Exercise 1 (Partial Derivatives)

a)

$$\frac{\partial f}{\partial x} = 4x^3y^3 + 5x^4$$

$$\frac{\partial f}{\partial y} = 3y^2x^4 - e^y$$

b)

$$\frac{\partial f}{\partial x} = -\frac{1}{2(x^3+xy+y^2)^{\frac{3}{2}}}\frac{\partial}{\partial x}(x^3+xy+y^2)$$

$$\frac{\partial f}{\partial x} = -\frac{3x^2+y}{2(x^3+xy+y^2)^{\frac{3}{2}}}$$

$$\frac{\partial f}{\partial y} = -\frac{1}{2(x^3+xy+y^2)^{\frac{3}{2}}}\frac{\partial}{\partial y}(x^3+xy+y^2)$$

$$\frac{\partial f}{\partial y} = -\frac{2y+x}{2(x^3+xy+y^2)^{\frac{3}{2}}}$$

c)

$$\frac{\partial f}{\partial x} = \frac{\frac{\partial}{\partial x}(x^3+y^2)(x+y) - \frac{\partial}{\partial x}(x+y)(x^3+y^2)}{(x+y)^2}$$

$$\frac{\partial f}{\partial x} = \frac{3x^2(x+y) - (x^3+y^2)}{(x+y)^2}$$

$$\frac{\partial f}{\partial x} = \frac{3x^3+3x^2y-x^3-y^2}{(x+y)^2}$$

$$\frac{\partial f}{\partial x} = \frac{2x^3+3x^2y-y^2}{(x+y)^2}$$

$$\frac{\partial f}{\partial y} = \frac{\frac{\partial}{\partial y}(x^3+y^2)(x+y) - \frac{\partial}{\partial y}(x+y)(x^3+y^2)}{(x+y)^2}$$

$$\frac{\partial f}{\partial y} = \frac{2y(x+y) - (x^3+y^2)}{(x+y)^2}$$

$$\frac{\partial f}{\partial y} = \frac{2y^2+2xy-x^3-y^2}{(x+y)^2}$$

$$\frac{\partial f}{\partial y} = \frac{y^2+2xy-x^3}{(x+y)^2}$$

Exercise 2 (Gradients)

a)

$$\nabla f(\vec{x}) = 2\vec{x}$$

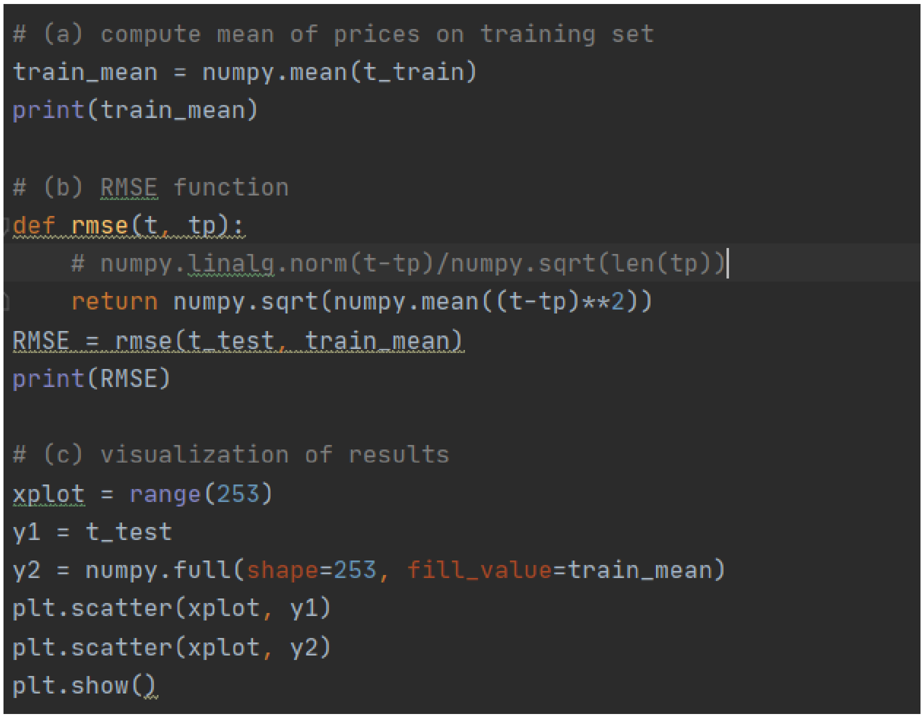
b)

$$\nabla f(\vec{x}) = \vec{b}$$

c)

$$\nabla f(\vec{x}) = 2A\vec{x} + \vec{b}$$

Exercise 3 (Estimating House)

