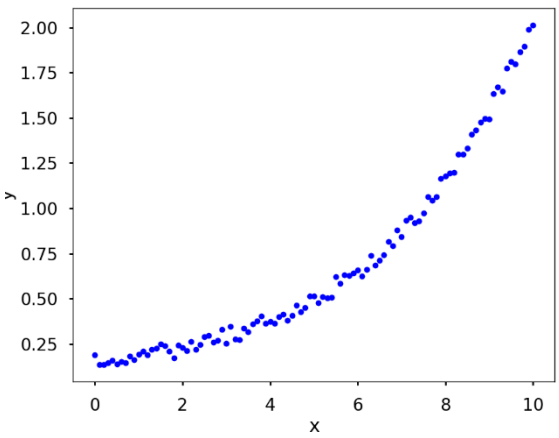
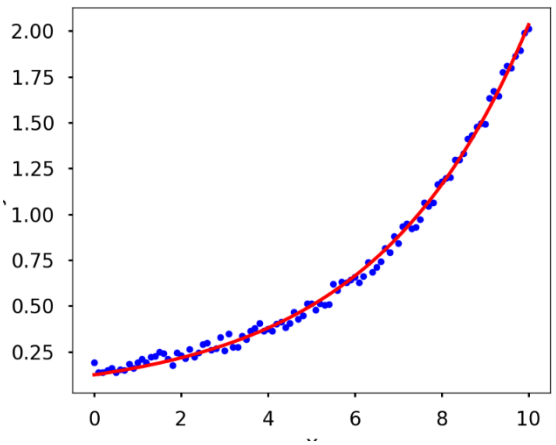
Scipy.optimize 패키지는 최적화 문제를 푸는데 도움을 주는 라이브러리이다. 해당 라이브러리 안에서 Least_squares minimization 과 curve fitting 에 활용할 수 있는 메서드는 least_squares, curve_fit 이 있다.

```
leastsq
    Levenberg-Marquadt 알고리즘의 MINPACK 구현을 위한 레거시 래퍼입니다.
curve_fit
    곡선 맞춤 문제에 적용된 최소 제곱 최소화.
```

