```
Python 3.11.5 (main, Aug 24 2023, 15:09:45) [Clang 14.0.3 (clang-1403.0.22.14.1)]
Type 'copyright', 'credits' or 'license' for more information
IPython 8.16.0 -- An enhanced Interactive Python. Type '?' for help.
```

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

import pandas as pd

from featureNormalize import featureNormalize
from gradientDescentMulti import gradientDescentMulti
from normalEqn import normalEqn
```

In [ ]:
```
print('\n ------------------------- \n')
print('Loading data ...')

# Load Data
path = 'ex1data2.txt'
data = pd.read_csv(path, header=None, names=['HouseSize', 'NbOfBedrooms',
data.head()

# Résumé des données
data.describe()

# set X (training data) and y (target variable)
nbCol = data.shape[1]
X = data.iloc[:,0:nbCol-1]
y = data.iloc[:,nbCol-1:nbCol]

# convert from data frames to numpy arrays
X = np.array(X.values)
y = np.array(y.values)
m = X.shape[0]



# Print out some data points
print('\n ------------------------- \n')
print('First 10 examples from the dataset:')
print(np.column_stack( (X[:10], y[:10]) ))


# Scale features and set them to zero mean
print('\n ------------------------- \n')
print('Normalizing Features ...')

X, mu, sigma = featureNormalize(X)
print('[mu] [sigma]')
print(mu, sigma)

# Add intercept term to X
X = np.concatenate((np.ones((m, 1)), X), axis=1)
```

```
 ------------------------

Loading data ...

 ------------------------

First 10 examples from the dataset:
[[  2104      3 399900]
 [  1600      3 329900]
 [  2400      3 369000]
 [  1416      2 232000]
 [  3000      4 539900]
 [  1985      4 299900]
 [  1534      3 314900]
 [  1427      3 198999]
 [  1380      3 212000]
 [  1494      3 242500]]

 ------------------------

Normalizing Features ...
[mu] [sigma]
[2000.68085106    3.17021277] [7.86202619e+02 7.52842809e-01]
```

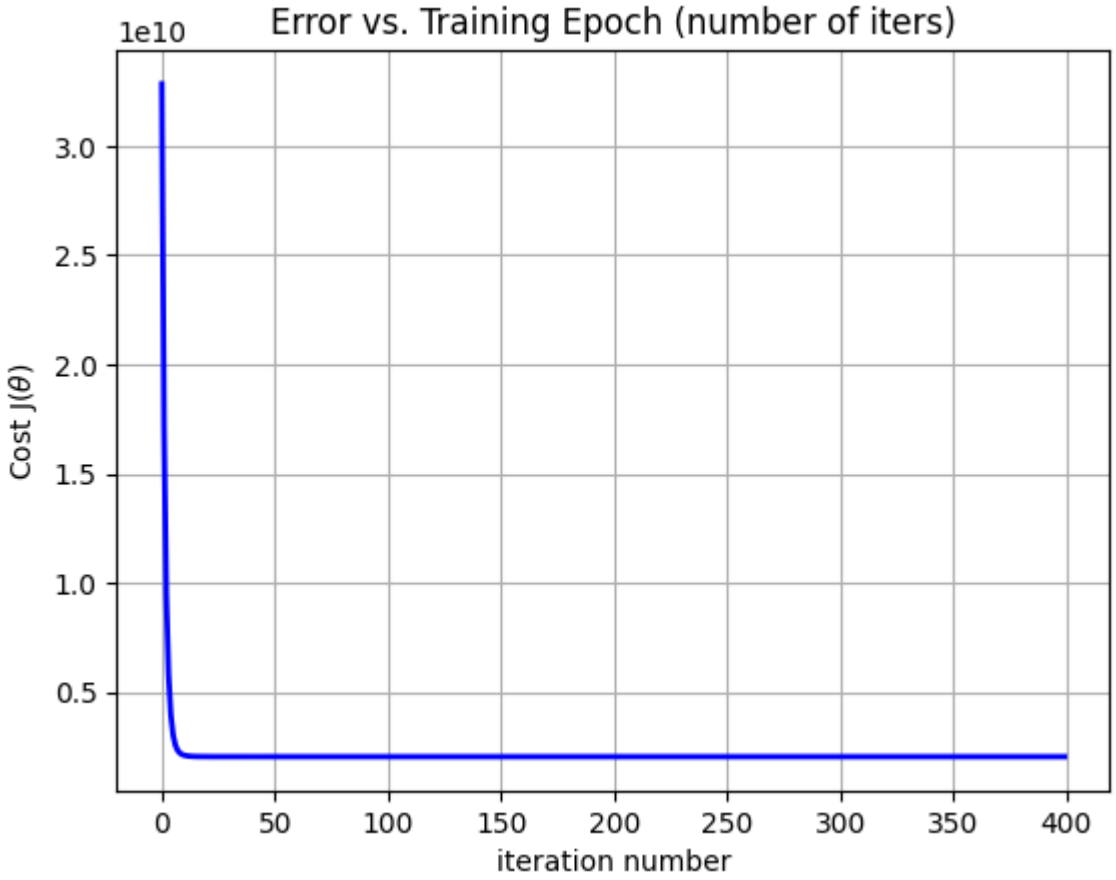
In [ ]:

```
#
# ====================== YOUR CODE HERE ======================
# Instructions: We have provided you with the following starter
#               code that runs gradient descent with a particular
#               learning rate (alpha).
#
#               Your task is to first make sure that your functions -
#               computeCost and gradientDescent already work with
#               this starter code and support multiple variables.
#
#               After that, try running gradient descent with
#               different values of alpha and see which one gives
#               you the best result.
#
#               Finally, you should complete the code at the end
#               to predict the price of a 1650 sq-ft, 3 br house.
#
#
# Hint: At prediction, make sure you do the same feature normalization.
#

print('\n ------------------------ \n')
print('Running gradient descent ...')

# Choose some alpha value
alpha = 0.3
num_iters = 400

# Init Theta and Run Gradient Descent
n = X.shape[1]
theta = np.zeros((n,1))
theta, cost_history, theta_history = gradientDescentMulti(X, y, theta, al

# Plot the convergence graph
fig = plt.figure()
ax = plt.gca()
```

```
ax.plot(np.arange(num_iters), cost_history, color="blue", linewidth=2.0,
ax.grid()
ax.set_xlabel('iteration number')
ax.set_ylabel(r'Cost J($\theta$)')
ax.set_title('Error vs. Training Epoch (number of iters)')
fig.show()


# Display gradient descent's result
print('\n ------------------------- \n')
print('Theta computed from gradient descent: ')
print(theta)

# Estimate the price of a 1650 sq-ft, 3 br house
new_house = np.array([[1650, 3]])
norm_newHouse = (new_house-mu)/sigma
norm_newHouse = np.concatenate((np.ones((1,1)),norm_newHouse), axis=1)
price = norm_newHouse.dot(theta)


# price = np.array(price).dot(theta)

print('\n ------------------------- \n')
print('Predicted price of a 1650 sq-ft, 3 br house')
print('(using gradient descent): ')
print(price)
```

```
 -------------------------

Running gradient descent ...

 -------------------------

Theta computed from gradient descent:
[[340412.65957447]
 [109447.79646964]
 [ -6578.35485416]]

 -------------------------

Predicted price of a 1650 sq-ft, 3 br house
(using gradient descent):
[[293081.4643349]]
```

```
<ipython-input-3-a28d9bb4cb52>:43: UserWarning: FigureCanvasAgg is non-int
eractive, and thus cannot be shown
  fig.show()
```



In [ ]:

```
# ====================== YOUR CODE HERE ======================
# Instructions: The following code computes the closed form
#               solution for linear regression using the normal
#               equations. You should complete the code in
#               normalEqn.py
#
#               After doing so, you should complete this code
#               to predict the price of a 1650 sq-ft, 3 br house.
#
print('\n --------------------------- \n')
print('Solving with normal equations...')

# Load Data
data = np.loadtxt('ex1data2.txt', delimiter=',')
path = 'ex1data2.txt'
data = pd.read_csv(path, header=None, names=['HouseSize', 'NbOfBedrooms',
data.head()


# set X (training data) and y (target variable)
nbCol = data.shape[1]
X = data.iloc[:,0:nbCol-1]
y = data.iloc[:,nbCol-1:nbCol]

# convert from data frames to numpy arrays
X = np.array(X.values)
y = np.array(y.values)
m = X.shape[0]


# Add intercept term to X
X = np.concatenate((np.ones((m, 1)), X), axis=1)
```

```
# Calculate the parameters from the normal equation
theta = normalEqn(X, y)

# Display normal equation's result
print('Theta computed from the normal equations:')
print(' %s \n' % theta)

# Estimate the price of a 1650 sq-ft, 3 br house
price = np.array([[1, 1650, 3 ]]).dot(theta)


print("Predicted price of a 1650 sq-ft, 3 br house ")
print('(using normal equations):\n $%f\n' % price)

# ================================================================
```

```
 ---------------------------
```

```
Solving with normal equations...
Theta computed from the normal equations:
 [[89597.9095428 ]
 [  139.21067402]
 [-8738.01911233]]

Predicted price of a 1650 sq-ft, 3 br house
(using normal equations):
 $293081.464335
```

```
<ipython-input-4-3e841cd356f3>:48: DeprecationWarning: Conversion of an ar
ray with ndim > 0 to a scalar is deprecated, and will error in future. Ens
ure you extract a single element from your array before performing this op
eration. (Deprecated NumPy 1.25.)
  print('(using normal equations):\n $%f\n' % price)
```