

# ATM 2106 TA Class

May 22, 2019

Department of Atmospheric Sciences, Yonsei University  
Air-Sea Modeling Laboratory

---

## Last time

- Midterm review

## Today

- Ocean circulation
- Linear Regression Analysis
  - 1) using np.polyfit
  - 2) using sklearn library
  - 3) using matrix method
- Practice of Homework #3

# Linear Regression Analysis

## - Simple Example - Latent heat of condensation

$T(\text{°C})$	$e_s(\text{Pa})$	$e_i(\text{Pa})$	$L(\text{J/g})$
0	611.21	611.15	2501
5	872.47		2489
10	1227.94		2477
15	1705.32		2466
20	2338.54		2453
25	3168.74		2442
30	4245.20		2430
35	5626.45		2418
40	7381.27		2406

## - Simple Example - Latent heat

- 1) np.polyfit

```
In [1]: import numpy as np  
import matplotlib.pyplot as plt
```

```
In [2]: #test data set  
L=np.array([2501,2489,2477,2466,2453,2442,2430,2418,2406])  
T=np.linspace(273,273+40,9)  
#변수 선언  
L_new=np.zeros([9])  
L_google=np.zeros([9])
```

# Linear Regression Analysis

- 1) np.polyfit

```
In [4]: z=np.polyfit(T,L,1)
#해당 data set에서 z[0]는 기울기 z[1]은 y절편 값을 구해줌
```

```
In [5]: z
```

```
Out[5]: array([-2.37000000e+00, 3.14796556e+03])
```

```
In [6]: # np.polyfit으로 구한 추세선을 구해주면 아래와 같다.
for i in range(9):
    L_new[i]=z[0]*(273+5*i)+z[1]
```

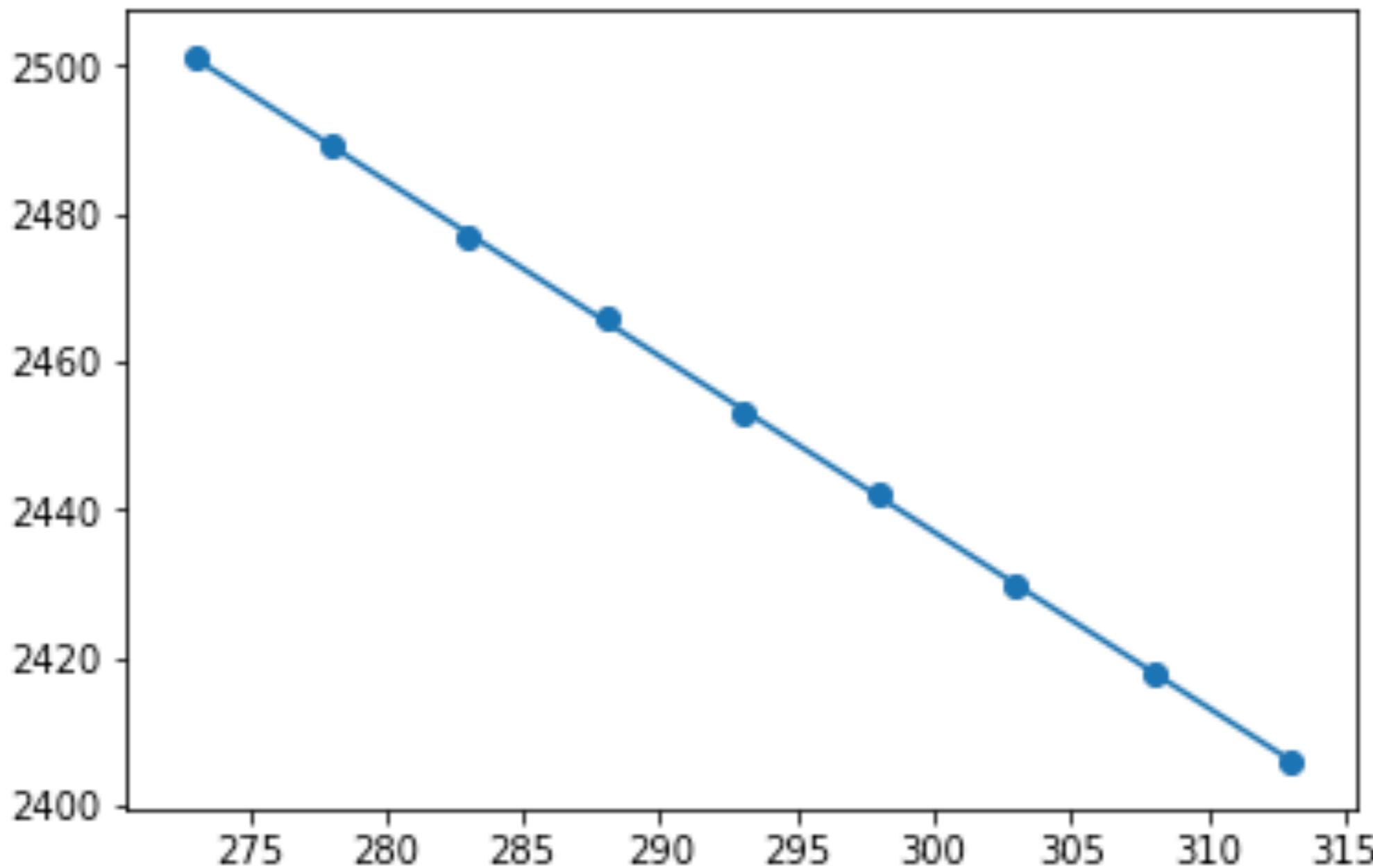
```
In [7]: plt.plot(T,L_new)
plt.scatter(T,L)
```