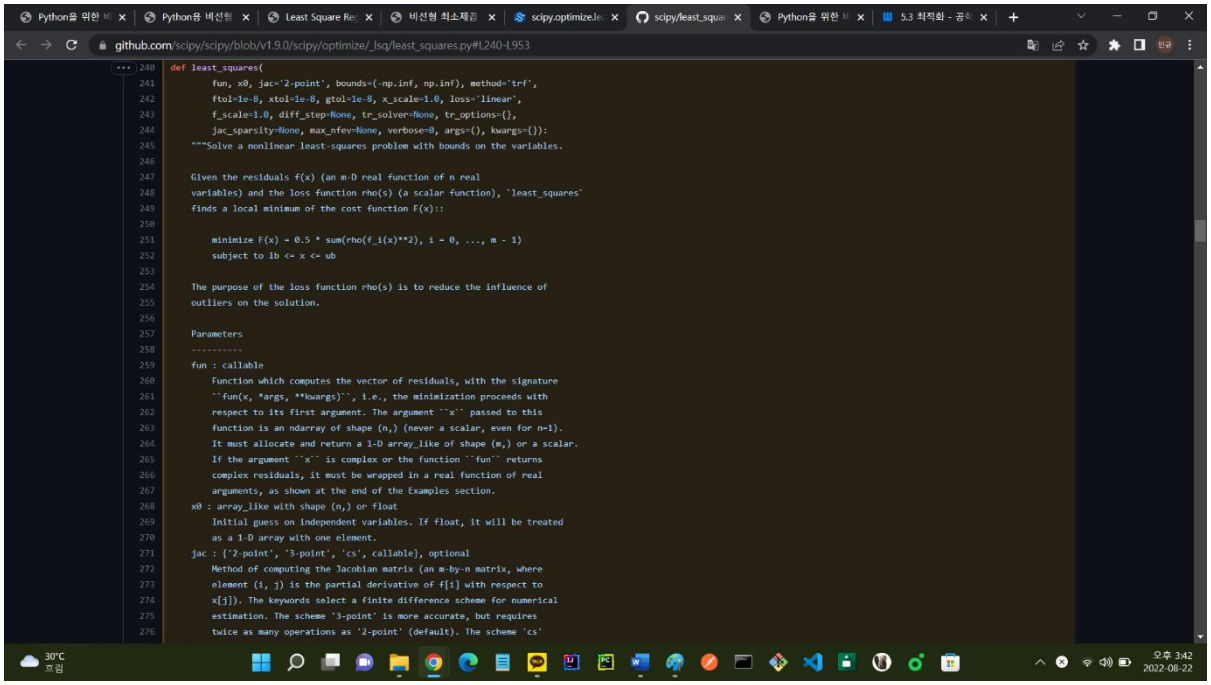
optimize.curve_fit from scipy

```python
import numpy as np
from scipy import optimize
import matplotlib.pyplot as plt

plt.style.use('seaborn-poster')

x = np.linspace(0, 10, 101)
y = 0.1*np.exp(0.3*x) + 0.1*np.random.random(len(x))

plt.figure(figsize = (10,8))
plt.plot(x, y, 'b.')
plt.xlabel('x')
plt.ylabel('y')
plt.show()

def func(x, a, b):
    y = a*np.exp(b*x)
    return y

alpha, beta = optimize.curve_fit(func, xdata = x, ydata = y)[0]
print(f'alpha={alpha}, beta={beta}')

# Let's have a look of the data
plt.figure(figsize = (10,8))
plt.plot(x, y, 'b.')
plt.plot(x, alpha*np.exp(beta*x), 'r')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

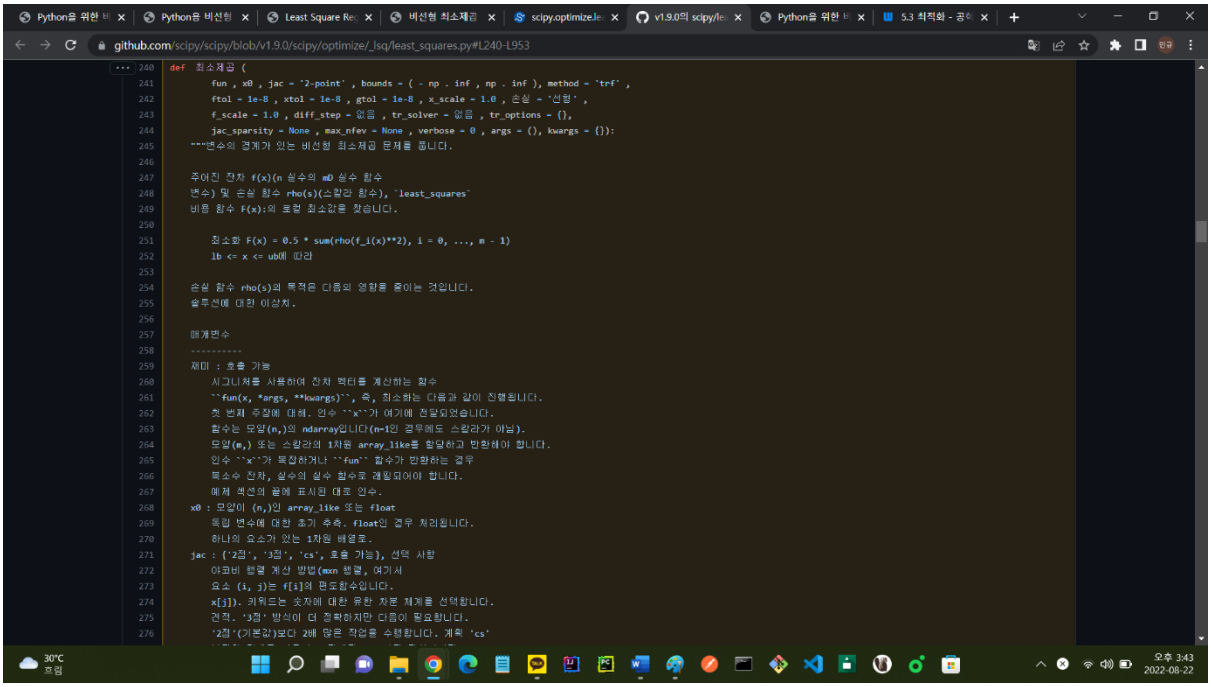




optimize.least_squares from scipy



