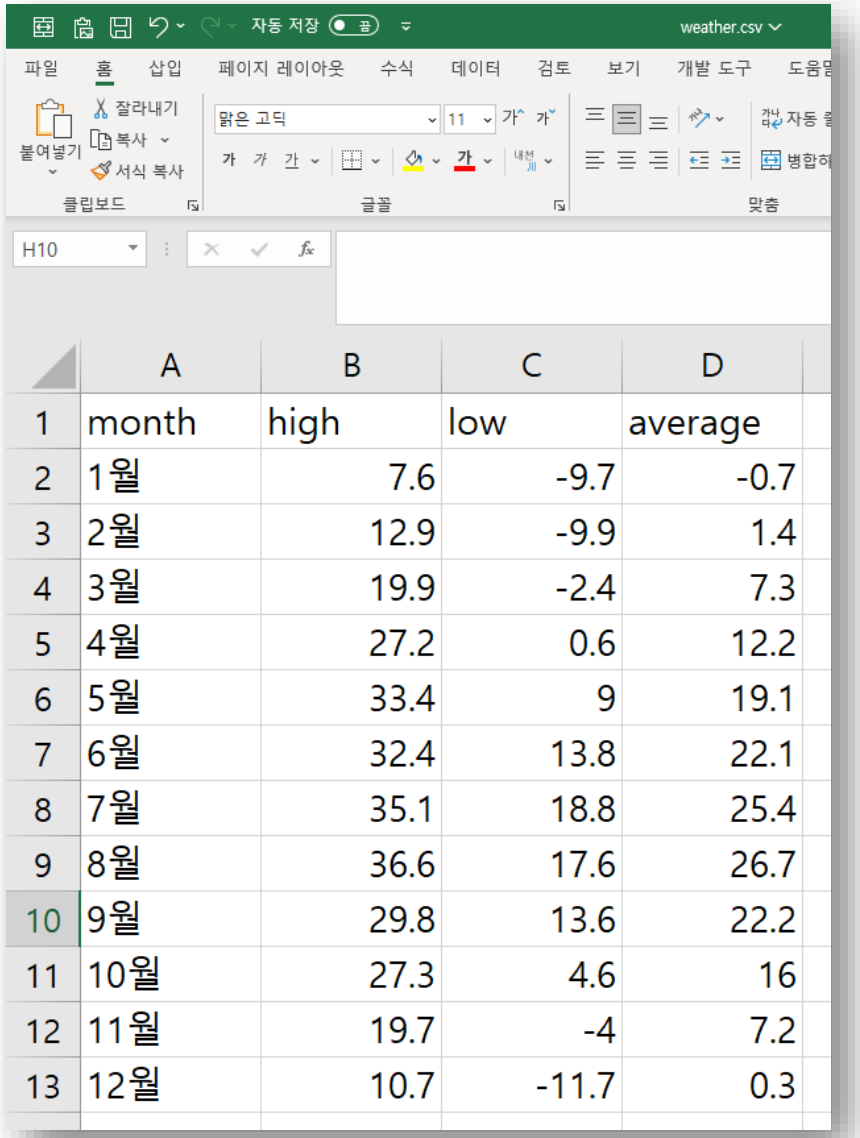
2.3 날씨 데이터 분석

날씨 데이터 분석

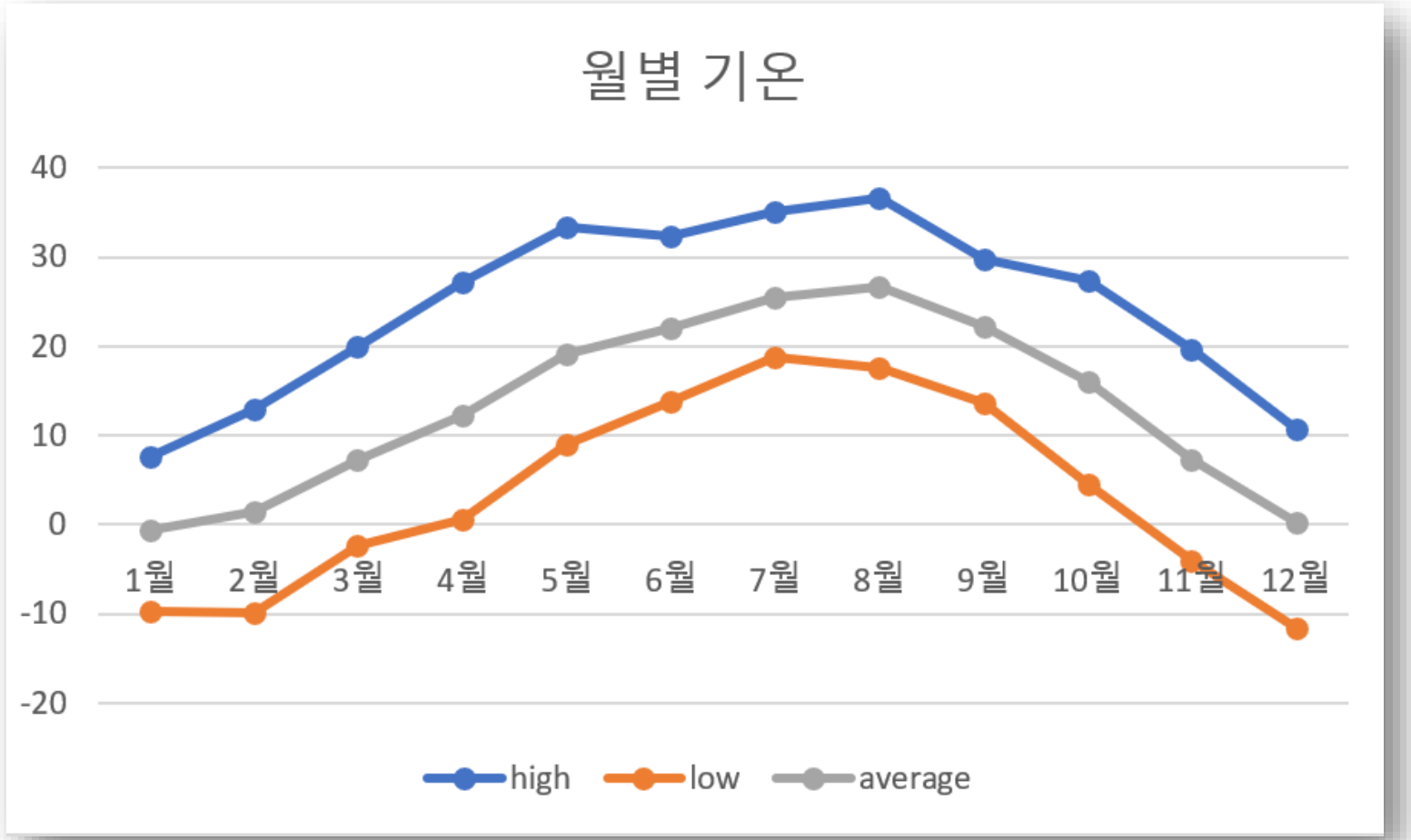


<출처> https://www.data.go.kr/

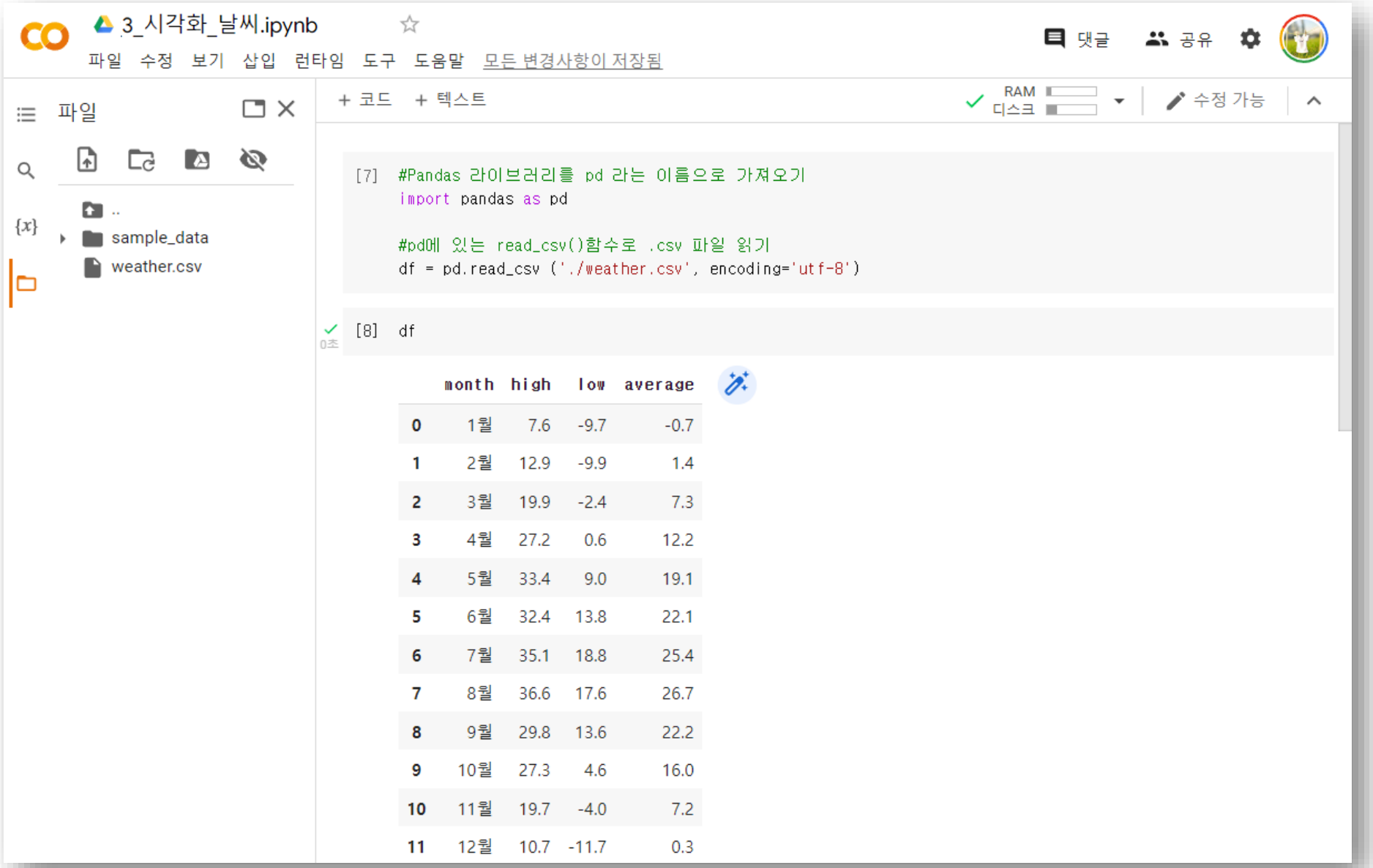
날씨 데이터 분석



	A	B	C	D
1	month	high	low	average
2	1월	7.6	-9.7	-0.7
3	2월	12.9	-9.9	1.4
4	3월	19.9	-2.4	7.3
5	4월	27.2	0.6	12.2
6	5월	33.4	9	19.1
7	6월	32.4	13.8	22.1
8	7월	35.1	18.8	25.4
9	8월	36.6	17.6	26.7
10	9월	29.8	13.6	22.2
11	10월	27.3	4.6	16
12	11월	19.7	-4	7.2
13	12월	10.7	-11.7	0.3



날씨 데이터 분석



	month	high	low	average
0	1월	7.6	-9.7	-0.7
1	2월	12.9	-9.9	1.4
2	3월	19.9	-2.4	7.3
3	4월	27.2	0.6	12.2
4	5월	33.4	9.0	19.1
5	6월	32.4	13.8	22.1
6	7월	35.1	18.8	25.4
7	8월	36.6	17.6	26.7
8	9월	29.8	13.6	22.2
9	10월	27.3	4.6	16.0
10	11월	19.7	-4.0	7.2
11	12월	10.7	-11.7	0.3