---

## Linear Regression Analysis

---

- 1) np.polyfit



# Linear Regression Analysis

- 1) np.polyfit

## Specific latent heat for condensation of water in clouds [\[edit\]](#)

The specific latent heat of condensation of water in the temperature range from  $-25^{\circ}\text{C}$  to  $40^{\circ}\text{C}$  is approximated by the following empirical cubic function:

$$L_{\text{water}}(T) = (2500.8 - 2.36T + 0.0016T^2 - 0.00006T^3) \text{ J/g}, \text{ [11]}$$

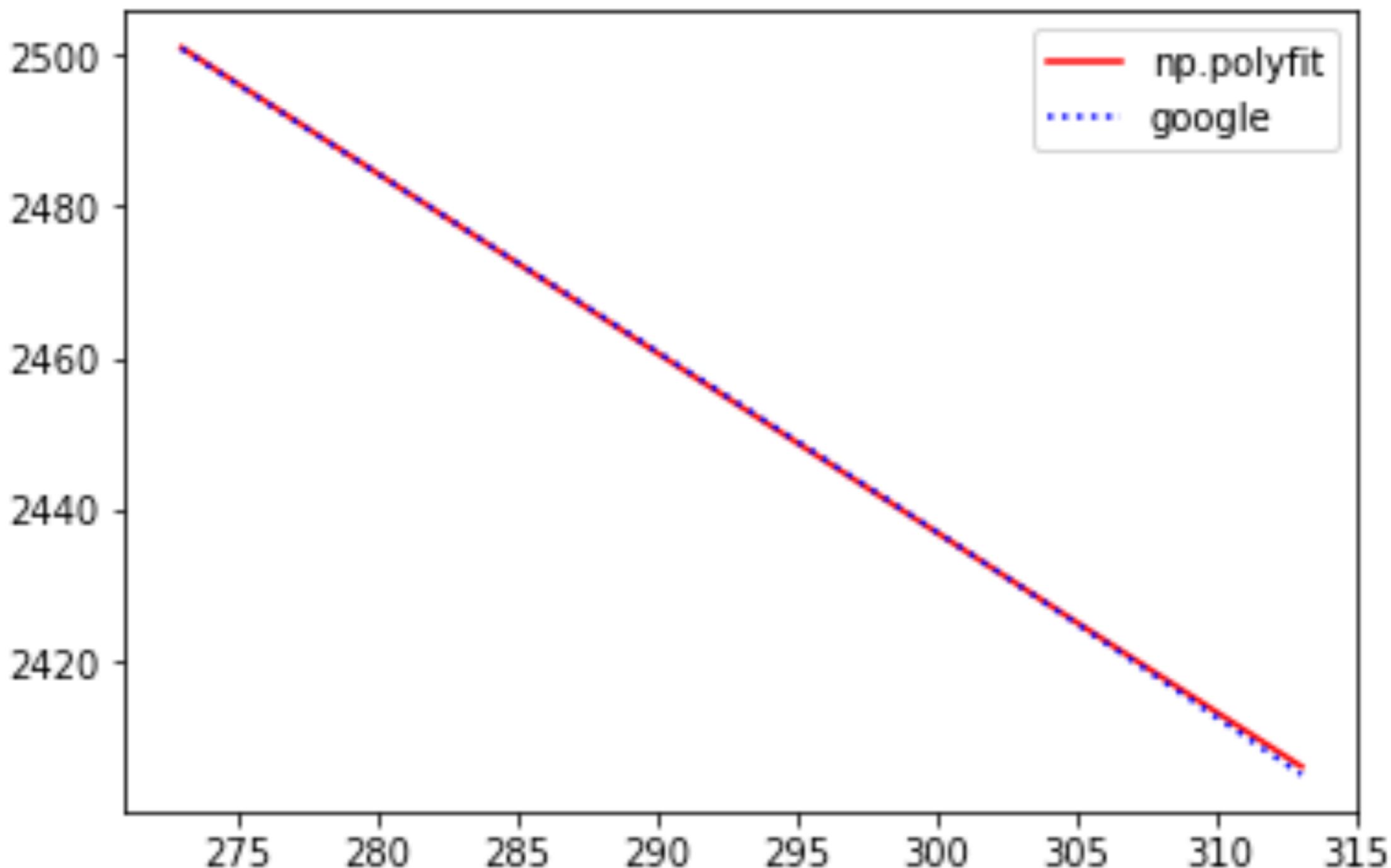
where the temperature  $T$  is taken to be the numerical value in  $^{\circ}\text{C}$ .

```
In [8]: for i in range(9):
    time=5*i
    L_google[i]=2500.8-2.36*time+0.0016*time**2-0.00006*time**3
```

```
In [9]: plt.plot(T,L_new,color='red',label='np.polyfit')
plt.plot(T,L_google,'k:',color='blue',label='google')
plt.legend()
plt.show()
```

