# PERCEPTRON OR PROBLEM(FROM SCRATCH)

## Aim

To demonstrate the training of a perceptron to simulate the behavior of an OR operation

## Problem description

The OR operation is a fundamental logic gate that takes two binary inputs and produces an output based on the following rule: the output is true (1) only when any of these inputs are true(1) otherwise the output is false(0).

| INPUT A | INPUT B | OUTPUT |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

The problem is to train the perceptron model to learn the behavior of OR gate based on the provided INPUT -OUTPUT pair. Then validate its performance by checking whether it correctly predicts the OUTPUT.

## Algorithm

1. Initialize Variables:
   - Set weights w to initial values [0.0, 0.3].
   - Set the threshold to 0.4.
   - Set learning rate learning_rate to 0.5.
   - Define input values a and b as well as actual output values y.

2. Perceptron Training Loop:
   - Iterate over the training examples (0 to 3).

- For each iteration:
  - Calculate the weighted sum summation of inputs and weights.
  - Apply the activation function using the threshold.
  - Print input, weights, summation, and actual vs. predicted output.
  - If the predicted output is different from the actual output:
    - Update the weights using the perceptron learning rule.
    - Reset the iteration index to start training again.

3. Perceptron Training Output:
   - Display the final weights after training.

4. Perceptron Prediction:
   - For a new input or_input, calculate the weighted sum.
   - Return the predicted output using the activation function.

5. Output:
   - Print the predicted output for the given input [0, 0].

## Program code/ Pseudocode

```
def activation(out,threshold):
    if out > threshold:
        return 1
    else:
        return 0

def perceptron(and_input):
    a = [0,0,1,1]
    b = [0,1,0,1]
    y = [0,1,1,1] # Actual Output
    w = [0.0,0.3]
    threshold = 0.4
```

```
    learning_rate = 0.5
    i=0
    print("Perceptron Training : ")

    print("-----------------")
    while i<4:
        summation = a[i]*w[0] + b[i]*w[1]
        o = activation(summation,threshold)
        print("Input : " + str(a[i]) +" , "+ str(b[i]))
        print("Weights : " + str(w[0]) +" , "+ str(w[1]))
        print("summation : "+str(summation) + " threshold : "+str(threshold) )
        print("Actual Output : "+str(y[i])+" Predicated Output : "+str(o))
        if(o!=y[i]):
            # w = w + learning_rate(actual_output - predicated_output)*input
            print("_______\nUpdating Weights")
            w[0]=w[0]+learning_rate*(y[i]-o)*a[i]
            w[1]=w[1]+learning_rate*(y[i]-o)*b[i]
            print("Updated Weights : " + str(w[0]) +" , "+ str(w[1]))
            i = -1
            print("\nWeights Updated Training Again : ")

        i=i+1
        print("------------------")
    # Prediction Part
    summation = and_input[0]*w[0] + and_input[1]*w[1]
    return activation(summation,threshold)


or_input = [0,0]
print("OR GAte Output For "+str(or_input) + " : " + str(perceptron(or_input)))
```

## Result

The trained perceptron model correctly predicts the OR gate OUTPUT of various combinations of INPUT.