날씨 데이터 분석

3_시각화_날씨.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유

+ 코드 + 텍스트 RAM 디스크 수정 가능

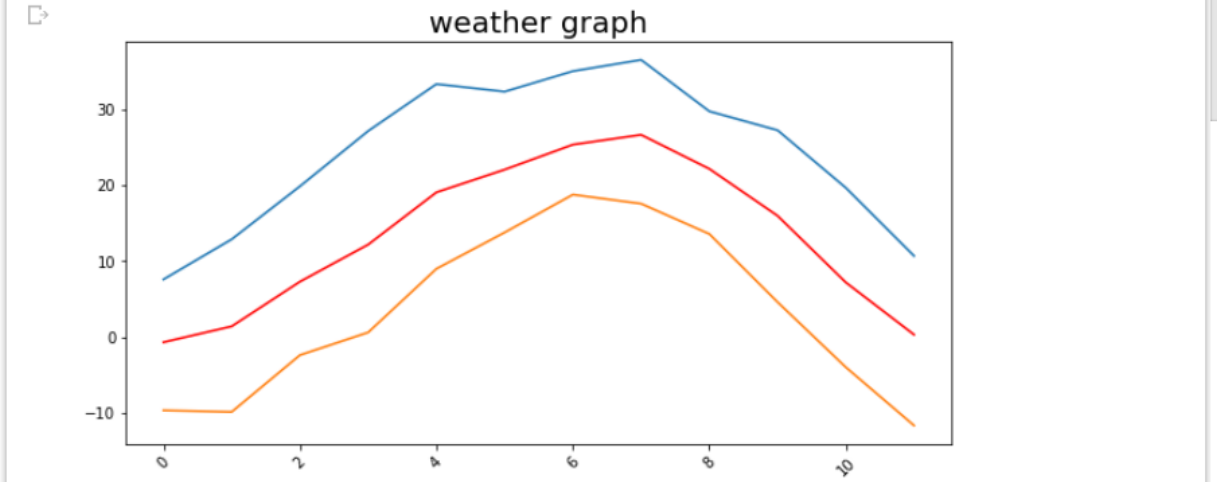
파일

..
sample_data
weather.csv

[8]

10	11월	19.7	-4.0	7.2
11	12월	10.7	-11.7	0.3

```
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(10,5)) ## 캔버스 생성
fig.set_facecolor('white') ## 캔버스 색상 설정
ax = fig.add_subplot() ## 그림 뼈대(프레임) 생성

ax.plot(df['high']) ## 선그래프 생성
ax.plot(df['low']) ## 선그래프 생성
ax.plot(df['average'], color='#ff0000') ## 선그래프 생성

plt.xticks(rotation=45) ## x축 눈금 라벨 설정 - 40도 회전
plt.title('weather graph',fontsize=20) ## 타이틀 설정
plt.show()
```