## Linear Regression Analysis

- 1) np.polyfit



## Linear Regression Analysis

- 2) Using sklearn library

```
In [1]: import numpy as np  
import matplotlib.pyplot as plt  
from sklearn import linear_model  
from sklearn.metrics import mean_squared_error, r2_score
```

```
In [2]: L=np.array([2501,2489,2477,2466,2453,2442,2430,2418,2406])  
T=np.linspace(273,273+40,9)  
L_new=np.zeros([9])  
L_google=np.zeros([9])
```

# Linear Regression Analysis

## • 2) Using sklearn library

```
In [3]: train_x=T.reshape(-1,1)
train_y=L.reshape(-1,1)
#회귀분석하려는 데이터셋을 쭉 펴준다(1차원 배열로 잡아줌)
```

```
In [4]: regr=linear_model.LinearRegression()
regr.fit(train_x,train_y)
predict_y=regr.predict(train_x)
#위에서 reshape 해준 데이터의 표본을 넣어주고 이를 바탕으로 회귀분석 값을 구해준다.
```

```
In [5]: regr.coef_
#회귀분석에서의 기울기 값
```

Out[5]: array([[-2.37]])

```
In [6]: regr.intercept_
#회귀분석에서의 y절편 값
```

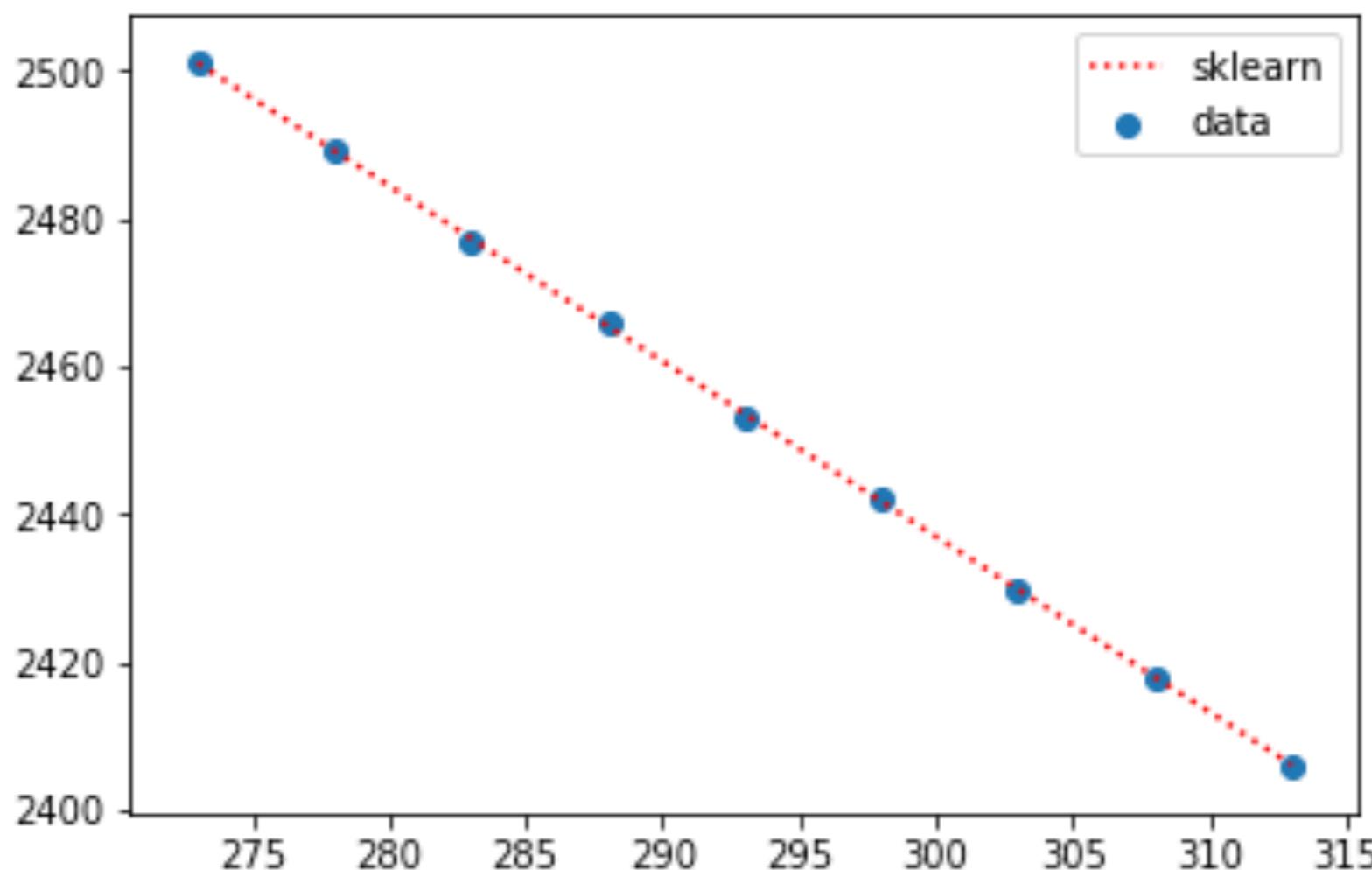
Out[6]: array([3147.96555556])

