Experiment 7,8: To implement perceptron and backpropogation on non-binary inputs.

```
import random
import math


class Perceptron:
  def __init__(self, inputs: list, output: int):
    super(Perceptron, self).__init__()
    self.inputs = inputs
    self.output = output
    self.weights = [random.randrange(1) for i in range(len(inputs))]

  def forward_backward_pass(self):
    weighted_sum = sum([x*y for x,y in zip(self.inputs, self.weights)])
    # activation = 0 if weighted_sum < 0 else 1
    activation = 1 / 1 + math.exp(-weighted_sum)
    output_x = activation
    error = 1/2 * (self.output - output_x) ** 2

    # Backward Pass (Backpropogation)
    new_weights = []
    delta_w1 = (output_x * self.output) * \
      activation * (1 - activation) * self.inputs[0]
    delta_w2 = (output_x * self.output) * \
      activation * (1 - activation) * self.inputs[1]
    new_w1 = self.weights[0] - delta_w1
    new_w2 = self.weights[1] - delta_w2
    new_weights.append(new_w1)
    new_weights.append(new_w2)
    new_weighted_sum = sum([x*y for x,y in zip(self.inputs, new_weights)])

    new_activation = 1 / 1 + math.exp(-new_weighted_sum)
    new_output_x = new_activation
    new_error = 1/2 * (self.output - new_output_x) ** 2
    return error, new_error


inputs = [0.3, 0.7]
output = 1

error_fp, error_bp = Perceptron(inputs, output).forward_backward_pass()


print(f"Forward pass error is: {error_fp}")
print(f"After Backpropagation error is: {error_bp}")
```

```
Forward pass error is: 0.5
After Backpropagation error is: 0.004828848813768884
```