날씨 데이터 분석

3_시각화_날씨.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유

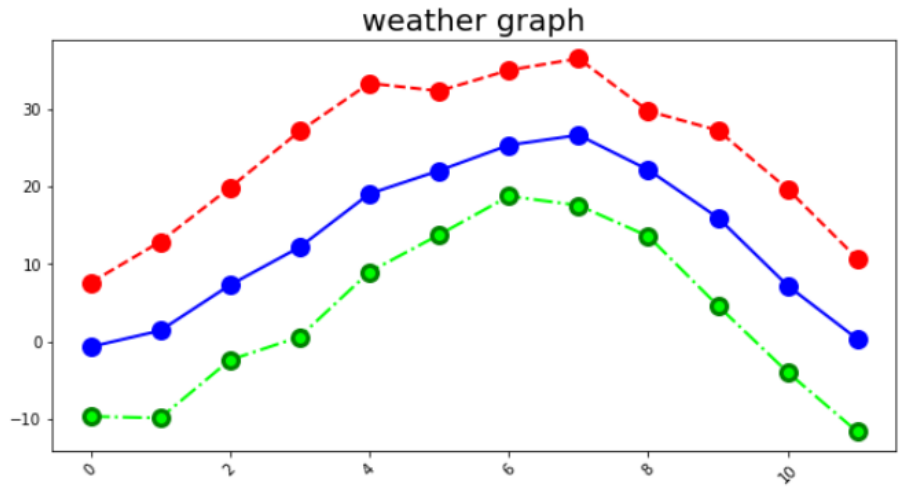
+ 코드 + 텍스트

RAM 디스크 수정 가능

파일

..
sample_data
weather.csv

```
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(10,5)) ## 캔버스 생성
fig.set_facecolor('white') ## 캔버스 색상 설정
ax = fig.add_subplot() ## 그림 뼈대(프레임) 생성

ax.plot(df['high'], color='#ff0000', linestyle='--',linewidth=2, marker='o',markersize=10, markeredgecolor='red', markeredgewidth=3)
ax.plot(df['low'], color='#00ff00', linestyle='dashdot',linewidth=2, marker='o',markersize=10, markeredgecolor='green', markeredgewidth=3)
ax.plot(df['average'], color='#0000ff', linestyle='solid', linewidth=2, marker='o',markersize=10, markeredgecolor='blue', markeredgewidth=3)

plt.xticks(rotation=45) ## x축 눈금 라벨 설정 - 40도 회전
plt.title('weather graph',fontsize=20) ## 타이틀 설정
plt.show()
```



날씨 데이터 분석

3_시각화_날씨.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유

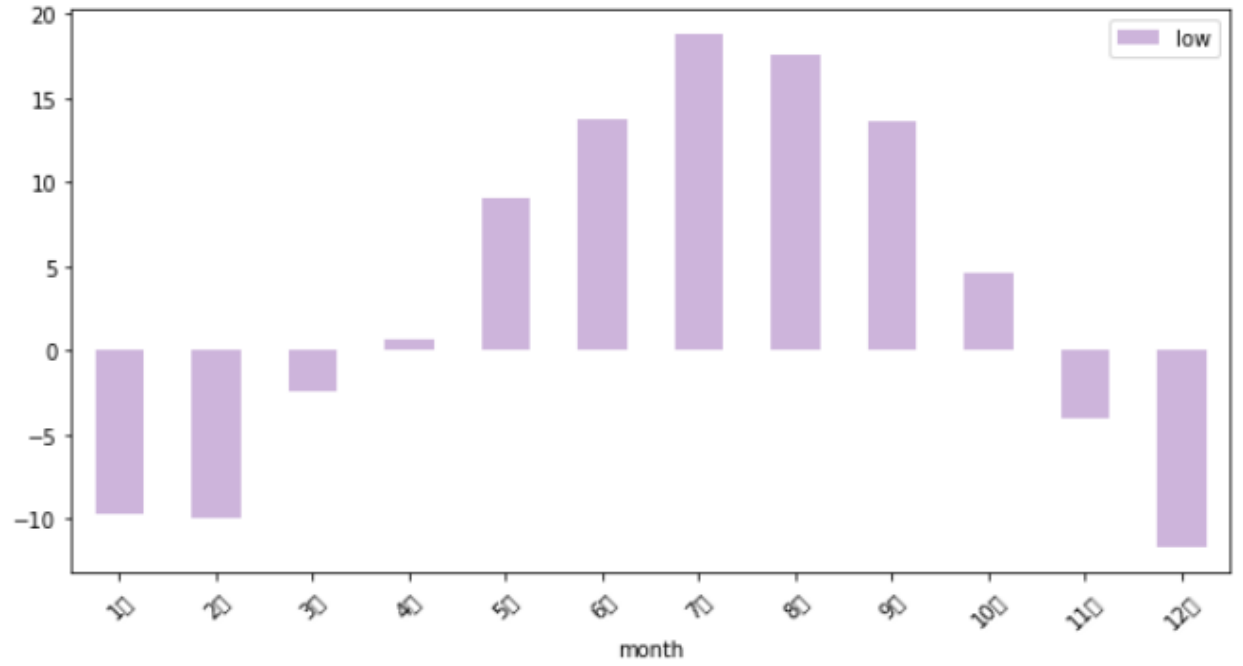
+ 코드 + 텍스트 RAM 디스크 수정 가능

파일

..
sample_data
weather.csv

```
df.plot.bar(x='month', y='low', rot=45, figsize=(10,5), color='#cdb4db')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8dbf059190>
```



