```
import numpy as np

# Load patterns
x1= np.array([ [ -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],[ -1, -1,
-1, 1, 1, 1,
    1, -1, -1, -1],[ -1, -1, 1, 1, 1, 1, 1, 1, -1, -1],[ -1, 1, 1,
1, -1, -1, 1,
        1, 1, -1],[ -1, 1, 1, 1, -1, -1, 1, 1, 1, -1],[ -1, 1, 1, 1,
-1, -1, 1,
            1, 1, -1],[ -1, 1, 1, 1, -1, -1, 1, 1, 1, -1],[ -1, 1,
1, 1, -1, -1,
                1, 1, 1, -1],[ -1, 1, 1, 1, -1, -1, 1, 1, 1, -1],
[ -1, 1, 1, 1,
                    -1, -1, 1, 1, 1, -1],[ -1, 1, 1, 1, -1, -1, 1,
1, 1, -1],[
                        -1, 1, 1, 1, -1, -1, 1, 1, 1, -1],[ -1, 1,
1, 1, -1, -1,
                            1, 1, 1, -1],[ -1, -1, 1, 1, 1, 1, 1, 1,
-1, -1],[
                                -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],
[ -1, -1,
                                    -1, -1, -1, -1, -1, -1, -1,
-1] ])
x2=np.array([ [ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1,
1, 1, 1, -1, -1,
    -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1,
1, -1, -1,
        -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1,
1, 1, 1, -1,
            -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1,
-1, 1, 1, 1,
                1, -1, -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1,
-1],[ -1, -1,
                    -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1, -1, 1, 1,
1, 1, -1,
                        -1, -1],[ -1, -1, -1, 1, 1, 1, 1, -1, -1,
-1],[ -1, -1,
                            -1, 1, 1, 1, 1, -1, -1, -1],[ -1, -1,
-1, 1, 1, 1,
                                1, -1, -1, -1],[ -1, -1, -1, 1, 1,
1, 1, -1, -1,
                                    -1],[ -1, -1, -1, 1, 1, 1, 1,
-1, -1, -1] ])
x3=np.array([ [ 1, 1, 1, 1, 1, 1, 1, 1, -1, -1],[ 1, 1, 1, 1, 1, 1,
1, 1, -1, -1],[ -1,
    -1, -1, -1, -1, 1, 1, 1, -1, -1],[ -1, -1, -1, -1, -1, 1, 1, 1,
-1, -1],[
        -1, -1, -1, -1, -1, 1, 1, 1, -1, -1],[ -1, -1, -1, -1, -1,
1, 1, 1, -1,
            -1],[ -1, -1, -1, -1, -1, 1, 1, 1, -1, -1],[ 1, 1, 1, 1,
1, 1, 1, 1,
                -1, -1],[ 1, 1, 1, 1, 1, 1, 1, 1, -1, -1],[ 1, 1, 1,
-1, -1, -1,
                    -1, -1, -1, -1],[ 1, 1, 1, -1, -1, -1, -1, -1,
```

```
-1, -1],[ 1,
                        1, 1, -1, -1, -1, -1, -1, -1, -1],[ 1, 1, 1,
-1, -1, -1,
                            -1, -1, -1, -1],[ 1, 1, 1, -1, -1, -1,
-1, -1, -1,
                                -1],[ 1, 1, 1, 1, 1, 1, 1, 1, -1,
-1],[ 1, 1, 1,
                                    1, 1, 1, 1, 1, -1, -1] ])
x4= np.array([ [ -1, -1, 1, 1, 1, 1, 1, 1, -1, -1],[ -1, -1, 1, 1,
1, 1, 1, 1, 1, -1],[
    -1, -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1,
1, 1, 1,
        -1],[ -1, -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1,
-1, -1, 1,