# Linear Regression Analysis

- 2) Using sklearn library

```
In [7]: plt.plot(T,predict_y,'k:',color='red',label='sklearn')  
plt.scatter(T,L,label='data')  
plt.legend()
```

Out[7]: <matplotlib.legend.Legend at 0x2b9a13c3e1d0>



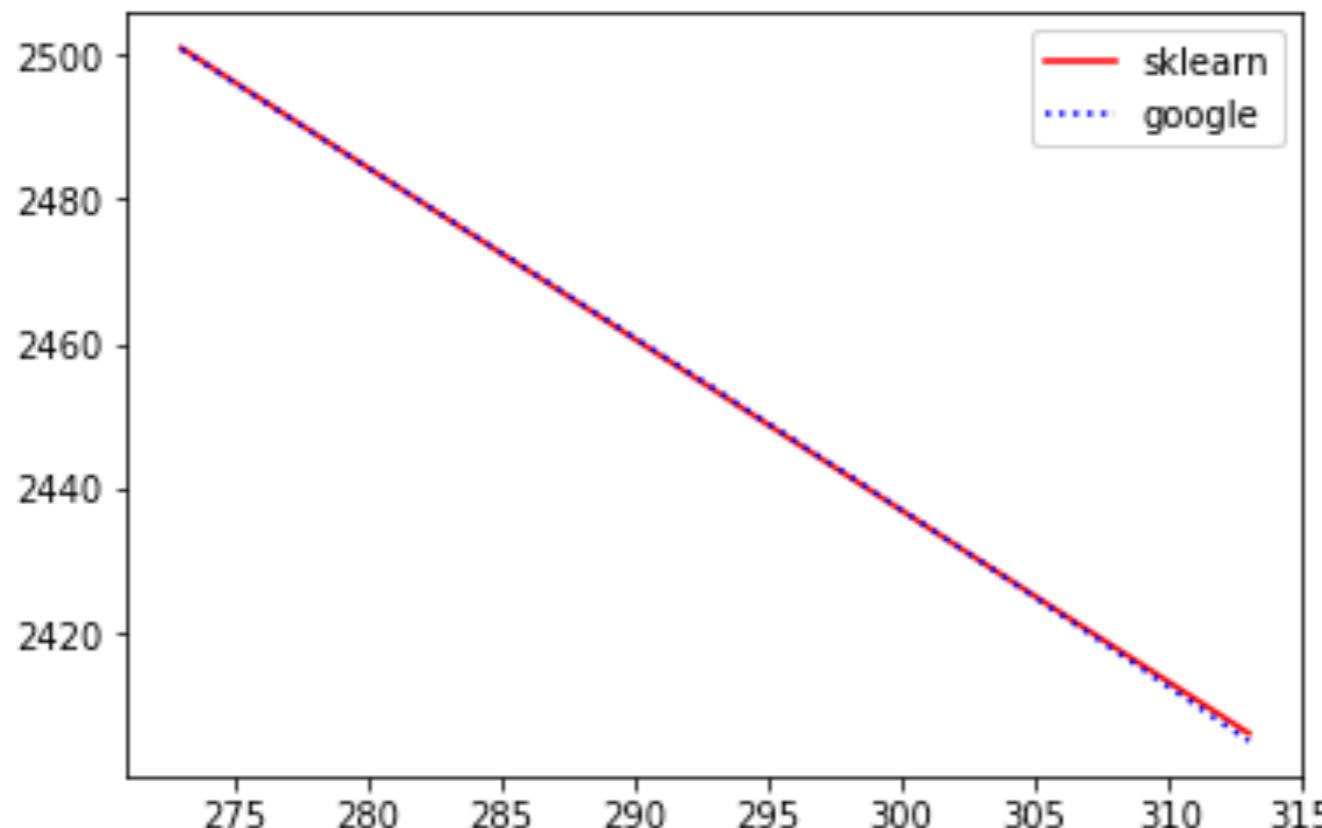
# Linear Regression Analysis

- 2) Using sklearn library

```
In [8]: for i in range(9):
    time=5*i
    L_google[i]=2500.8-2.36*time+0.0016*time**2-0.00006*time**3
```

```
In [9]: plt.plot(T,predict_y,color='red',label='sklearn')
plt.plot(T,L_google,'k:',color='blue',label='google')
#plt.scatter(T,L)
plt.legend()
```

Out[9]: <matplotlib.legend.Legend at 0x2b9a13eb2a90>



# Linear Regression Analysis

- 3) using matrix method

```
In [1]: import numpy as np  
import matplotlib.pyplot as plt
```

```
In [2]: #test data set  
L=np.array([2501,2489,2477,2466,2453,2442,2430,2418,2406])  
T=np.linspace(273,273+40,9)  
  
# data points in column vector [input(=x), output(=y)]  
# reshape하는 이유는 (n, )의 형태로 shape를 바꾼후 행렬연산을 하기 위함  
x = T.reshape(-1, 1)  
y = L.reshape(-1, 1)  
  
A = np.hstack([x, x**0])  
A = np.asmatrix(A)  
  
theta = (A.T*A).I*A.T*y  
  
print('theta:\n', theta)  
L_new=np.zeros([9])  
L_google=np.zeros([9])
```

```
theta:  
[[ -2.37000000e+00]  
 [ 3.14796556e+03]]
```

# Linear Regression Analysis

- theta를 구할 때 사용한 식 설명

