In [1]:
```
import torch
import numpy as np
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
from torch.autograd import Variable
import pandas as pd
```

In [2]:
```
class MyDataset(Dataset):
    # Initialize your data, download, etc.
    def __init__(self, file_path):
     # download, read data
        data = pd.read_csv(file_path)
        self.y_data = torch.from_numpy(data['y'].values).unsqueeze(dim=1).float()
        self.x_data = torch.from_numpy(data['x'].values).unsqueeze(dim=1).float()

    def __len__(self):
        # return the data length
        return len(self.x_data)

    def __getitem__(self, idx):
        # return one item on the index
        x = self.x_data[idx]
        y = self.y_data[idx]
        return x, y
dataset = MyDataset('./train.csv')
len(dataset)
```

Out[2]: 700

In [3]:
```
# Model
class LinearRegressionModel(nn.Module): # torch.nn.Module을 상속받는 파이썬 클래스
    def __init__(self): #
        super().__init__()
        self.linear = nn.Linear(1, 1) # 단순선형회귀이므로 input_dim=1, output_dim=1.

    def forward(self, x):
        return self.linear(x)
```

In [8]:
```
# Hyperparameters
batch_size = 100
lr = 0.01
epochs = 500
num_workers = 0
dataset = MyDataset('./train.csv')

train_loader = DataLoader(dataset = dataset,
                          batch_size = batch_size,
                          shuffle = True,
                          num_workers = num_workers)
model = LinearRegressionModel()
optimizer = torch.optim.SGD(model.parameters(), lr = lr)
criterion = nn.MSELoss()
```

In [1]:
```
for epoch in range(epochs):
    for i, data in enumerate(train_loader):
        get data
        x, y = data
        x, y = Variable(x), Variable(y)

        # forward pass
        y_pred = model(x)

        # compute loss
        loss = criterion(y_pred, y)

        if epoch % 10 == 0:
        # 100번마다 로그 출력
          print('Epoch: {:4d}/{}    loss: {:.6f}'.format(
              epoch, epochs, loss.item()
          ))
        # Zero gradients, perform a backward pass, and update the weights.
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

In [11]:
```
dataset = MyDataset('./train.csv')
dataset[1]
```

Out[11]: (tensor([50.]), tensor([47.4645]))