**Question 1**   
  
  
1. The following are the reasons that a linear regression model is appropriate to describe the relationship between

Column C "CO(GT)" & Column H "NOx(GT)"  
  
The relationship between the variables is linear because of the following reasons  
  
1. Establishing the dependent variable and independent variables

We know the exact values of X and Y even before we start the model  
  
2. Establishing the independent variable.

In the case of a multivariate linear regression, the explanatory variables have to be independent. In other words, we do not use collinear variables in the same model.  
Note: To check this, plot one variable against the other. If you detect a strong linear or nonlinear pattern, they are dependent.  
  
3. Establishing the homogeneity:

 We can plot the residuals of your model against the fitted values.  
  
4. The data is homoscedastic, meaning the variance in the residuals (the difference in the real and predicted values) is more or less constant  
  
5. The residuals are independent, meaning the residuals are distributed randomly and not influenced by the residuals in previous observations.   
   If the residuals are not independent of each other, they’re considered to be auto correlated.  
     
     
6. The residuals are normally distributed. This assumption means the probability  
density function of the residual values is normally distributed at each x value.  
  
  
**2. Comparison of functions from Pandas, Numpy V/S sklearn library**  
  
Using Pandas  
  
It is very easy to read the excel and perform manipulations using Pandas.  
We can drop the columns, reshape for arrays, analyze the data and can have quick peeks and trends of the data.  
It has very powerful functions to preprocess the data as required.  
Pandas library is good for analyzing tabular data. You can use it for exploratory data analysis, statistics, visualization.  
  
  
**Using Numpy**  
  
All mathematical formulae can be easily calculated using Numpy but they are not as good as Sklearn libraries as clearly seen in the differences below of Co-Efficients, RMSE and R2.

NumPy is a library for efficient array computations, modeled after Matlab.   
Arrays differ from plain Python lists in the way they are stored and handled.

Array elements stay together in memory, so they can be quickly accessed. NumPy also supports quick subindexing, e.g., a[0, :, 2] gives you all array elements whose first index is 0 and third index is 2.  
  
Furthermore, NumPy provides vectorized mathematical functions. When, e.g., you call numpy.sin(a), the sine function is applied  
on every element of array a.

**Using Numpy for Question 1**  
  
Time taken to calculate Co-Efficient ::  0.012964487075805664  
  
Time taken to calculate RMSE ::  0.012964010238647461  
  
Time taken to calculate RMSE ::  0.018987178802490234  
  
**Using sklearn for Question 1**  
  
Scikit-learn is a collection of advanced machine-learning algorithms for Python. It also is built upon Numpy and SciPy.  
  
As demonstrated in the python solution the algorithms are super-fast in establishing linear regression.  
  
Time taken to calculate Coefficients ::  0.0009975433349609375  
Time taken to calculate RMSE ::  0.001971006393432617  
Time taken to calculate R2 ::  0.000997781753540039

**Question 2**

1. See the juniper note book attached.
2. The combination of attributes that suggest good air quality

'NOx(GT)'

'NO2(GT)'

'T'

'RH'

'AH'

Along with the following dummy variables

'dayofweek','hour','month', 'day'

produce the best linear regression model.

**The following are the metrics**

('Mean Absolute Error:', 0.303789041330613)

('Mean Squared Error:', 0.16406585765144469)

('Root Mean Squared Error:', 0.40505043840421245)

('R2(R Squared):'0.9230678989627656)

**The Co-Efficients obtained are as follows:**

NOx(GT) 1.388501e-02

NO2(GT) 6.627762e-03

T -3.552464e-02

RH -1.802497e-02

AH 1.095408e+00

dayofweek 9.570151e-03

hour 6.938894e-17

month 1.064444e-01

day 3.850905e-03

1. The following algorithm is the best

Mini-Batch Gradient Descent

* It is splits the training data-set into small batches that are used to calculate model error and update model coefficients.
* The batched updates provide a computationally more efficient process than stochastic gradient descent.
* Error information must be accumulated across mini-batches of training examples like batch gradient descent.

Mini-batch gradient descent is the recommended variant of gradient descent for most applications, especially in deep learning.

Mini-batch sizes, commonly called “batch sizes” for brevity, are often tuned to an aspect of the computational architecture on which the implementation is being executed.

[Batch size](https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/) is a slider on the learning process

* Small values give a learning process that converges quickly at the cost of noise in the training process.
* Large values give a learning process that converges slowly with accurate estimates of the error gradient.