```
def least_squares(
        fun, x0, jac='2-point', bounds=(-np.inf, np.inf), method='trf',
        ftol=1e-8, xtol=1e-8, gtol=1e-8, x_scale=1.0, loss='linear',
        f_scale=1.0, diff_step=None, tr_solver=None, tr_options={},
        jac_sparsity=None, max_nfev=None, verbose=0, args=(), kwargs={}):
    """Solve a nonlinear least-squares problem with bounds on the variables.

    Given the residuals f(x) (an m-D real function of n real
    variables) and the loss function rho(s) (a scalar function), `least_squares`
    finds a local minimum of the cost function F(x)::

        minimize F(x) = 0.5 * sum(rho(f_i(x)**2), i = 0, ..., m - 1)
        subject to lb <= x <= ub

    The purpose of the loss function rho(s) is to reduce the influence of
    outliers on the solution.

    Parameters
    ----------
    fun : callable
        Function which computes the vector of residuals, with the signature
        ``fun(x, *args, **kwargs)``, i.e., the minimization proceeds with
        respect to its first argument. The argument ``x`` passed to this
        function is an ndarray of shape (n,) (never a scalar, even for n=1).
        It must allocate and return a 1-D array_like of shape (m,) or a scalar.
        If the argument ``x`` is complex or the function ``fun`` returns
        complex residuals, it must be wrapped in a real function of real
        arguments, as shown at the end of the Examples section.
    x0 : array_like with shape (n,) or float
        Initial guess on independent variables. If float, it will be treated
        as a 1-D array with one element.
    jac : {'2-point', '3-point', 'cs', callable}, optional
        Method of computing the Jacobian matrix (an m-by-n matrix, where
        element (i, j) is the partial derivative of f[i] with respect to
        x[j]). The keywords select a finite difference scheme for numerical
        estimation. The scheme '3-point' is more accurate, but requires
        twice as many operations as '2-point' (default). The scheme 'cs'
```



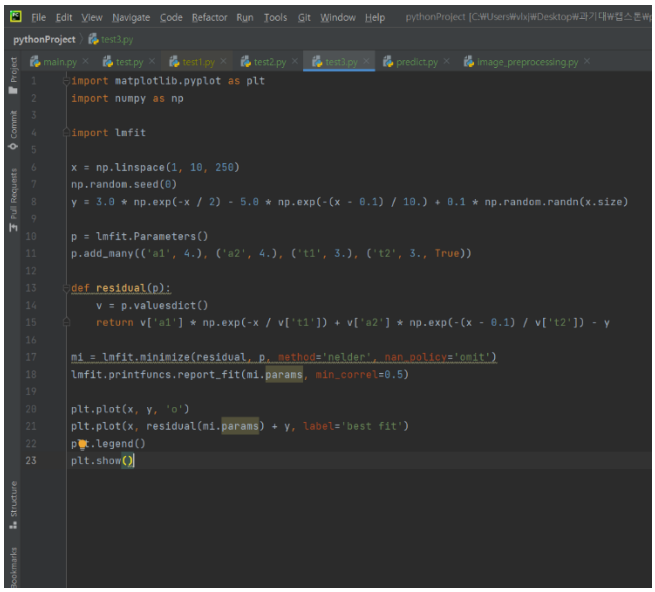
```
def 최소제곱 (
        fun , x0 , jac = '2-point' , bounds = ( - np . inf , np . inf ), method = 'trf' ,
        ftol = 1e-8 , xtol = 1e-8 , gtol = 1e-8 , x_scale = 1.0 , 손실 = '선형' ,
        f_scale = 1.0 , diff_step = 없음 , tr_solver = 없음 , tr_options = {},
        jac_sparsity = None , max_nfev = None , verbose = 0 , args = (), kwargs = {}):
    """변수의 경계가 있는 비선형 최소제곱 문제를 풉니다.

    주어진 잔차 f(x)(n 실수의 mD 실수 함수
    변수) 및 손실 함수 rho(s)(스칼라 함수), `least_squares`
    비용 함수 F(x):의 로컬 최소값을 찾습니다.

        최소화 F(x) = 0.5 * sum(rho(f_i(x)**2), i = 0, ..., m - 1)
        lb <= x <= ub에 따라

    손실 함수 rho(s)의 목적은 다음의 영향을 줄이는 것입니다.
    솔루션에 대한 이상치.

    매개변수
    ----------
    제미 : 호출 가능
        시그니쳐를 사용하여 잔차 벡터를 계산하는 함수
        ``fun(x, *args, **kwargs)``, 즉, 최소화는 다음과 같이 진행됩니다.
        첫 번째 주장에 대해. 인수 ``x``가 여기에 전달되었습니다.
        함수는 모양(n,)의 ndarray입니다(n=1인 경우에도 스칼라가 아님).
        모양(m,) 또는 스칼라의 1차원 array_like를 할당하고 반환해야 합니다.
        인수 ``x``가 복잡하거나 ``fun`` 함수가 반환하는 경우
        복소수 잔차, 실수의 실수 함수로 래핑되어야 합니다.
        예제 섹션의 끝에 표시된 대로 인수.
    x0 : 모양이 (n,)인 array_like 또는 float
        독립 변수에 대한 초기 추측. float인 경우 처리됩니다.
        하나의 요소가 있는 1차원 배열로.
    jac : ('2점', '3점', 'cs', 호출 가능), 선택 사항
        야코비 행렬 계산 방법(mxn 행렬, 여기서
        요소 (i, j)는 f[i]의 편도함수입니다.
        x[j]). 키워드는 숫자에 대한 유한 차분 체계를 선택합니다.
        권적. '3점' 방식이 더 정확하지만 다음이 필요합니다.
        '2점'(기본값)보다 2배 많은 작업을 수행합니다. 계획 'cs'
```

Non-Linear Least-Squares Minimization and Curve-Fitting for python

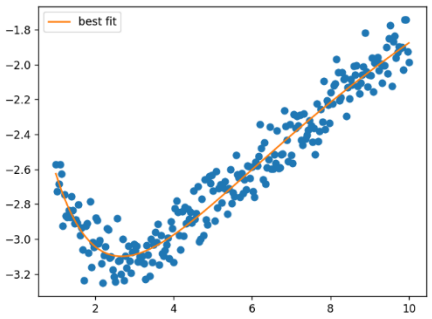
lmfit 패키지는 비선형 최소제곱 문제에 대한 복잡한 피팅 모델을 구축하고 이러한 모델을 실제 데이터에 적용하는데 도움이 되는 간단한 도구를 제공한다.

pip install lmfit

lmfit.minimize 사용예시



```python
import matplotlib.pyplot as plt
import numpy as np

import lmfit

x = np.linspace(1, 10, 250)
np.random.seed(0)
y = 3.0 * np.exp(-x / 2) - 5.0 * np.exp(-(x - 0.1) / 10.) + 0.1 * np.random.randn(x.size)

p = lmfit.Parameters()
p.add_many(('a1', 4.), ('a2', 4.), ('t1', 3.), ('t2', 3., True))

def residual(p):
    v = p.valuesdict()
    return v['a1'] * np.exp(-x / v['t1']) + v['a2'] * np.exp(-(x - 0.1) / v['t2']) - y

mi = lmfit.minimize(residual, p, method='nelder', nan_policy='omit')
lmfit.printfuncs.report_fit(mi.params, min_correl=0.5)

plt.plot(x, y, 'o')
plt.plot(x, residual(mi.params) + y, label='best fit')
plt.legend()
plt.show()
```



https://lmfit.github.io/lmfit-py/intro.html

관련 웹사이트 링크.