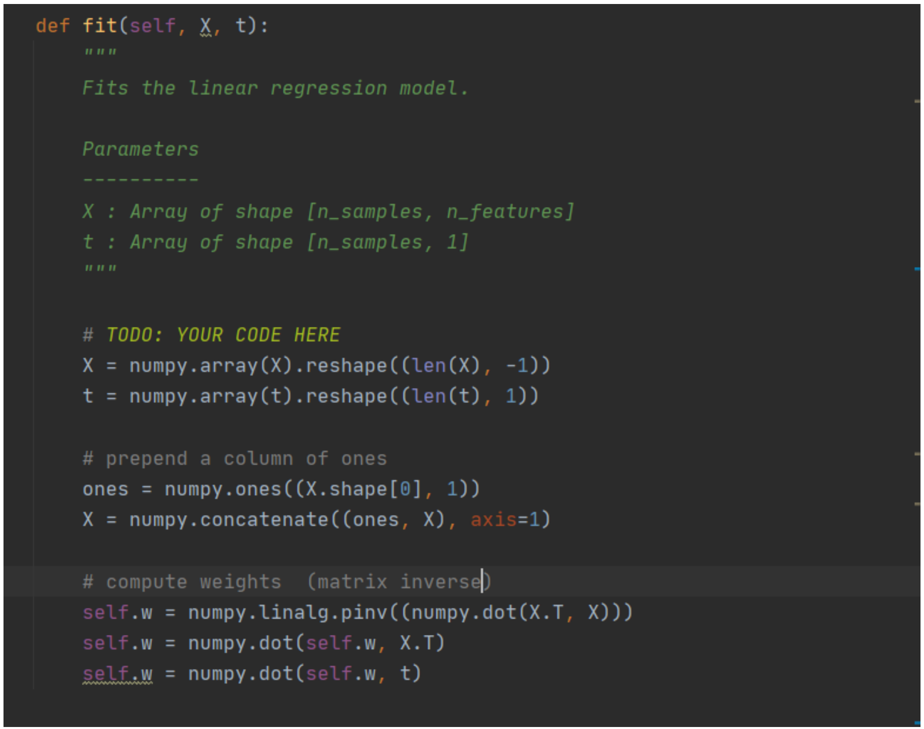


```
# (a) compute mean of prices on training set
train_mean = numpy.mean(t_train)
print(train_mean)

# (b) RMSE function
def rmse(t, tp):
    # numpy.linalg.norm(t-tp)/numpy.sqrt(len(tp))
    return numpy.sqrt(numpy.mean((t-tp)**2))
RMSE = rmse(t_test, train_mean)
print(RMSE)

# (c) visualization of results
xplot = range(253)
y1 = t_test
y2 = numpy.full(shape=253, fill_value=train_mean)
plt.scatter(xplot, y1)
plt.scatter(xplot, y2)
plt.show()
```

Exercise 4 (Estimating House Prices II)



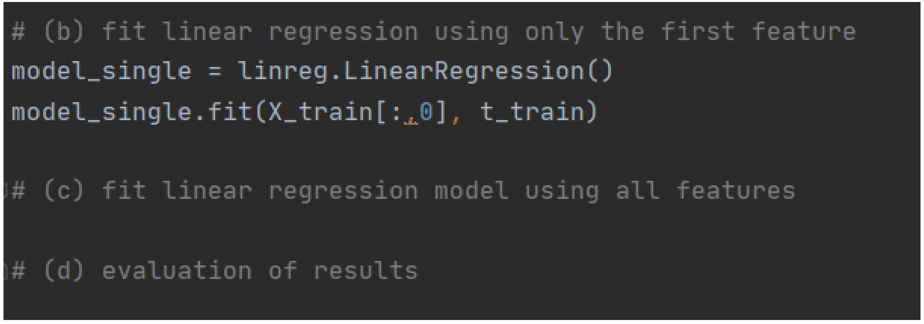
```
def fit(self, X, t):
    """
    Fits the linear regression model.

    Parameters
    ----------
    X : Array of shape [n_samples, n_features]
    t : Array of shape [n_samples, 1]
    """

    # TODO: YOUR CODE HERE
    X = numpy.array(X).reshape((len(X), -1))
    t = numpy.array(t).reshape((len(t), 1))

    # prepend a column of ones
    ones = numpy.ones((X.shape[0], 1))
    X = numpy.concatenate((ones, X), axis=1)

    # compute weights  (matrix inverse)
    self.w = numpy.linalg.pinv((numpy.dot(X.T, X)))
    self.w = numpy.dot(self.w, X.T)
    self.w = numpy.dot(self.w, t)
```



```
# (b) fit linear regression using only the first feature
model_single = linreg.LinearRegression()
model_single.fit(X_train[:,0], t_train)

# (c) fit linear regression model using all features

# (d) evaluation of results
```

Exercise 5 (Total Training Loss)

$$\mathcal{L} = (Xw - t)^{\intercal}(Xw - t)$$

$$\mathcal{L} = w^{\intercal}X^{\intercal}Xw - 2w^{\intercal}X^{\intercal}t + t^{\intercal}t$$

$$\frac{\partial \mathcal{L}}{\partial w} = 2X^{\intercal}Xw - 2X^{\intercal}t = 0$$

$$w = (X^{\intercal}X)^{-1}X^{\intercal}t$$

When having a big error, the optimal least squares will show a larger loss compared with the derived average loss, because it is squared.