2.4 지하철 데이터 분석

지하철 데이터 분석

4_시각화_지하철.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

파일

..
sample_data
subway.csv

+ 코드 + 텍스트

[1] 0초
```
import pandas as pd                    #Pandas 라이브러리를 pd 라는 이름으로 가져오기

df = pd.read_csv ('./subway.csv', encoding='utf-8')    #pd에 있는 read_csv()함수로 .csv 파일 읽기
```

[2] 0초
```
print(df)
```

```
           date line     station     In    Out
0      20221001  3호선        고속터미널  59124  62989
1      20221001  3호선   교대(법원.검찰청)   8040   4875
2      20221001  3호선         학여울   3355   3401
3      20221001  3호선          대청   6517   5926
4      20221001  3호선          일원   6231   6025
...         ...  ...         ...    ...    ...
18780  20221031  6호선         버티고개   2366   2239
18781  20221031  6호선          약수   3778   3751
18782  20221031  6호선          청구   3842   4067
18783  20221031  6호선          신당   8136   8814
18784  20221031  6호선         동묘앞   8434   8751

[18785 rows x 5 columns]
```

[3] 0초
```
df['line'].unique()
```

```
array(['3호선', '4호선', '5호선', '9호선', '경춘선', '8호선', '경인선', '경의선', '9호선2~3단계',
       '6호선', '2호선', '1호선', '분당선', '과천선', '공항철도 1호선', '경원선', '경부선', '수인선',
       '신림선', '안산선', '우이신설선', '일산선', '장항선', '중앙선', '경강선', '7호선'],
      dtype=object)
```

지하철 데이터 분석

4_시각화_지하철.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

파일

..
sample_data
subway.csv

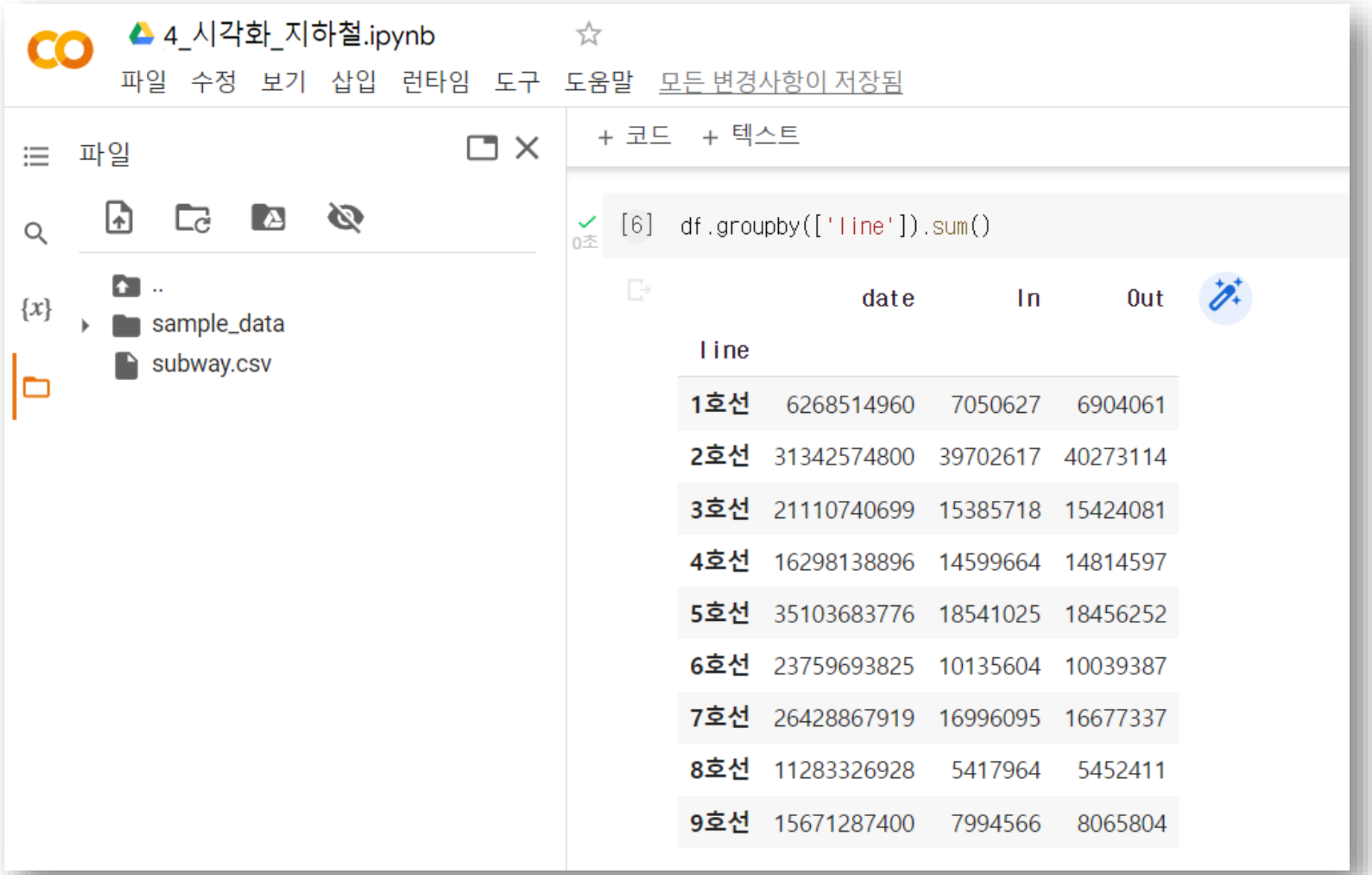
```
filter = df['line'].str.endswith('호선') & ~df['line'].str.contains('공항철도')
df = df[filter]

df['line'].unique()
```

```
array(['3호선', '4호선', '5호선', '9호선', '8호선', '6호선', '2호선', '1호선', '7호선'],
      dtype=object)
```

```
df.groupby('line').mean()
```

