Function

In [1]:
```
def McCulloch_Pitts_ANDNOT(input1, input2, w1, w2, threshold):
    # Calculate the weighted sum
    weighted_sum = (w1 * input1) + (w2 * input2)
    # Check if the sum is greater than or equal to the threshold
    if weighted_sum >= threshold:
        return 1
    else:
        return 0
```

Define initial parameters

In [2]:
```
w1 = 0.5
w2 = -0.5
threshold = -0.5
learning_rate = 0.1
```

Define target outputs for ANDNOT function

In [3]:
```
target_outputs = [0, 0, 1, 0]
```

Training loop

In [4]:
```
epochs = 1000
for epoch in range(epochs):
    # Iterate through each input combination
    for inputs, target_output in zip([(0, 0), (0, 1), (1, 0), (1, 1)], targ
et_outputs):
        input1, input2 = inputs

        # Calculate the actual output
        output = McCulloch_Pitts_ANDNOT(input1, input2, w1, w2, threshold)

        # Calculate the error
        error = target_output - output

        # Update parameters using gradient descent
        w1 += learning_rate * error * input1
        w2 += learning_rate * error * input2
        threshold -= learning_rate * error
```

Test the updated neuron

In [5]:
```
print("ANDNOT(0, 0) =", McCulloch_Pitts_ANDNOT(0, 0, w1, w2, threshold))
print("ANDNOT(0, 1) =", McCulloch_Pitts_ANDNOT(0, 1, w1, w2, threshold))
print("ANDNOT(1, 0) =", McCulloch_Pitts_ANDNOT(1, 0, w1, w2, threshold))
print("ANDNOT(1, 1) =", McCulloch_Pitts_ANDNOT(1, 1, w1, w2, threshold))
print(f"Weights:w1 = {w1}\n\tw2 = {w2}\nThreshold : {threshold}")
```

```
ANDNOT(0, 0) = 0
ANDNOT(0, 1) = 0
ANDNOT(1, 0) = 1
ANDNOT(1, 1) = 0
Weights:w1 = 0.30000000000000004
        w2 = -0.7
Threshold : 0.09999999999999998
```

In [ ]: