```python
# Ejercicios de redes bayesianas con Python (Trabajo calificado)
# Realizado por: Correa Adrian
# Fecha: 18/05/2025

#link de GitHub:https://github.com/afca2002/Ejercicios-de-redes-bayesianas-con-Python-Trabajo-calificado-.git


# ------------------------------------------------------------
# Pregunta 2. Usando el código proporcionado en el aula virtual
# para el problema del tren, resolver las siguientes probabilidades:
#
# - P(heavy, yes, delayed, attend)
# - P(none, no, on time, miss)
# - P(none, yes, delayed)
# - P(none | miss)
# ------------------------------------------------------------

from pomegranate import *

# Rain node has no parents
rain = Node(DiscreteDistribution({
    "none": 0.7,
    "light": 0.2,
    "heavy": 0.1
}), name="rain")

# Track maintenance node is conditional on rain
maintenance = Node(ConditionalProbabilityTable([
    ["none", "yes", 0.4],
    ["none", "no", 0.6],
    ["light", "yes", 0.2],
    ["light", "no", 0.8],
    ["heavy", "yes", 0.1],
    ["heavy", "no", 0.9]
], [rain.distribution]), name="maintenance")

# Train node is conditional on rain and maintenance
train = Node(ConditionalProbabilityTable([
    ["none", "yes", "on time", 0.8],
    ["none", "yes", "delayed", 0.2],
    ["none", "no", "on time", 0.9],
    ["none", "no", "delayed", 0.1],
    ["light", "yes", "on time", 0.6],
    ["light", "yes", "delayed", 0.4],
    ["light", "no", "on time", 0.7],
    ["light", "no", "delayed", 0.3],
    ["heavy", "yes", "on time", 0.4],
    ["heavy", "yes", "delayed", 0.6],
    ["heavy", "no", "on time", 0.5],
    ["heavy", "no", "delayed", 0.5],
], [rain.distribution, maintenance.distribution]), name="train")

# Appointment node is conditional on train
appointment = Node(ConditionalProbabilityTable([
    ["on time", "attend", 0.9],
    ["on time", "miss", 0.1],
    ["delayed", "attend", 0.6],
    ["delayed", "miss", 0.4]
], [train.distribution]), name="appointment")

# Create a Bayesian Network and add states
model = BayesianNetwork()
model.add_states(rain, maintenance, train, appointment)

# Add edges connecting nodes
model.add_edge(rain, maintenance)
model.add_edge(rain, train)
model.add_edge(maintenance, train)
model.add_edge(train, appointment)
```

```python
# Finalize model
model.bake()

# Consultas de probabilidad
print("P(heavy, yes, delayed, attend):", model.probability([["heavy", "yes", "delayed", "attend"]]))
print("P(none, no, on time, miss):", model.probability([["none", "no", "on time", "miss"]]))
print("P(none, yes, delayed):", model.probability([["none", "yes", "delayed", None]]))

# Para P(none|miss), necesitamos calcular la probabilidad condicional
# Esto se hace utilizando el método predict_proba
observations = {"appointment": "miss"}
predictions = model.predict_proba(observations)
rain_prediction = predictions[0]  # El nodo "rain" es el primero en el modelo

if isinstance(rain_prediction, DiscreteDistribution):
    print("P(none | miss):", rain_prediction.parameters[0]["none"])
```