NAME:SUNDEEP A SRN:PES1UG20CS445

ROLL NO:48 SECTION: H

QUESTION:

Implementation of Markov Chains for:

a.. Population migration distribution between two Indian states

b.. Vote changing pattern of three political parties from one election to the next.

CODE:

a.. Population migration distribution between two Indian states

```
import numpy as np
state =["S", "T"]
transitionName = [["SS","ST"],["TS","TT"]]
transitionMatrix = [[0,400],[500,0]]
if len(transitionMatrix) == 2:
    pass
else:
    print("Error,some transitions are missing")
def pop_mig(transition):
    # Choose the starting state
    activityToday = "S"
    print("Start state: " + activityToday)
    # Shall store the sequence of states taken. So, this only has the starting
state for now.
    activityList = [activityToday]
    i = 0
    prob = 0
    while i != transition:
        if activityToday == "S":
            change = np.random.choice(transitionName[0],replace=True)
            if change == "SS":
                prob = prob +0
                activityList.append("S")
                pass
            else :
                prob = prob + 400
                activityToday = "ST"
                activityList.append("T")
        elif activityToday == "T":
            change = np.random.choice(transitionName[1],replace=True)
```

```
            if change == "TS":
                prob = prob + 500
                activityList.append("S")
                pass
            else:
                prob = prob + 0
                activityToday = "TT"
                activityList.append("T")
        else:
            return -1;
        i += 1
    print("Possible states: " + str(activityList))
    print("End state after migration after "+ str(transition) + " transition:
" +activityToday+" and now the population in "+ str(transition)+ " is "+
str(prob) )
pop_mig(1)
```

b.. Vote changing pattern of three political parties from one election to the next.

```
import numpy as np
states =["X","Y","Z"] #Names of political parties are A , B , C
transitionName = [["XX","XY","XZ"],["YX","YY","YZ"],["ZX","ZY","ZZ"]]
if len(transitionName) == 3:
    pass
else:
    print("Error,some transitions are missing")
def vote_change(elections):
    # Choose the starting state
    Startstate = "A"
    print("Start state: " + Startstate)
    # Shall store the sequence of states taken. So, this only has the starting
state for now.
    activityList = [Startstate]
    i = 0
    while i != elections:
        if Startstate == "A":
            change = np.random.choice(transitionName[0],replace=True)
            if change == "AA":
                activityList.append("A")
                pass
            elif change == "AB" :
                Startstate = "B"
                activityList.append("B")
            else:
                Startstate = "C"
                activityList.append("C")
        elif Startstate == "B":
            change = np.random.choice(transitionName[1],replace=True)
```

```
            if change == "BB":
                activityList.append("B")
                pass
            elif change == "BC" :
                Startstate = "C"
                activityList.append("C")
            else:
                Startstate = "A"
                activityList.append("A")
        elif Startstate == "C":
            change = np.random.choice(transitionName[2],replace=True)
            if change == "CC":
                activityList.append("C")
                pass
            elif change == "CA" :
                Startstate = "A"
                activityList.append("A")
            else:
                Startstate = "B"
                activityList.append("B")
        i += 1
    print("Possible states reach: " + str(activityList))
    print("After "+ str(elections) + " elections the votes were changed to
party : "+Startstate)
vote_change(1)
```

SCREENSHOT:

a.. Population migration distribution between two Indian states

```
C:\Users\HP\Desktop\PESU\SEM-4\Math\assignment\unit 2>python population_migration.py
Start state: S
Possible states: ['S', 'T']
End state after migration after 1 transition: ST and now the population in 1 is 400
```

b.. Vote changing pattern of three political parties from one election to the next.

After 1 election:

```
C:\Users\HP\Desktop\PESU\SEM-4\Math\assignment\unit 2>python vote_changing.py
Start state: A
Possible states reach: ['A', 'C']
After 1 elections the votes were changed to party : C
```

After 2 elections:

```
C:\Users\HP\Desktop\PESU\SEM-4\Math\assignment\unit 2>python vote_changing.py
Start state: A
Possible states reach: ['A', 'C', 'B']
After 2 elections the votes were changed to party : B
```