Let  $X = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix}$   $y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$   $\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}$

기울기  
y절편

Find optional  $\beta$  set  $X\beta \rightarrow y$  best approximation

$$X\beta = \begin{bmatrix} \beta_0 x_1 + \beta_1 \\ \beta_0 x_2 + \beta_1 \\ \vdots \\ \beta_0 x_n + \beta_1 \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

# Linear Regression Analysis

- theta를 구할 때 사용한 식 설명

define error as  $e(\beta) = y - X\beta = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}$

$$\text{MSE}(\beta) = \frac{1}{n} \sum_{i=1}^n e_i^2(\beta) = \frac{1}{n} (e_1^2 + e_2^2 + \dots + e_n^2)$$

Mean Squared Error

이를 matrix form으로 바꾸면

$$\frac{1}{n} [e_1 \ e_2 \ \dots \ e_n] \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix} = \frac{1}{n} e^T e$$

$$\begin{aligned} \text{MSE}(\beta) &= \frac{1}{n} e^T e = \frac{1}{n} (y - X\beta)^T (y - X\beta) \\ &= \frac{1}{n} (y^T y - 2\beta^T X^T y + \beta^T X^T X \beta) \end{aligned}$$

## Linear Regression Analysis

- theta를 구할 때 사용한 식 설명

error의 최소값을 구하려면 미분해서 gradient = 0 을 찾아야 한다.

$$\nabla \text{MSE}(\beta) = \frac{2}{n} (x^T x \beta - x^T y)$$

error를 최소화 하는  $\beta$ 를  $\hat{\beta} = [\hat{\beta}_0, \hat{\beta}_1]$  이라고 하면

$$\hat{\beta} = (x^T x)^{-1} x^T y$$

따라서 우리는 코드에 마지막 줄만 사용하여 구해주면 된다.