            1, 1, -1],[ -1, -1, -1, -1, -1, -1, 1, 1, 1, -1],[ -1,
-1, 1, 1, 1,
                1, 1, 1, -1, -1],[ -1, -1, 1, 1, 1, 1, 1, 1, -1,
-1],[ -1, -1,
                    -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1, -1, -1,
-1, -1, 1, 1,
                        1, -1],[ -1, -1, -1, -1, -1, -1, 1, 1, 1,
-1],[ -1, -1,
                            -1, -1, -1, -1, 1, 1, 1, -1],[ -1, -1,
-1, -1, -1,
                                -1, 1, 1, 1, -1],[ -1, -1, 1, 1, 1,
1, 1, 1, 1,
                                    -1],[ -1, -1, 1, 1, 1, 1, 1, 1,
-1, -1] ])
x5= np.array([ [ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1],[ -1, 1, 1, -1,
-1, -1, -1, 1, 1,
    -1],[ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1],[ -1, 1, 1, -1, -1,
-1, -1, 1, 1,
        -1],[ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1],[ -1, 1, 1, -1,
-1, -1, -1, 1,
            1, -1],[ -1, 1, 1, -1, -1, -1, -1, 1, 1, -1],[ -1, 1, 1,
1, 1, 1, 1,
                1, 1, -1],[ -1, 1, 1, 1, 1, 1, 1, 1, 1, -1],[ -1,
-1, -1, -1,
                    -1, -1, -1, 1, 1, -1],[ -1, -1, -1, -1, -1, -1,
-1, 1, 1,
                        -1],[ -1, -1, -1, -1, -1, -1, -1, 1, 1, -1],
[ -1, -1,
                            -1, -1, -1, -1, -1, 1, 1, -1],[ -1, -1,
-1, -1, -1,
                                -1, -1, 1, 1, -1],[ -1, -1, -1, -1,
-1, -1, -1,
                                    1, 1, -1],[ -1, -1, -1, -1, -1,
-1, -1, 1,
                                        1, -1] ])
x_dist1 = np.array([[1, 1, -1, -1, -1, -1, -1, -1, 1, 1], [-1, -1,
1, 1, 1, 1, 1,
    1, 1, -1], [-1, -1, -1, -1, -1, -1, 1, 1, 1, -1], [-1, -1, -1,
-1, -1, -1,
        1, 1, 1, -1], [-1, -1, -1, -1, -1, -1, 1, 1, 1, -1], [-1,
```

```
-1, -1, -1,
            -1, -1, 1, 1, 1, -1], [-1, -1, -1, -1, -1, -1, 1, 1, 1,
-1], [-1,
               -1, 1, 1, 1, 1, 1, 1, -1, -1], [-1, -1, 1, 1, 1, 1,
1, 1, -1,
                   -1], [-1, -1, -1, -1, -1, -1, 1, 1, 1, -1], [-1,
-1, -1, -1,
                       -1, -1, 1, 1, 1, -1], [-1, -1, -1, -1, -1,
-1, 1, 1, 1,
                           -1], [-1, -1, -1, -1, -1, -1, 1, 1, 1,
-1], [-1, -1,
                               -1, -1, -1, -1, 1, 1, 1, -1], [-1,
-1, 1, 1, 1,
                                   1, 1, 1, 1, -1], [-1, -1, 1, 1,
1, 1, 1, 1,
                                       -1, -1]])
x_dist2 = np.array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [-1, -1, -1, 1,
1, 1, 1, -1, -1, -1], [-1, -1,
    1, 1, 1, 1, 1, 1, -1, -1], [-1, 1, 1, 1, -1, -1, 1, 1, 1, -1],
[-1, 1, 1, 1,
        -1, -1, 1, 1, 1, -1], [-1, 1, 1, 1, -1, -1, 1, 1, 1, -1],
[-1, 1, 1, 1,
            -1, -1, 1, 1, 1, -1], [-1, 1, 1, 1, -1, -1, 1, 1, 1,
-1], [-1, 1, 1,
                1, -1, -1, 1, 1, 1, -1], [-1, 1, 1, 1, -1, -1, 1, 1,
1, -1],
             [-1, 1, 1, 1, -1, -1, 1, 1, 1, -1], [-1, 1, 1, 1, -1,
-1, 1, 1, 1,
                -1], [-1, 1, 1, 1, -1, -1, 1, 1, 1, -1], [-1, -1, 1,
1, 1, 1, 1,
                    1, -1, -1], [-1, -1, -1, 1, 1, 1, 1, -1, -1,
-1], [-1, -1,
                        -1, -1, -1, -1, -1, -1, -1, -1]])
x_dist3 = np.array([[1, -1, -1, 1, -1, 1, -1, 1, 1, -1], [1, -1, -1,
1, -1, 1, -1, 1, -1, -1], [1,
    -1, 1, -1, 1, -1, -1, 1, -1, -1], [1, -1, 1, -1, 1, -1, -1, 1,
-1, -1], [1,
        -1, 1, -1, 1, -1, -1, 1, -1, -1], [1, -1, 1, -1, 1, -1, -1,
1, -1, -1],
    [1, -1, 1, -1, 1, -1, -1, 1, -1, -1], [1, -1, -1, 1, -1, 1, -1,
1, 1, -1],
    [1, -1, -1, 1, -1, 1, -1, 1, 1, -1], [1, -1, 1, -1, 1, -1, -1,
1, -1, -1],
    [1, -1, 1, -1, 1, -1, -1, 1, -1, -1], [1, -1, 1, -1, 1, -1, -1,
1, -1, -1],
    [1, -1, 1, -1, 1, -1, -1, 1, -1, -1], [1, -1, 1, -1, 1, -1, -1,
1, -1, -1],
    [1, -1, -1, 1, -1, 1, -1, 1, -1, -1], [1, -1, -1, 1, -1, 1, -1,
1, 1, -1]])

# Flatten arrays
x1 = x1.ravel()
x2 = x2.ravel()
x3 = x3.ravel()
```

```
x4 = x4.ravel()
x5 = x5.ravel()
x_dist1 = x_dist1.ravel()
x_dist2 = x_dist2.ravel()
x_dist3 = x_dist3.ravel()

# Construct matrix P
P = np.array([x1,x2,x3,x4,x5])

# Hebb's rule
def hebbs(P):
    N = P.shape[1]
    W = np.zeros((N, N))
    for i in range(N):
        for j in range(N):
            if (j != i):
                W[i,j] = 1/N * np.dot(P[:,i],P[:,j])
    return W

# Asynchronous update functions
def asynch_update(W, s_in):
    s_out = np.sign(np.dot(W, s_in)).astype(int)
    return s_out
def asynch_T_times(W, s_in):
    s_new = s_in
    while True:
        s_old = s_new
        s_new = asynch_update(W,s_old)
        if (s_new == s_old).all():
            break
    return s_new

# Find attractor function
def find_attractor(P, s):
    attractors = np.concatenate((P,-P), axis = 0)
    indices = np.array([np.arange(1,6), -1*np.arange(1, 6)]).ravel()
    for it in range(attractors.shape[0]):
        nr_correct = sum(s == attractors[it,:])
        if (nr_correct == 160):
            return indices[it]
    return 6

# Calculate weight matrix
W = hebbs(P)

# Distorted pattern 1
s_dist1 = asynch_T_times(W,x_dist1)
print("Steady state for dist 1:", "\n",repr(np.reshape(s_dist1,
(16,10))))
print("Dist 1 converges to pattern" ,find_attractor(P,s_dist1))

# Distorted pattern 2
s_dist2 = asynch_T_times(W,x_dist2)
print("Steady state for dist 2:", "\n", repr(np.reshape(s_dist2,
```

```
(16,10))))
print("Dist 2 converges to pattern" ,find_attractor(P,s_dist2))

# Distorted pattern 3
s_dist3 = asynch_T_times(W,x_dist3)
print("Steady state for dist 3:", "\n", repr(np.reshape(s_dist3,
(16,10))))
print("Dist 3 converges to pattern" ,find_attractor(P,s_dist3))
```