```
# -*- coding: utf-8 -*-
"""
Created on Thu Mar  9 16:46:38 2023

@author: tiver
"""

def hebbian_learning(samples):
     print(f'{"INPUT":^8} {"TARGET":^16}{"WEIGHT CHANGES":^15}{"WEIGHTS":^25}')
     w1, w2, b = 1, 1, 1
     print(' ' * 45, f'({w1:2}, {w2:2}, {b:2})')
     for x1, x2, y in samples:
         w1 = w1 + x1 * y
         w2 = w2 + x2 * y
         b = b + y
         print(f'({x1:2}, {x2:2}) {y:2} ({x1*y:2}, {x2*y:2}, {y:2}) ({w1:2}, {w2:2}, {b:2})')
AND_samples = {
    'binary_input_binary_output': [
        [1, 1, 1],
        [1, 0, 0],
        [0, 1, 0],
        [0, 0, 0]
    ],
    'binary_input_bipolar_output': [
        [1, 1, 1],
        [1, 0, -1],
        [0, 1, -1],
        [0, 0, -1]
    ],
    'bipolar_input_bipolar_output': [
        [ 1, 1, 1],
        [ 1, -1, -1],
        [-1, 1, -1],
        [-1, -1, -1]
    ]
}
OR_samples = {
    'binary_input_binary_output': [
        [1, 1, 1],
        [1, 0, 1],
        [0, 1, 1],
        [0, 0, 0]
    ],
    'binary_input_bipolar_output': [
        [1, 1, 1],
        [1, 0, 1],
        [0, 1, 1],
        [0, 0, -1]
    ],
    'bipolar_input_bipolar_output': [
        [ 1, 1, 1],
        [ 1, -1, 1],
        [-1, 1, 1],
        [-1, -1, -1]
    ]
}
XOR_samples = {
    'binary_input_binary_output': [
        [1, 1, 0],
        [1, 0, 1],
        [0, 1, 1],
        [0, 0, 0]
    ],
    'binary_input_bipolar_output': [
        [1, 1, -1],
        [1, 0, 1],
        [0, 1, 1],
        [0, 0, -1]
    ],
    'bipolar_input_bipolar_output': [
        [ 1, 1, -1],
        [ 1, -1, 1],
        [-1, 1, 1],
        [-1, -1, -1]
    ]
}

#For AND gate
#print('-'*20, 'HEBBIAN LEARNING', '-'*20)
#print('AND with Binary Input and Binary Output')
#hebbian_learning(AND_samples['binary_input_binary_output'])
```

```
#print('AND with Binary Input and Bipolar Output')
#hebbian_learning(AND_samples['binary_input_bipolar_output'])
#print('AND with Bipolar Input and Bipolar Output')
#hebbian_learning(AND_samples['bipolar_input_bipolar_output'])

# #OR Gate
print('-'*20, 'HEBBIAN LEARNING', '-'*20)
print('OR with binary input and binary output')
hebbian_learning(OR_samples['binary_input_binary_output'])
print('OR with binary input and bipolar output')
hebbian_learning(OR_samples['binary_input_bipolar_output'])
print('OR with bipolar input and bipolar output')
hebbian_learning(OR_samples['bipolar_input_bipolar_output'])


# #XOR Gate
# print('-'*20, 'HEBBIAN LEARNING', '-'*20)
# print('XOR with binary input and binary output')
# hebbian_learning(XOR_samples['binary_input_binary_output'])
# print('XOR with binary input and bipolar output')
# hebbian_learning(XOR_samples['binary_input_bipolar_output'])
# print('XOR with bipolar input and bipolar output')
# hebbian_learning(XOR_samples['bipolar_input_bipolar_output'])
```

```
-------------------- HEBBIAN LEARNING --------------------
OR with binary input and binary output
 INPUT        TARGET     WEIGHT CHANGES          WEIGHTS
                                              ( 1,  1,  1)
( 1,  1)  1 ( 1,  1,  1) ( 2,  2,  2)
( 1,  0)  1 ( 1,  0,  1) ( 3,  2,  3)
( 0,  1)  1 ( 0,  1,  1) ( 3,  3,  4)
( 0,  0)  0 ( 0,  0,  0) ( 3,  3,  4)
OR with binary input and bipolar output
 INPUT        TARGET     WEIGHT CHANGES          WEIGHTS
                                              ( 1,  1,  1)
( 1,  1)  1 ( 1,  1,  1) ( 2,  2,  2)
( 1,  0)  1 ( 1,  0,  1) ( 3,  2,  3)
( 0,  1)  1 ( 0,  1,  1) ( 3,  3,  4)
( 0,  0) -1 ( 0,  0, -1) ( 3,  3,  3)
OR with bipolar input and bipolar output
 INPUT        TARGET     WEIGHT CHANGES          WEIGHTS
                                              ( 1,  1,  1)
( 1,  1)  1 ( 1,  1,  1) ( 2,  2,  2)
( 1, -1)  1 ( 1, -1,  1) ( 3,  1,  3)
(-1,  1)  1 (-1,  1,  1) ( 2,  2,  4)
(-1, -1) -1 ( 1,  1, -1) ( 3,  3,  3)
```