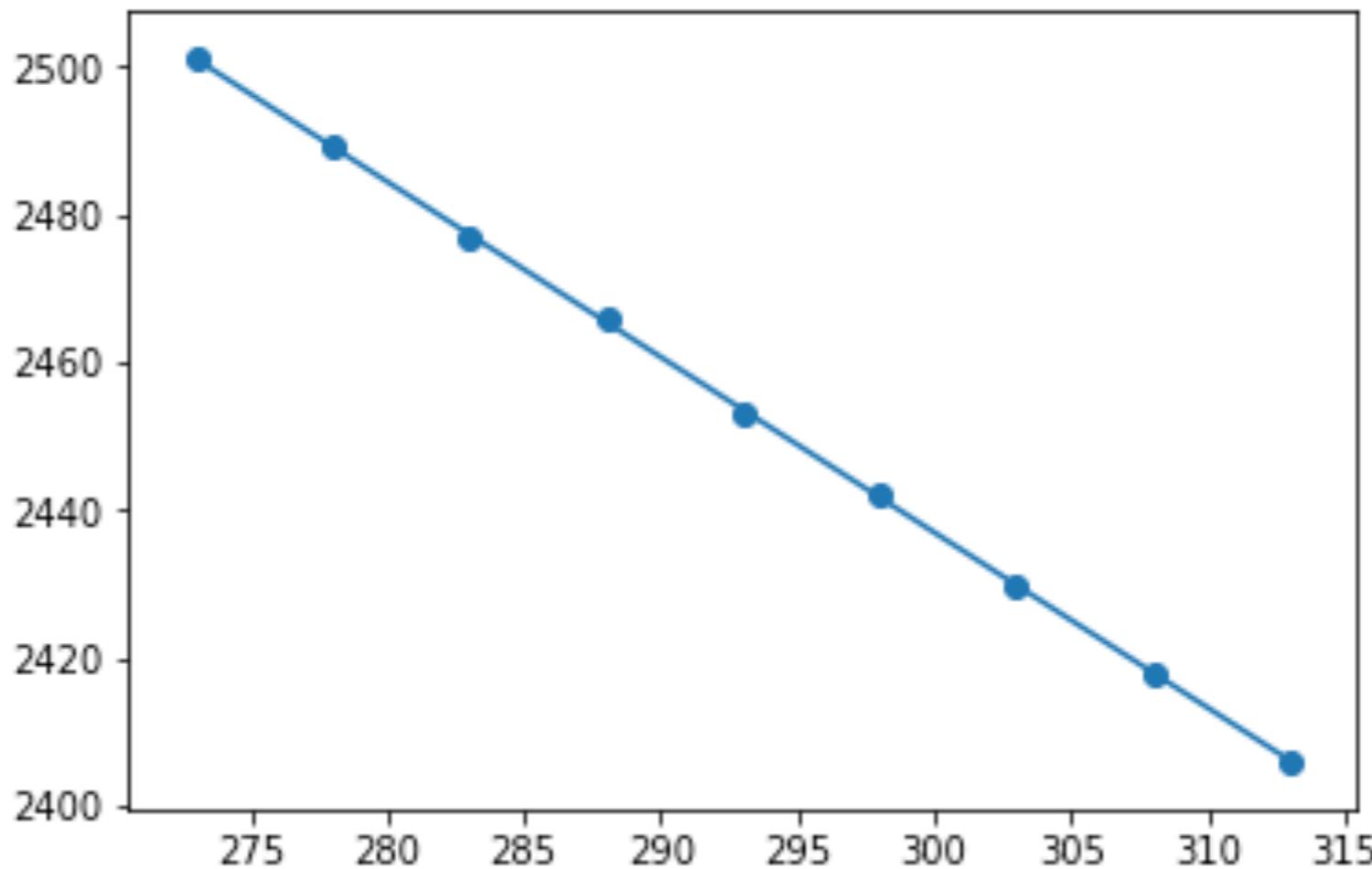
# Linear Regression Analysis

- 3) using matrix method

```
In [3]: # 위에서 구한 방법으로 구한 추세선을 구해주면 아래와 같다.  
for i in range(9):  
    L_new[i]=theta[0,0]*(273+5*i)+theta[1,0]
```

```
In [4]: plt.plot(T,L_new)  
plt.scatter(T,L)
```

