In [18]:
```
import numpy as np

states = ["Rain", "no Rain"]

observations = ["Umbrella", "no Umbrella"]

#transition matrix P(X_t | X_t-1)
transition_matrix = np.array ([
    [0.7, 0.3],
    [0.3, 0.7]
])

observations_matrix = np.array([
    [0.9, 0.2],
    [0.1, 0.8]
])

#initial state, assumed equal probability of Rain and no Rain
initial_state = np.array([0.5, 0.5])

print("Hidden Markov Model (HMM) Representation\n")
print(f"states: {states}")
print(f"observations: {observations}")

print("\nTransition Matrix P(X_t | X_t-1)")
print(transition_matrix)

print("\nObservations Matrix P(E_t | X_t)")
print(observations_matrix)

print("\nInitial State")
print(initial_state)
```

```
Hidden Markov Model (HMM) Representation

states: ['Rain', 'no Rain']
observations: ['Umbrella', 'no Umbrella']

Transition Matrix P(X_t | X_t-1)
[[0.7 0.3]
 [0.3 0.7]]

Observations Matrix P(E_t | X_t)
[[0.9 0.2]
 [0.1 0.8]]

Initial State
[0.5 0.5]
```

In [19]:
```
observation_sequence = [0, 0, 1, 0, 0]  # {Umbrella, Umbrella, No Umbrell
```

In [20]:
```
def forward_algorithm(observation_sequence, transition_matrix, observatio
```

```
    """
    Computes the forward probabilities f_t using the Forward Algorithm.
    Returns a list of all normalized forward messages.
    """
    num_states = len(initial_state)
    num_observations = len(observation_sequence)

    # store forward messages in the hashset
    forward_messages = []

    # initialize with prior belief and first observation
    f_t = initial_state * observation_matrix[observation_sequence[0]]
    f_t /= np.sum(f_t)  # Normalize
    forward_messages.append(f_t.copy())

    # recursive filtering for t >= 1
    for t in range(1, num_observations):
        f_t = observation_matrix[observation_sequence[t]] * (transition_m
        f_t /= np.sum(f_t)  # Normalize
        forward_messages.append(f_t.copy())

    return forward_messages

# run forward algorithm
forward_messages = forward_algorithm(observation_sequence, transition_mat

# display results
for t, f_t in enumerate(forward_messages, start=1):
    print(f"P(X_{t} | e_1:{t}): {f_t}")

# extract and print final probability of rain at day 2 and 5
p_rain_day_2 = forward_messages[1][0]  # P(Rain | e1:2)
p_rain_day_5 = forward_messages[4][0]  # P(Rain | e1:5)

print(f"\nProbability of Rain at Day 2: {p_rain_day_2:.3f} (Expected: 0.8
print(f"Probability of Rain at Day 5: {p_rain_day_5:.3f}")
```

```
P(X_1 | e_1:1): [0.81818182 0.18181818]
P(X_2 | e_1:2): [0.88335704 0.11664296]
P(X_3 | e_1:3): [0.19066794 0.80933206]
P(X_4 | e_1:4): [0.730794 0.269206]
P(X_5 | e_1:5): [0.86733889 0.13266111]

Probability of Rain at Day 2: 0.883 (Expected: 0.883)
Probability of Rain at Day 5: 0.867
```

In [ ]: