**Artificial Intelligence Lab**

**Name** - N Aditya Sai

**Register Number**- RA1911030010075

**Topic**: Implementation of uncertain methods for an application

**Experiment** – 9

**Aim:** To study Implementation of uncertain methods for an application.

**Code:**

import matplotlib.pyplot as plt

import seaborn; seaborn.set\_style('whitegrid')

import numpy

from pomegranate import \*

numpy.random.seed(0)

numpy.set\_printoptions(suppress=True)

# The guests initial door selection is completely random

guest = DiscreteDistribution({'A': 1./3, 'B': 1./3, 'C': 1./3})

# The door the prize is behind is also completely random

prize = DiscreteDistribution({'A': 1./3, 'B': 1./3, 'C': 1./3})

# Monty is dependent on both the guest and the prize.

monty = ConditionalProbabilityTable(

[[ 'A', 'A', 'A', 0.0 ],

[ 'A', 'A', 'B', 0.5 ],

[ 'A', 'A', 'C', 0.5 ],

[ 'A', 'B', 'A', 0.0 ],

[ 'A', 'B', 'B', 0.0 ],

[ 'A', 'B', 'C', 1.0 ],

[ 'A', 'C', 'A', 0.0 ],

[ 'A', 'C', 'B', 1.0 ],

[ 'A', 'C', 'C', 0.0 ],

[ 'B', 'A', 'A', 0.0 ],

[ 'B', 'A', 'B', 0.0 ],

[ 'B', 'A', 'C', 1.0 ],

[ 'B', 'B', 'A', 0.5 ],

[ 'B', 'B', 'B', 0.0 ],

[ 'B', 'B', 'C', 0.5 ],

[ 'B', 'C', 'A', 1.0 ],

[ 'B', 'C', 'B', 0.0 ],

[ 'B', 'C', 'C', 0.0 ],

[ 'C', 'A', 'A', 0.0 ],

[ 'C', 'A', 'B', 1.0 ],

[ 'C', 'A', 'C', 0.0 ],

[ 'C', 'B', 'A', 1.0 ],

[ 'C', 'B', 'B', 0.0 ],

[ 'C', 'B', 'C', 0.0 ],

[ 'C', 'C', 'A', 0.5 ],

[ 'C', 'C', 'B', 0.5 ],

[ 'C', 'C', 'C', 0.0 ]], [guest, prize])

# State objects hold both the distribution, and a high level name.

s1 = State(guest, name="guest")

s2 = State(prize, name="prize")

s3 = State(monty, name="monty")

# Create the Bayesian network object with a useful name

model = BayesianNetwork("Monty Hall Problem")

# Add the three states to the network

model.add\_states(s1, s2, s3)

# Add edges which represent conditional dependencies, where the second node is # conditionally dependent on the first node (Monty is dependent on both guest and prize)

model.add\_edge(s1, s3)

model.add\_edge(s2, s3)

model.bake()

model.probability([['A', 'B', 'C']])

model.probability([['A', 'B', 'C']])

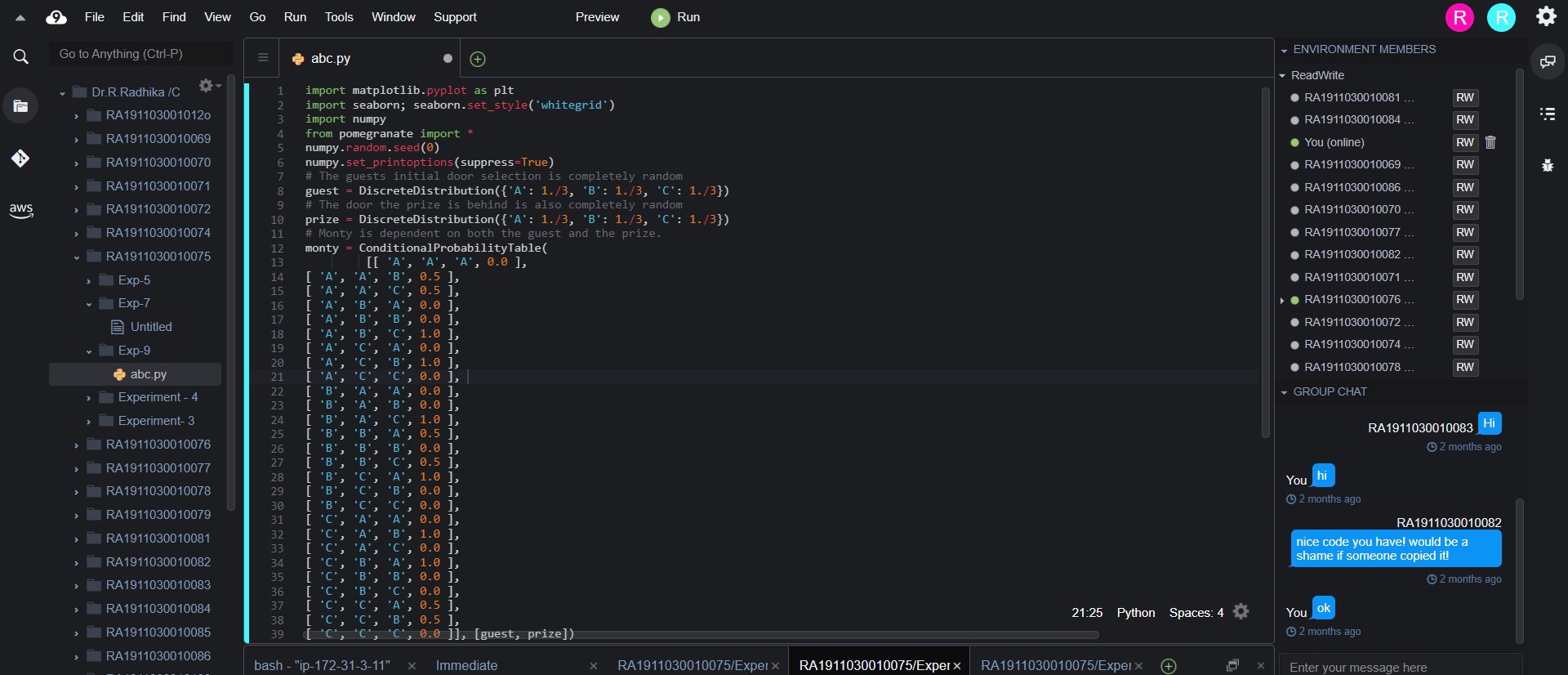
print(model.predict\_proba({}))

