Assignment 3

```
In [1]: import numpy as np

        # Define input data and target values
        input_data = np.array([[1,1,1,0,1,1,1],   # 0
                               [0,0,1,0,0,1,0],   # 1
                               [1,0,1,1,1,0,1],   # 2
                               [1,0,1,1,0,1,1],   # 3
                               [0,1,1,1,0,1,0],   # 4
                               [1,1,0,1,0,1,1],   # 5
                               [1,1,0,1,1,1,1],   # 6
                               [1,0,1,0,0,1,0],   # 7
                               [1,1,1,1,1,1,1],   # 8
                               [1,1,1,1,0,1,1]])  # 9


        target_values = np.array([[1,0],   # Even
                                  [0,1],   # Odd
                                  [1,0],   # Even
                                  [0,1],   # Odd
                                  [1,0],   # Even
                                  [0,1],   # Odd
                                  [1,0],   # Even
                                  [0,1],   # Odd
                                  [1,0],   # Even
                                  [0,1]])  # Odd
```

```
In [2]: # Define the Perceptron Neural Network class
        class PerceptronNN:
            def __init__(self, num_inputs, num_outputs):
                self.weights = np.zeros((num_inputs, num_outputs))
                self.bias = np.zeros(num_outputs)

            def activation(self, x):
                return 1 if x >= 0 else 0

            def feedforward(self, inputs):
                net = np.dot(inputs, self.weights) + self.bias
                output = np.array([self.activation(x) for x in net])
                return output

            def train(self, input_data, target_values, learning_rate, epochs):
                for epoch in range(epochs):
                    errors = 0
                    for i, input in enumerate(input_data):
                        target = target_values[i]
                        output = self.feedforward(input)
                        error = target - output
                        self.weights += learning_rate * np.outer(input, error)
                        self.bias += learning_rate * error
                        errors += np.abs(error).sum()
```

```python
# Train the Perceptron Neural Network
perceptron_nn = PerceptronNN(num_inputs=7, num_outputs=2)
perceptron_nn.train(input_data, target_values, learning_rate=0.1, epochs=100)

# Test the Perceptron Neural Network on some sample inputs
input_test = np.array([[1,1,0,1,0,1,1], # 5
                       [1,1,1,0,1,1,1], # 0
                       [1,1,1,1,1,1,1]]) # 8

for input in input_test:
    output = perceptron_nn.feedforward(input)
    if output[0] == 1:
        print("Even")
    else:
        print("Odd")
```

```
Odd
Even
Even
```