```
import numpy as np
```

```
X = np.array([[0, 0],
              [0, 1],
              [1, 0],
              [1, 1]])
```

```
Y = np.array([0, 0, 0, 1])
```

```
w = np.array([0.3, -0.2])
theta=0.4
lr=0.2
epochs=4
print(f"Initial weights:{w},theta={theta},lr={lr}\n")
```

```
Initial weights:[ 0.3 -0.2],theta=0.4,lr=0.2
```

```
for epoch in range(1, epochs+1):
  print(f"---Epoch {epoch} ---")
  print(f"x1\tx2\ty_des\ty_est\terror\tw1\tw2")
  error_count = 0
  for i in range(len(X)):
    x=X[i]
    y_des=Y[i]

    net = np.dot(x, w) + theta
    y_est=1 if net>=0 else 0
    error = y_des-y_est
    if error != 0:
      w = w + lr * error * x
      error_count += 1
    print(f"{x[0]}\t{x[1]}\t{y_des}\t{y_est}\t{error}\t{w[0]:.1f}\t{w[1]:.1f}")

  if error_count == 0:
    print("\nTraining complete!")
    break
```

```
---Epoch 1 ---
x1      x2      y_des   y_est   error   w1      w2
0       0       0       1       -1      0.3     -0.2
0       1       0       1       -1      0.3     -0.4
1       0       0       1       -1      0.1     -0.4
1       1       1       1       0       0.1     -0.4
---Epoch 2 ---
x1      x2      y_des   y_est   error   w1      w2
0       0       0       1       -1      0.1     -0.4
0       1       0       1       -1      0.1     -0.6
1       0       0       1       -1      -0.1    -0.6
1       1       1       0       1       0.1     -0.4
---Epoch 3 ---
x1      x2      y_des   y_est   error   w1      w2
0       0       0       1       -1      0.1     -0.4
0       1       0       0       0       0.1     -0.4
1       0       0       1       -1      -0.1    -0.4
1       1       1       0       1       0.1     -0.2
---Epoch 4 ---
x1      x2      y_des   y_est   error   w1      w2
0       0       0       1       -1      0.1     -0.2
0       1       0       1       -1      0.1     -0.4
1       0       0       1       -1      -0.1    -0.4
1       1       1       0       1       0.1     -0.2
```

```
print("\nFinal perceptron outputs:")
for i in range(len(X)):
  x=X[i]
  net = np.dot(x, w) + theta
  y_est=1 if net>=0 else 0
  print(f"Input {x}->Net={round(net,1)},Yest={y_est},Ydes={Y[i]}")
```

```
Final perceptron outputs:
Input [0 0]->Net=0.4,Yest=1,Ydes=0
Input [0 1]->Net=0.2,Yest=1,Ydes=0
Input [1 0]->Net=0.5,Yest=1,Ydes=0
Input [1 1]->Net=0.3,Yest=1,Ydes=1
```