Name : Parigya Jain
Roll No. : 2K18/MC/077

PROGRAM 8

AIM :

Demonstrating Markov Chain. WAP to implement Markov Chain

(a) Gambler Ruin Problem (b) Weather Forecasting Problem

THEORY :

A Markov chain is a mathematical system that experiences transitions from one state to another according to certain probabilistic rules. The defining characteristic of a Markov chain is that no matter *how* the process arrived at its present state, the possible future states are fixed. In other words, the
probability of transitioning to any particular state is dependent solely on the current state and time elapsed. The state space, or set of all possible states, can be anything: letters, numbers, weather conditions, baseball scores, or stock performances.

RESULT :

(a) Gambler ruin problem

Eg: A gambler has a fortune of Rs. 2 . He bets Rs 1 at a time and wins Rs 1 with probability 1⁄2 . He stops playing if he loses all his fortune or doubles it. Write the transition probability matrix that he loses his fortune at the end of three plays?

Name : Parigya Jain
Roll No. : 2K18/MC/077

CODE :

```
p = 0.5
money = 2
b=np.zeros(shape=(int(2*money+1),int(2*money+1)))
print("The absorbing barriers are at the {} and
{}".format(0,2*money))
for i in range(0, int(2*money+1):
    for j in range(0, int(2*money+1)):
        if i==0 pr i==int(2*money):
            b[i][i]=1
        else:
            if i+1==j:
                b[i][j]=p
            elif i-1==j:
                b[i][j]=1-p
 n=3
b=np.linalg.matrix_power(b,n)
print("The Probability that the gambler loses his fortune
at the end of {} plays: {}".format(n,b[int(money)][0]))
```

OUTPUT :

```
array([[1. , 0. , 0. , 0. , 0. ],
       [0.5, 0. , 0.5, 0. , 0. ],
       [0. , 0.5, 0. , 0.5, 0. ],
       [0. , 0. , 0.5, 0. , 0.5],
       [0. , 0. , 0. , 0. , 1. ]])
The absorbing barriers are at the 0 and 4.0
The Probability that the gambler loses his fortune at the
end of 3 plays: 0.25
```

Name : Parigya Jain
Roll No. : 2K18/MC/077

(b) Weather Forecasting Problem

Eg: Given every state's probability that rain occurs yesterday and today. Let it rain on both Monday and Tuesday. What is the probability that it will rain on Thursday?

Code:

```
import numpy as np
import math

state_0 = 0.7
state_1 = 0.5
state_2 = 0.4
state_3 = 0.2
a[0][0] = state_0;
a[0][2] = 1-state_0
a[1][0] = state_1
a[1][2] = 1-state_1
a[2][1] = state_2
a[2][3] = 1-state_2
a[3][1] = state_3
a[3][3] = 1-state_3
a=np.linalg.matric_power(a,2)
print("Probability that it rained the past 2 days and it will rain tomorrow:{}".format(a[0][[0]+a[0][1]))
```

OUTPUT :

```
array(([0.7, 0. , 0.3, 0. ],
       [0.5, 0. , 0.5, 0. ],
       [0. , 0.4, 0. , 0.6],
       [0. , 0.2, 0. , 0.8]])
Probability that it rained the past 2 days and it will rain tomorrow:0.61
```

DISCUSSION :

The program is executed using Python. We observe that using the transition matrix is more efficient than Champan's equation.