In [93]:
```
import pandas as pd
import numpy as np
```

7) Weather forecasting, a Markov Chain problem (Moderate)

In a particular area, there are 2 types of weather: sunny and rainy. The following is observed across a couple year period.

- The probability of weather staying sunny the following week is 80%.
- The probability of the weather changing from sunny to rainy is 20%.
- The probability of the weather staying rainy from the following week is 70%.
- The probability of weather changing from rainy to sunny over a week is 30%.

Given this information, can you create a transition matrix and calculate the steady state vector?

If you need a refresher, here is a resource on Markov Chains (https://en.wikipedia.org/wiki/Markov_chain).

Answer

If you do not know about Markov Chains, watch the first 2 min of this video (https://www.youtube.com/watch?v=Flj52QaHYVU) and the first 2 min of this video (https://www.youtube.com/watch?v=8noldJCb86Y)

In [94]:
```
#Summary
df=pd.DataFrame({'Sunny': [0.8,0.3], 'Rainy':[0.2,0.7]}, index=['Sunny','Rainy'])
df
```

Out[94]:

	Sunny	Rainy
Sunny	0.8	0.2
Rainy	0.3	0.7

Basically, a steady vector state L is the vector that cannon be transform by the transition matrix, ie:

$$A \times L = L$$
$$(A - I)L = 0$$

I is the identity matrix. Note that by solving this equation, you will have infinitely many solution.

Therefore, we also need to take into account that the row must add up to 1.

In [95]:
```
A=np.array([[0.8,0.2],[0.3,0.7]])
I=np.array([[1,0],[0,1]])
left=(A-I).T
left
```

Out[95]:
```
array([[-0.2,  0.3],
       [ 0.2, -0.3]])
```

In [96]:
```
#The top one can be reduce to 0.2x - 0.3y =0
#Also, x+y=1
A_new=np.array([[1,1],[0.2,-0.3]])
b=np.array([1,0]).T

print("The steady vector state L is")
np.linalg.solve(A_new,b)
```

```
The steady vector state L is
```

Out[96]:
```
array([0.6, 0.4])
```

8) Reverse an array, up to a a point (Easy)

Suppose you're given an array, a, as well as a position, p. Write a function to reverse your array, but only up to the position given in p. The rest of your array should remain untouched.

For example, given the following:

```
a = [1, 3, 4, 6]
p = 2
```

Your function should return:

```
[3,1,4,6]
```

In [97]:
```
#Thought: just cut off the array at that position and add them back

def p_inverse(array_a,p):
    a_change=array_a[:p]
    new_array=[]
    for i in range(len(a_change)):
        new_array.append(a_change[-(i+1)])

    new_array=new_array+array_a[p:]
    return new_array

p_inverse([1, 3, 4, 6],2)
```

Out[97]:
```
[3, 1, 4, 6]
```

9) Querying San Francisco Public Worker Salaries (Easy)

In [98]:
```
df=pd.read_csv("sf_salaries.csv")
df.head()
```

```
/Users/huybui/anaconda3/lib/python3.7/site-packages/IPython/core/interactiveshell.py:3044: DtypeWarning: Col
umns (3,4,5,6,12) have mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[98]:

	Id	EmployeeName	JobTitle	BasePay	OvertimePay	OtherPay	Benefits	TotalPay	TotalPayBenefits	Year	Notes	Agency	Status
0	1	NATHANIEL FORD	GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	167411	0	400184	NaN	567595.43	567595.43	2011	NaN	San Francisco	NaN
1	2	GARY JIMENEZ	CAPTAIN III (POLICE DEPARTMENT)	155966	245132	137811	NaN	538909.28	538909.28	2011	NaN	San Francisco	NaN
2	3	ALBERT PARDINI	CAPTAIN III (POLICE DEPARTMENT)	212739	106088	16452.6	NaN	335279.91	335279.91	2011	NaN	San Francisco	NaN
3	4	CHRISTOPHER CHONG	WIRE ROPE CABLE MAINTENANCE MECHANIC	77916	56120.7	198307	NaN	332343.61	332343.61	2011	NaN	San Francisco	NaN
4	5	PATRICK GARDNER	DEPUTY CHIEF OF DEPARTMENT, (FIRE DEPARTMENT)	134402	9737	182235	NaN	326373.19	326373.19	2011	NaN	San Francisco	NaN

In [99]:
```
#PANDAS
#3 highest and 3 lowest paid job title