Out[4]: <matplotlib.collections.PathCollection at 0x2b3d98c7d9e8>

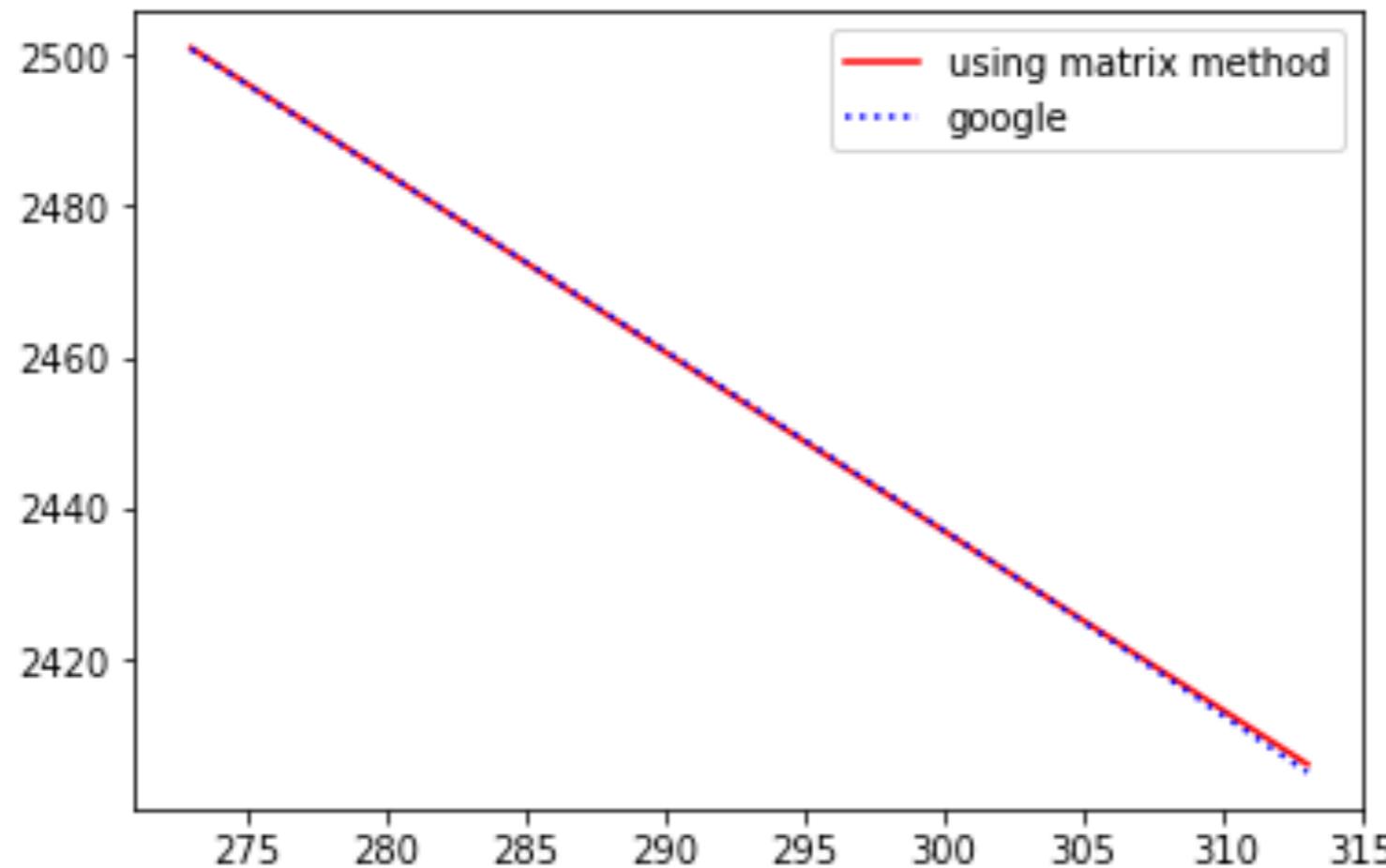


# Linear Regression Analysis

- 3) using matrix method

```
In [5]: for i in range(9):
    time=5*i
    L_google[i]=2500.8-2.36*time+0.0016*time**2-0.00006*time**3
```

```
In [8]: plt.plot(T,L_new,color='red',label='using matrix method')
plt.plot(T,L_google,'k:',color='blue',label='google')
plt.legend()
plt.show()
```



위의 방법중 하나를 사용해서 HW#3에 접근해보자

- **meshgrid**

1D 변수를 2D 변수로 바꾸어줌

```
[X,Y]=np.meshgrid(lon,lat)
```

- **pcolormesh**

그림 그리는 방법 설명 (2D 변수 그리기)

```
plt.plot(X, variable[1D])
```

```
plt.pcolormesh(X,Y, variable[2D])
```

- **2D 배열 유의하기!**

## Practice of Homework #3

Plot 2D graph example)

