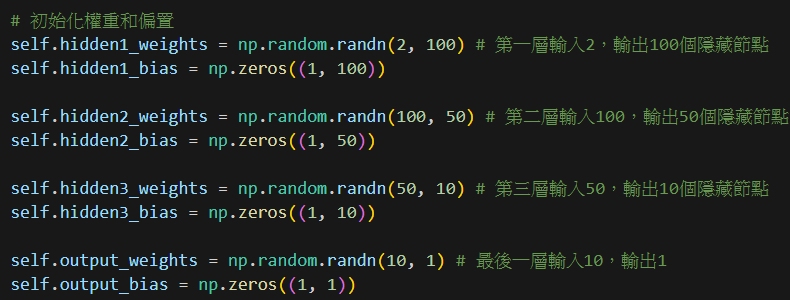
**DL Lab4: forward pass & back-propagation**

(1) Implement a simple neural network with three hidden layers (with 100 nodes in hidden layer 1 , 50 nodes in hidden layer 2 and 10 nodes in hidden layer 3).

由於我底下的forward寫法是用np.dot，所以這邊第一層hidden layer要設置成2x100，矩陣的長寬才會對應。

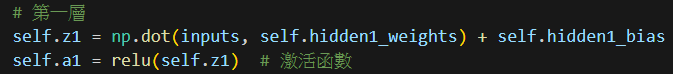


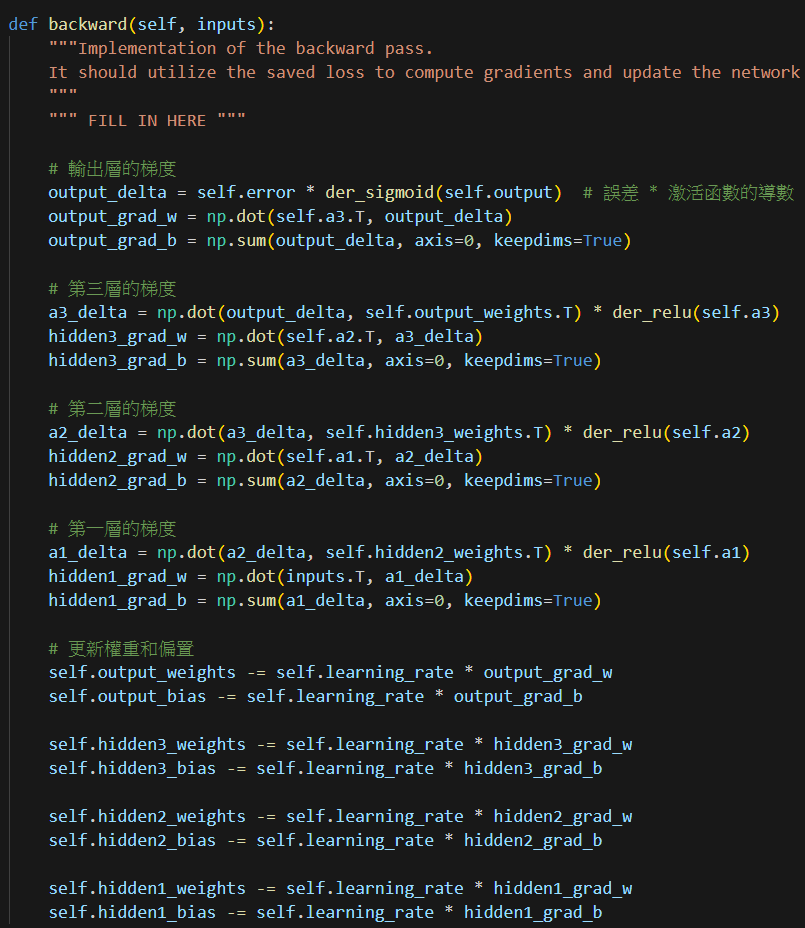
(2) The number of data points generated should > 100, and > 1000 for chessboard.

在Linear、XOR以及Chessboard這三種型態的資料，我分別生成了1000、1000和2000筆，我發現由於Chessboard的資料比較分散，對於神經網路來說比較難學習，因此他train所需要的資料量也會比較多。



(3) You must use the back-propagation algorithm in this NN and build it from scratch. Only Numpy and other Python standard libraries are allowed.

為了使模型更加靈活，我在forward那邊除了原本的矩陣相乘之外，我還加入了偏置矩陣，避免矩陣乘完之後的輸出為0，通過activation function後結果還是0，這樣會導致網路整體的學習率降低，因為運用到的神經元會比較少，因此加入偏置矩陣來使一些很小或者為0的特徵可以正常運作。

因此我在backward的時候，除了每一層的權重梯度之外，我還有運算了每一層的偏置梯度，最後再把梯度乘上學習率並更新權重和偏置。

(4) Learning Rate Schedules and Adaptive Learning Rate Methods.

在學習率方面，我一開始是使用0.001當作固定的學習率，不過這樣會使模型的參數最後很難收斂，因此我引入了學習率下降的機制，我嘗試過每100個epoch下降為0.99倍，可是我發現這種固定倍率下降的方式對於我們網路的效果並不好，最後我想到使用Exponential的方式去做下降，在Jupyter notebook檔案內可以看到我把學習率用圖表的方式展示出來，並且模型的準確度提升蠻明顯的。