line	date	In	Out
1호선	2.022102e+07	22743.958065	22271.164516
2호선	2.022102e+07	25614.591613	25982.654194
3호선	2.022102e+07	14737.277778	14774.023946
4호선	2.022102e+07	18113.727047	18380.393300
5호선	2.022102e+07	10680.313940	10631.481567
6호선	2.022102e+07	8626.045957	8544.159149
7호선	2.022102e+07	13003.898240	12760.013007
8호선	2.022102e+07	9709.612903	9771.345878
9호선	2.022102e+07	10315.569032	10407.489032

지하철 데이터 분석



line	date	In	Out
1호선	6268514960	7050627	6904061
2호선	31342574800	39702617	40273114
3호선	21110740699	15385718	15424081
4호선	16298138896	14599664	14814597
5호선	35103683776	18541025	18456252
6호선	23759693825	10135604	10039387
7호선	26428867919	16996095	16677337
8호선	11283326928	5417964	5452411
9호선	15671287400	7994566	8065804

지하철 데이터 분석

4_시각화_지하철.ipynb

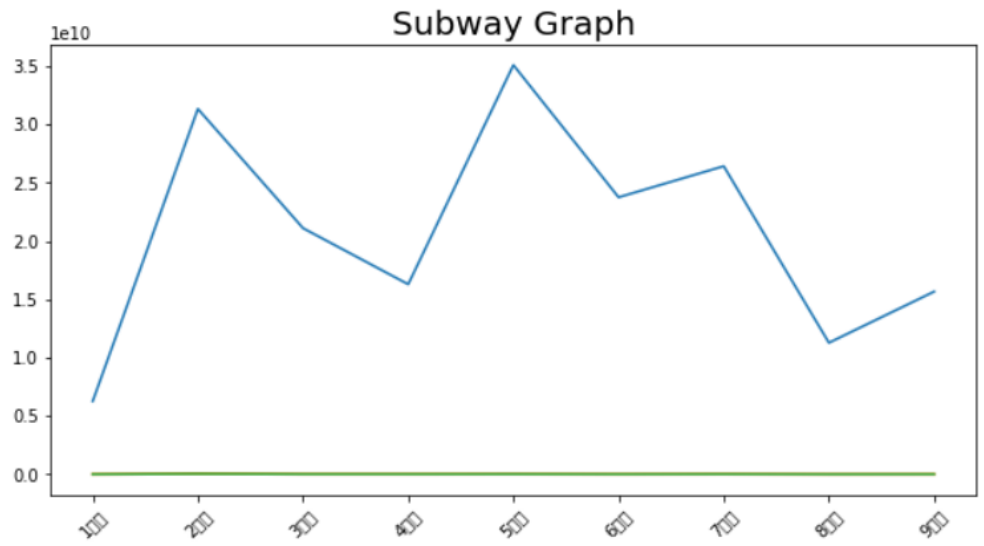
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

