```
#Perceptron for AND gate
import numpy as np

def activation_function(x):
  """
  Step function as activation function.
  """
  return 1 if x > 0 else 0

def perceptron_and(x1, x2, weights, bias):
  """
  Perceptron implementation for AND logic gate.
  """
  weighted_sum = x1 * weights[0] + x2 * weights[1] + bias
  return activation_function(weighted_sum)

# Define weights and bias for AND gate
weights = [1, 2]
bias = -2

# Test the AND gate with all possible inputs
for x1 in [0, 1]:
  for x2 in [0, 1]:
    output = perceptron_and(x1, x2, weights, bias)
    print(f"Input: {x1}, {x2}, Output: {output}")
```

```
Input: 0, 0, Output: 0
Input: 0, 1, Output: 0
Input: 1, 0, Output: 0
Input: 1, 1, Output: 1
```

```
#Perceptron for OR gate
def activation_function(x):
  """
  Step function as activation function.
  """
  return 1 if x > 0 else 0

def perceptron_or(x1, x2, weights, bias):
  """
  Perceptron implementation for OR logic gate.
  """
  weighted_sum = x1 * weights[0] + x2 * weights[1] + bias
  return activation_function(weighted_sum)

# Define weights and bias for OR gate
weights = [1, 1]
bias = -0.5

# Test the OR gate with all possible inputs
for x1 in [0, 1]:
  for x2 in [0, 1]:
    output = perceptron_or(x1, x2, weights, bias)
    print(f"Input: {x1}, {x2}, Output: {output}")
```

```
Input: 0, 0, Output: 0
Input: 0, 1, Output: 1
Input: 1, 0, Output: 1
Input: 1, 1, Output: 1
```

```
#Perceptron for NOT gate
def activation_function(x):
  """
  Step function as activation function.
  """
  return 1 if x > 0 else 0

def perceptron_not(x, weight, bias):
  """
  Perceptron implementation for NOT logic gate.
  """
  weighted_sum = x * weight + bias
  return activation_function(weighted_sum)

# Define weight and bias for NOT gate
weight = -1
bias = 0.5

# Test the NOT gate with all possible inputs
for x in [0, 1]:
  output = perceptron_not(x, weight, bias)
  print(f"Input: {x}, Output: {output}")
```

```
Input: 0, Output: 1
Input: 1, Output: 0
```

Start coding or generate with AI.