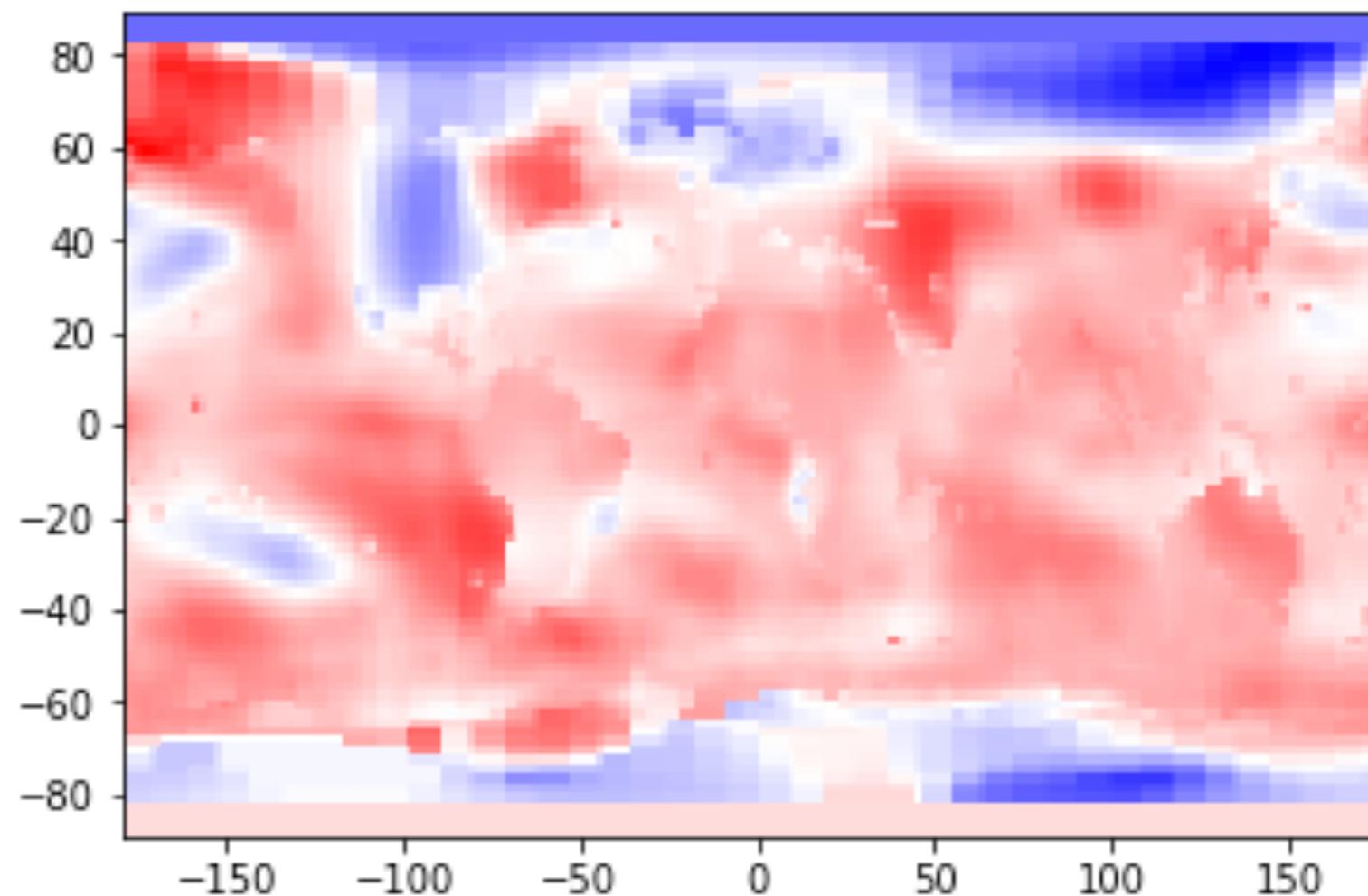
: 예시 파일 HW#3 결과와는 관련없음

In [8]: `[X,Y]=np.meshgrid(lon,lat)`

`plt.pcolormesh(X,Y,T[0,:,:],cmap='bwr')`

#yr=0(1979년)에서의 경도,위도에 해당하는 값을 2D graph로 그린 내용

Out[8]: <matplotlib.collections.QuadMesh at 0x11bfa5c18>



## Practice of Homework #3

반복문 for문 example)

: 예시 파일 HW#3 결과와는 관련없음

In [3]: `test=np.zeros([90,180])`

In [4]: #반복문은 여러종류가 있는데 그중 예제에 나와있는 for문을 살펴보자

`for i in range(90):`

#위 의미는 i라는 변수를 0~89(90개)까지 +1씩 더하면서 아래의 코드를 반복하게 된다

`for j in range(180):`

#마찬가지로 j라는 변수를 0~179(180개)까지 +1씩 더하면서 아래의 코드를 반복하게 된다.

`test[i,j]=i+j`

#위의 for문은 한칸 tab 간격 들여쓰기 되는 코드를 반복하게 되는데 위의 `test[i,j]=i+j`라는 코드를 i 90번,

#각각의 i에 대하여 j 180번 총  $90 \times 180$ 번 반복하게 된다.

#위에서 언급했던 선형회귀분석을 통해 각 지점에서의 온도의 평균 변화율을 구하게되면

#for문을 통해 모든 지점의 평균변화율을 구하고 이를 그래프로 나타낼 수 있다.

---

# Thank You

- End of the Document -

---