파일

..
sample_data
subway.csv

```
import matplotlib.pyplot as plt

fig = plt.figure(figsize=(10,5)) ## 캔버스 생성
fig.set_facecolor('white') ## 캔버스 색상 설정
ax = fig.add_subplot() ## 그림 뼈대(프레임) 생성

sss = df.groupby(['line']).sum()
ax.plot(sss) ## 선그래프 생성

plt.xticks(rotation=45) ## x축 눈금 라벨 설정 - 40도 회전
plt.title('Subway Graph',fontsize=20) ## 타이틀 설정
plt.show()
```



Pandas Package

pandas	엑셀, CSV 등의 파일을 읽어서 데이터프레임에 저장하고 작업한 데이터를 데이터프레임 형태로 구성하여 엑셀, CSV 등의 파일에 저장하는 작업을 수행하는 모듈(이후 pd라는 이름을 사용)
glob	경로와 이름을 지정하여 파일을 다루는 파일 처리 작업을 위한 모듈
re	메타 문자를 이용하여 특정 규칙을 작성하는 정규식을 사용하기 위한 모듈
reduce	2차원 리스트를 1차원 리스트로 차원을 줄이기 위한 모듈
word_tokenize	자연어 처리 패키지(from nltk.tokenize) 중에서 단어 토큰화 작업을 위한 모듈
stopwords	자연어 처리 패키지(from nltk.corpus) 중에서 불용어 정보를 제공하는 모듈
WordNetLemmatizer	자연어 처리 패키지(from nltk.stem) 중에서 단어 형태의 일반화를 위해 표제어 추출을 제공하는 모듈
Counter	데이터 집합에서 개수를 자동으로 계산하기 위한 모듈
matplotlib.pyplot	히스토그램을 그리기 위한 matplotlib 패키지의 내부 모듈(이후 plt라는 이름을 사용)
STOPWORDS, WordCloud	워드클라우드를 그리기 위해 사용할 워드클라우드용 불용어 모듈과 워드클라우드 모듈

<출처> IT CookBook, 데이터 과학 기반의 파이썬 빅데이터 분석, 이지영(2020), 한빛아카데미㈜

Pandas 시각화 라이브러리

시각화 라이브러리	특징
내장라이브러리	판다스에 내장된 기본 그래프 라이브러리로 별도의 라이브러리 임포트 없이 사용 가능
맷플롭립 (Matplotlib)	판다스에서 가장 많이 쓰는 라이브러리로 데이터 프레임을 시각화할 때도 내부적으로 맷플롭립을 사용함
시본 (Seaborn)	시본은 맷플롭립을 기반으로 다양한 색 테마, 차트 기능을 추가한 라이브러리로 맷플롭립 에 의존성을 가지고 있다.
플로트나인 (Plotnine)	plotnine은 R의 ggplot2에 기반해 그래프를 작성하는 라이브러리로 R로 시각화하는 것이 익숙하면 편리하게 사용할 수 있다.
폴리움 (Folium)	Folium은 지도 데이터(Open Street Map)에 leaflet.js를 이용해 위치정보를 시각화하는 라이브러리
파이차트 (Pyecharts)	중국 바이두에서 데이터 시각화를 위해 만든 Echarts.js의 파이썬
플로틀리 (Plot.ly)	plotly는 인터랙티브 그래프를 그려주는 라이브러리로 R, 스칼라, 파이썬, 자바스크립트, 매트랩 등에서 사용할 수 있다.

<출처> 빅데이터 분석과 머신러닝, 이영호(2021), 생능출판사

정리

- 배열 연산을 위한 Numpy의 명령어를 설명할 수 있다.
- 데이터 분석을 위한 Pandas의 명령어를 설명할 수 있다.
- 데이터 시각화를 위한 Matplotlib의 명령어를 설명할 수 있다.

감사합니다