Load Dataset
Load ***diabetes.csv*** to two separate numpy arrays **X1**, **y**
Load features into **X1** [Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age]
Load labels into **y** [Outcome 0/1]
Skip the first header row
X1 = (768, 8) # 8 columns with 768 rows
y = (768, 1) # 1 columns with _ rows

Normalize columns of X1 (Values should be between 0-1):

$$X1 = \frac{X1 - X1.min(axis{=}0)}{X1.ptp(axis{=}0)}$$

Augment X at 1st column with a column of 1s:
Add a new column of 1s at first column
X = *(768, 9) # 9 columns with 768 rows*

Shuffling
Randomize X and corresponding y

Train Test Split
X_train = first 80% of X
y_train = first 80% of y
X_test = rest X
y_test = rest y

 # X_train = (M, 9) # 9 columns with M rows
 # y_train = (M, 1) # 1 columns with M rows
 # X_test = (M', 9) # 9 columns with M' rows
 # y_test = (M', 1) # 1 columns with M' rows

Initialize θ:
lr = 0.01
loss_list = []
total_loss = 0.0
theta_old = new numpy array with 9 random values of shape (9, 1)
theta_new = None

Training Loop:
For e in iterations(100):

Calculate Predictions:
# matrix multiply X_train and theta_old
Z = matmul(X_train, theta_old) # Z should be of dimension (M, 1)
h = sigmoid(Z) # h should be of dimension (M, 1)

Calculate Loss values:
J_vect = -y_train*log(h) - (1-y_train)*log(1-h) # should be of dimension (M, 1)
J = J_vect.mean() # should be a single numeric value
total_loss += J
loss_list.append(total_loss)

Update θ with gradient:
grad = numpy.matmul(X_train^T, (h-y_train))
# grad should be of shape (9, 1) i.e. same as theta_old
theta_new = theta_old - lr * grad
# theta_new should be of shape (9, 1) i.e. same as theta_old

theta_old = theta_new

Repeat next iteration

Evaluation on Test Set:
# calculate predictions for X_test
Z = matmul(X_test, theta_old) # Z should be of dimension (M', 1)
h = sigmoid(Z) # h should be of dimension (M', 1)
h[h >= 0.5] = 1 # predict 1 where sigmoid value is greater than or equal to 0.5
h[h < 0.5] = 0 # predict 0 where sigmoid value is less than 0.5

y_test_predicted = h
Calculate the accuracy by comparing y_test and y_test_predicted
Print the accuracy (Test)

similarly calculate predictions for X_train too and report

Notes:

- Load csv into numpy array
- shuffle two numpy arrays together
- train test split
- Concatenate numpy arrays
- Sigmoid function
- Column wise normalize