Question 3
What happens if the Sigmoid function has too steep of a slope transitioning from 0 to 1?

The steepness of a Sigmoid function from 0 to 1 can also be defined as the each of the derivative number of the function. Which is sigmoidSlope() in the code. If the Sigmoid function is too steep between 0 and 1 it also means that the output of function sigmoidSlope will be large.

According to this logic, the more steeper the function, the bigger the value is given by sigmoidSlope(). This can be elaborated based on question 2. It is already given from the contents of question 2 that changing the slope can change the number of steps in trainingNumber. Therefore, it can be easily inferred that if steepness of a slope transitioning from 0 to 1 is steep, the number of steps in traingNumber will be reduced.

This is because according to code :

outputDelta = outputError * sigmoidSlope(outputPerceptronLayer)

The components(outputDelta) which change the weights01 (Backpropagation), is influenced by the function of sigmoidSlope. Not only that, outputPerceptronLayer is also influenced by sigmoid function. Therefore, if the slope of the sigmoid function is steep the value of outputDelta will increase and the weights01will be updated with a bigger range.

Moreover, sigmoidSlope is somewhat similar to learning rate, which is an hyperparameter for the model, control how quickly the model is adapted to the problem. Smaller learning rates will require more training iterations due to smaller changes made to the weights each update, whereas larger learning rates allow rapid changes, which requires fewer iterations. For this case, if the slope is "too" steep, which is in the case of fast learning rate, only require few iterations. However, due to the huge range of changes it makes, can cause the model hard to find the local minimum of the cost, and converge to a suboptimal solution.
To text this, I have changed the coefficient of the sigmoid function as -100 instead of -1, and try running the program. The result was, program unable to give out value due to searching. This partly proves that a steepness too steep will be hard for the program to find the right training iterations.

For more information, in case of slow learning rate, there is a chance of process getting stuck and not able to learn.

Question 4
What happens when you choose different trainingNumber and output?

To begin with, trainingNumber is iteration of neuron learning through the process of propagation and backpropagation. It can be easily inferred from the code, that until a certain number, the bigger the trainingNumber, the outputPerceptronLayer is similar with the reference output.

Therefore, if a different trainingNumber is chosen, the learning progress of the model will be processed just the amount of the number of trainingNumber. However, just like there are some exceptions for every system, in some cases even if the trainingNumber is changed upward, some might cause no difference. In case of the inputdata and the referenceOutput ([[0],[1],[1],[0]]) given on Exercise 4, the outputPerceptronLayer value converge to ([[0.5],[0.5],[0.5],[0.5]) instead of getting closer to the referenceOutput. Furthermore, it is stated from the professor that at a more complex neural network, the error can become larger even after a certain point at some circumstances. However, this is a simple neural network with only one neuron. Therefore, not much of a possibility to happen.

For the case of changing reference output, in the case of a single neuron neural network, is basically making a whole new model teaching neuron a whole new different information.

For example, the code use to teach the neuron that the inputData ([[0,0,1],[0,1,1],[1,0,1],[1,1,1]]) should output the referenceoutput ([[0][0][1][1]]). However, if the referenceoutput is changed to ([[0][1][1][0]]), it will be a whole new model teaching that the inputData should output ([[0][1][1][0]]). However, in the case of ([[0][1][1][0]]), the model gives out the value ([[0.5][0.5][0.5][0.5]]) instead of ([[0][1][1][0]]). It can be inferred that due to the simple structure of this particular mechanism, the neuron was not able to learn even with lot of process. This can be solved by adding layers in between the input and outputPerceptronLayer to make a more complex model.