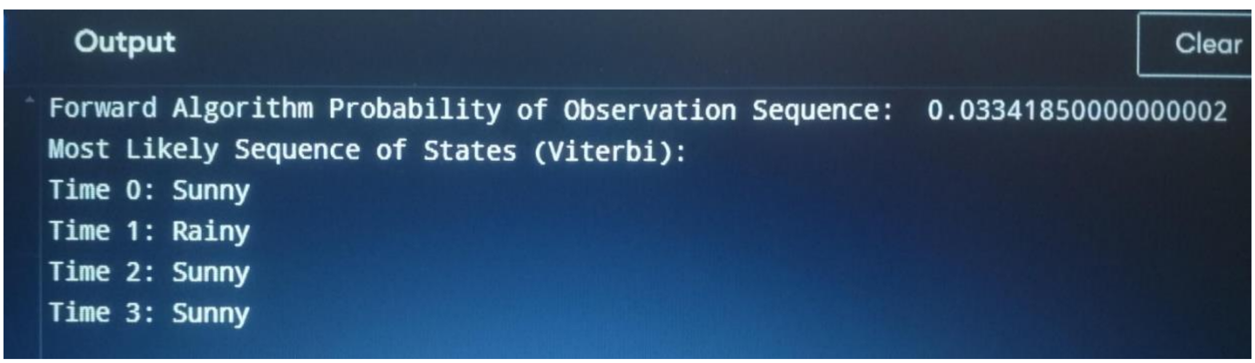
PROGRAM 1:

```
Import numpy as np

States = ['Rainy', 'Sunny']

Observations = ['Playing', 'Studying']

Initial_prob = [0.6, 0.4]

Transition_prob = np.array([
  [0.7, 0.3],
  [0.4, 0.6]
])

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

Obs_seq = ['Playing', 'Studying', 'Playing', 'Playing']

Obs_map = {obs: idx for idx, obs in enumerate(observations)}

State_map = {state: idx for idx, state in enumerate(states)}

Obs_seq_indices = [obs_map[obs] for obs in obs_seq]

Def forward_algorithm(initial_prob, transition_prob, emission_prob, obs_seq_indices):
  Num_states = len(states)
  Num_obs = len(obs_seq_indices)
  Alpha = np.zeros((num_obs, num_states))
  For state in range(num_states):
    Alpha[0, state] = initial_prob[state] * emission_prob[state, obs_seq_indices[0]]
  For t in range(1, num_obs):
    For state in range(num_states):
```

```
      Alpha[t, state] = np.sum(alpha[t-1] * transition_prob[:, state]) * emission_prob[state, 
obs_seq_indices[t]]
  Final_prob = np.sum(alpha[num_obs-1])
  Return final_prob, alpha
Def viterbi_algorithm(initial_prob, transition_prob, emission_prob, obs_seq_indices):
  Num_states = len(states)
  Num_obs = len(obs_seq_indices)
  Delta = np.zeros((num_obs, num_states))
  Psi = np.zeros((num_obs, num_states), dtype=int)
  For state in range(num_states):
    Delta[0, state] = initial_prob[state] * emission_prob[state, obs_seq_indices[0]]
    Psi[0, state] = 0
  For t in range(1, num_obs):
    For state in range(num_states):
      Max_prob = -1
      Max_state = -1
      For prev_state in range(num_states):
        Prob = delta[t-1, prev_state] * transition_prob[prev_state, state]
        If prob > max_prob:
          Max_prob = prob
          Max_state = prev_state
      Delta[t, state] = max_prob * emission_prob[state, obs_seq_indices[t]]
      Psi[t, state] = max_state
  Best_final_state = np.argmax(delta[num_obs-1])
  Best_path = [best_final_state]
  For t in range(num_obs-2, -1, -1):
```

```
        Best_state = psi[t+1, best_path[-1]]
        Best_path.append(best_state)
    Best_path.reverse()
    Return best_path, delta
Final_prob, alpha = forward_algorithm(initial_prob, transition_prob, emission_prob, obs_seq_indices)
Print(“Forward Algorithm Probability of Observation Sequence: “, final_prob)
Best_path, delta = viterbi_algorithm(initial_prob, transition_prob, emission_prob, obs_seq_indices)
Print(“Most Likely Sequence of States (Viterbi):”)
For t, state_idx in enumerate(best_path):
    Print(f”Time {t}: {states[state_idx]}”)
```



PROGRAM 2:

```
Import numpy as np
N_states = 2
N_observations = 3
Transition_probs = np.array([[0.7, 0.3],
                             [0.4, 0.6]])
```

```
Emission_probs = np.array([[0.5, 0.4, 0.1],
                          [0.1, 0.3, 0.6]])
Initial_probs = np.array([0.6, 0.4])
Observations = [0, 1, 2, 1, 0, 1, 2, 0, 1, 2]
Def forward_algorithm(obs, A, B, pi):
    N_states = A.shape[0]
    N_obs = len(obs)
     Alpha = np.zeros((n_states, n_obs))
       For s in range(n_states):
        Alpha[s, 0] = pi[s] * B[s, obs[0]]
       For t in range(1, n_obs):
        For s in range(n_states):
            Alpha[s, t] = np.sum(alpha[:, t-1] * A[:, s]) * B[s, obs[t]]
        Return alpha
Def viterbi_algorithm(obs, A, B, pi):
    N_states = A.shape[0]
    N_obs = len(obs)
        Viterbi = np.zeros((n_states, n_obs))
    Path = np.zeros((n_states, n_obs), dtype=int)
        For s in range(n_states):
        Viterbi[s, 0] = pi[s] * B[s, obs[0]]
        Path[s, 0] = 0
       For t in range(1, n_obs):
        For s in range(n_states):
            Max_prob = -1
            Max_state = -1
```

