### Bayesian network

**Program:**

**import numpy as np**

**# Define probabilities**

**P\_A = [0.6, 0.4] # P(A=0) = 0.6, P(A=1) = 0.4**

**P\_B = [0.7, 0.3] # P(B=0) = 0.7, P(B=1) = 0.3**

**# Conditional probabilities P(C | A, B)**

**P\_C\_given\_A\_B = {**

**(0, 0): [0.9, 0.1], # P(C=0|A=0,B=0) = 0.9, P(C=1|A=0,B=0) = 0.1**

**(0, 1): [0.8, 0.2], # P(C=0|A=0,B=1) = 0.8, P(C=1|A=0,B=1) = 0.2**

**(1, 0): [0.7, 0.3], # P(C=0|A=1,B=0) = 0.7, P(C=1|A=1,B=0) = 0.3**

**(1, 1): [0.1, 0.9] # P(C=0|A=1,B=1) = 0.1, P(C=1|A=1,B=1) = 0.9**

**}**

**# Function to compute P(C | A, B)**

**def compute\_posterior(A, B):**

**P\_C = P\_C\_given\_A\_B[(A, B)] # Lookup table**

**return {"P(C=0)": P\_C[0], "P(C=1)": P\_C[1]}**

**# Given evidence: A=1, B=0**

**A\_given = 1**

**B\_given = 0**

**# Compute P(C | A=1, B=0)**

**result = compute\_posterior(A\_given, B\_given)**

**# Display result**

**print(f"P(C=0 | A={A\_given}, B={B\_given}) = {result['P(C=0)']:.3f}")**

**print(f"P(C=1 | A={A\_given}, B={B\_given}) = {result['P(C=1)']:.3f}")**

**Output:**

**P(C=0 | A=1, B=0) = 0.700**

**P(C=1 | A=1, B=0) = 0.300**