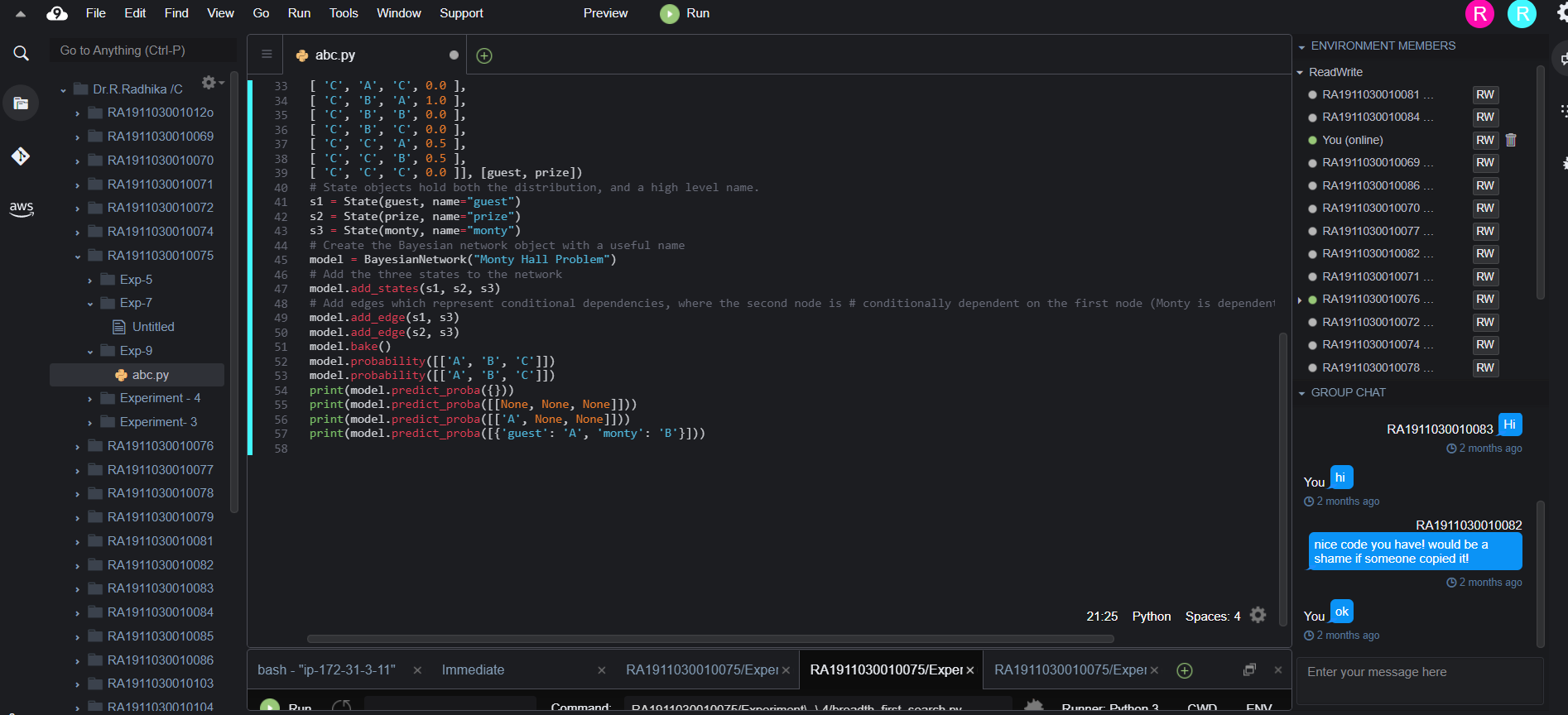
print(model.predict\_proba([[None, None, None]]))

print(model.predict\_proba([['A', None, None]]))

print(model.predict\_proba([{'guest': 'A', 'monty': 'B'}]))

**Output:**







**Result:** The following knowledge representation schemes have been successfully implemented in GUI Prolog.