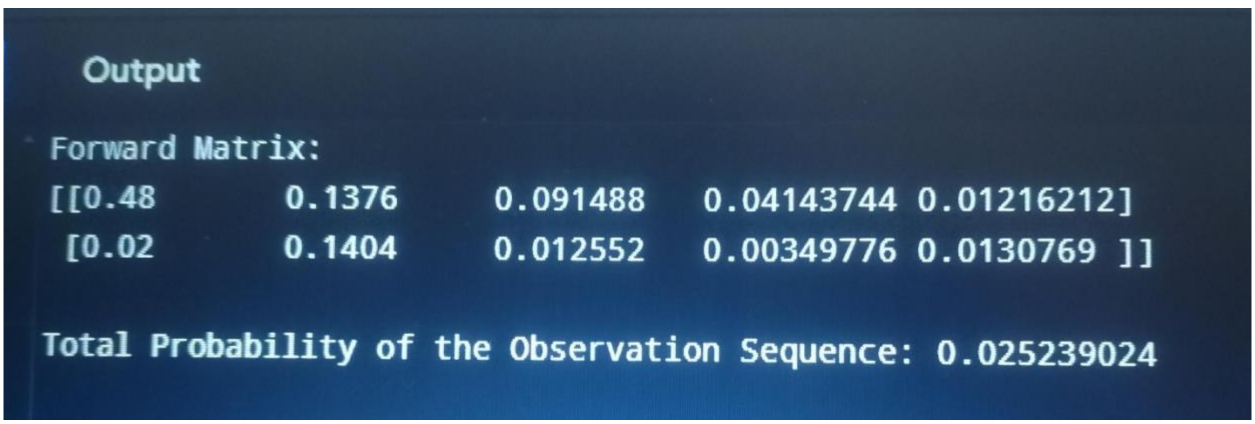
```
      For s_prev in range(n_states):
        Prob = viterbi[s_prev, t-1] * A[s_prev, s] * B[s, obs[t]]
        If prob > max_prob:
          Max_prob = prob
          Max_state = s_prev
      Viterbi[s, t] = max_prob
      Path[s, t] = max_state
    Best_path = np.zeros(n_obs, dtype=int)
  Best_path[-1] = np.argmax(viterbi[:, -1])
    For t in range(n_obs – 2, -1, -1):
    Best_path[t] = path[best_path[t + 1], t + 1]
    Return best_path
Alpha = forward_algorithm(observations, transition_probs, emission_probs, initial_probs)
Print("Forward Algorithm Result (Alpha):")
Print(alpha)
Best_path = viterbi_algorithm(observations, transition_probs, emission_probs,
initial_probs)
Print("Most Likely Hidden States (Viterbi):")
Print(best_path)
```

```
Output
Forward Algorithm Result (Alpha):
[[3.00000000e-01 9.04000000e-02 7.69600000e-03 6.72832000e-03
  3.52246400e-03 1.07462835e-03 9.18800845e-05 1.00825630e-04
  3.19682456e-05 2.76891758e-06]
 [4.00000000e-02 3.42000000e-02 2.85840000e-02 5.83776000e-03
  5.52115200e-04 4.16402496e-04 3.43338002e-04 2.33566826e-05
  1.32785096e-05 1.05345477e-05]]
Most Likely Hidden States (Viterbi):
[0 0 1 0 0 0 1 0 0 1]
```

PROGRAM 3:

```
Import numpy as np

States = [“Sunny”, “Rainy”]

Observations = [“Playing”, “Studying”]


Transition_probs = np.array([[0.7, 0.3],
                             [0.4, 0.6]])


Emission_probs = np.array([[0.6, 0.4],
                           [0.1, 0.9]])

Initial_probs = np.array([0.8, 0.2])

Obs_sequence = [0, 1, 0, 0, 1]

N_states = len(states)

N_observations = len(obs_sequence)

Forward = np.zeros((n_states, n_observations))
```

```
For s in range(n_states):
    Forward[s, 0] = initial_probs[s] * emission_probs[s, obs_sequence[0]]
For t in range(1, n_observations):
    For s in range(n_states):
        Forward[s, t] = np.sum(forward[:, t-1] * transition_probs[:, s]) * emission_probs[s, obs_sequence[t]]
Total_prob = np.sum(forward[:, n_observations – 1])
Print(“Forward Matrix:”)
Print(forward)
Print(“\nTotal Probability of the Observation Sequence:”, total_prob)
```



```
Output

Forward Matrix:
[[0.48       0.1376     0.091488   0.04143744 0.01216212]
 [0.02       0.1404     0.012552   0.00349776 0.0130769 ]]

Total Probability of the Observation Sequence: 0.025239024
```