Code for One-step Error Probability task

```
import numpy as np
import random

# The nr of patterns
p_vector = [12, 24, 48, 70, 100, 120]
# the number of bits
N = 120
diagonal_zero = True  # set true if you want w_ii = 0 otherwise false

probability_error = []
for p in p_vector:
    # Creating a errors variable that will grow when errors occur( when updated_bit != bit)
    errors = 0
    for _ in range(10 ** 5):

        # creating p random patterns with 120 bits in a 120xp matrix
        patterns = np.random.randint(0, 2, (N, p))
        patterns[patterns == 0] = -1

        # weight_matrix = sum([np.dot(patterns[:, i:i + 1], patterns[:, i:i + 1].T) for i in rang
        weight_matrix = np.dot(patterns, patterns.T)
        if diagonal_zero:
            np.fill_diagonal(weight_matrix, 0)

        # Drawing a random number between 0 and 11 to choose randomly one pattern to feed
        random_pattern_index = random.randint(1, 11)
        # Taking out one pattern to be feed randomly
        feed_pattern = patterns[:, random_pattern_index]
        # feed_pattern = patterns[:, random_pattern_index:random_pattern_index + 1]

        # Taking one bit/ neuron to update randomly
        random_bit_index = random.randint(0, 119)
        random_bit = feed_pattern[random_bit_index]

        # weight = weight_matrix[random_bit_index:random_bit_index + 1, :]
        weight = weight_matrix[random_bit_index, :]
        b = np.dot(weight, feed_pattern) / N
        if b == 0:
            b = 1

        new_bit = np.sign(b)

        if new_bit != random_bit:
            errors += 1

    probability_error.append(errors / 10 ** 5)
```

```
print(probability_error)
```