#Thought: groupby job titles and find the average? Eliminate job with 0 salaries
df=df[df.TotalPay>0]
df_new=df.groupby("JobTitle").mean()[['TotalPay']].sort_values(by='TotalPay',ascending=False)
df_new=df_new.head(3).append(df_new.tail(3))
df_new
```

Out[99]:

	TotalPay
JobTitle	
GENERAL MANAGER-METROPOLITAN TRANSIT AUTHORITY	399211.275000
Chief Investment Officer	339653.700000
Chief of Police	329183.646667
BOARD/COMMISSION MEMBER, GROUP II	296.511628
BdComm Mbr, Grp2,M=$25/Mtg	270.305785
PUBLIC SAFETY COMMUNICATIONS TECHNICIAN	149.510000

SQL

SELECT JobTitle , AVG(TotalPay) AS MeanTotalPay

FROM sf_salaries

GROUPBY JobTitle

HAVING TotalPay>0

ORDER BY MeanTotalPay DESC

LIMIT 5

Then do another one with ASC and union them together

In [100]:
```
df.Year.unique()
```

Out[100]:
```
array([2011, 2012, 2013, 2014])
```

In [101]:
```
#BONUS

#SALARY OF Data scientist, analyst, and, statistician
jobs=[i for i in df.JobTitle.unique()
 if ("data" in i.lower())
     or ("analyst" in i.lower())
    or ("statistician" in i.lower())]

#Top 5 paid individuals
df[df.JobTitle.isin(jobs)].sort_values(by="TotalPay",ascending=False)[["EmployeeName"
                                                                      ,"JobTitle","TotalPay"]].head(5)
```

Out[101]:

	EmployeeName	JobTitle	TotalPay
37235	Jeffrey Hildebrant	IS Business Analyst-Principal	178813.70
76364	Adam B Mazurkiewicz	Water Operations Analyst	152848.32
38605	Bharti Muni	IS Business Analyst-Principal	152734.79
2176	BRENDA WALKER	PRINCIPAL ADMINISTRATIVE ANALYST	150722.45
38905	Jan Crosbie Taylor	Pr Administrative Analyst	149630.98

10) Probability of passing through interview stages (Easy)

Given the information below, if you had a good first interview, what is the probability you will receive a second interview?

- 50% of all people who received a first interview received a second interview
- 95% of people that received a second interview had a good first interview
- 75% of people that did not receive a second interview had a good first interview

Answer

Let A be receiving a second interview, B be receiving a good first interview. Bayes' formula

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B|A) \cdot P(A) + P(B|\neg A) \cdot P(\neg A)}$$

where

- $P(B|A) = 0.95$
- $P(A) = 0.5$
- $P(B|\neg A) = 0.75$
- $P(\neg A) = 1 - P(A) = 0.5$

In [102]:
```
P_A_B=(0.95*0.5)/(0.95*0.5+0.75*0.5)
print("If you had a good first interview, the probability you will receive a second interview is: ")
print(P_A_B)
```

```
If you had a good first interview, the probability you will receive a second interview is: 
0.5588235294117647
```

11) Check whether two arrays are equal

Given two arrays, write a function in vanilla Python (e.g., no libraries) to check whether or not the arrays are equal. You can consider the two arrays equal if both of them contain the same set of elements - the order of elements can differ.

For example:

```
#Given the following:

arr1 = [1,5,6,7,8,0]
arr2 = [0,5,7,6,8,1]
#output = True

arr3 = [1,5,6,7,8,0]
arr4 = [0,7,7,7,8,1]
#output = False
```

In [103]:

```
arr1=[1,5,6,7,8,0]
arr2 = [0,5,7,6,8,1]
arr3 = [1,5,6,7,8,0]
arr4 = [0,7,7,7,8,1]

#With set
def compare_1(a,b):
    return set(a)==set(b)

print(compare_1(arr1,arr2))
print(compare_1(arr3,arr4))
```

```
True
False
```

In [104]:

```
#Without set

#Thought: sort them and compare
def compare_2(a,b):
    a.sort()
    b.sort()
    return a==b

print(compare_2(arr1,arr2))
print(compare_2(arr3,arr4))
```

```
True
False
```

12) Creating K-Mean Clustering from scratch

Pseudo Code:

Input: K, set of point $x_1, x_2, \ldots, x_n$

Place centroids: $c_1, c_2, \ldots, c_k$ at random location

Repeat until convergence:

- For each point x_i find nearest centroid c_j (Euclidean distance), then x_i will belong to cluster j
- Calculate the new centroid for k clusters.

Until next time ...