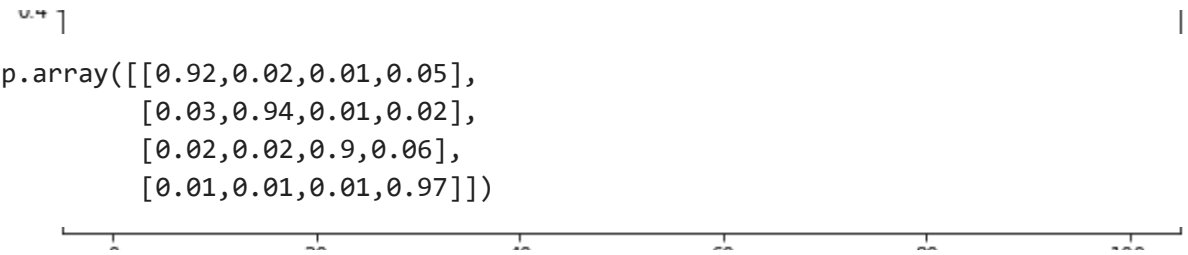
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


# Brand loyalty Problem
# I have taken a hypothetical situation based over some probabilities which I'll be puttin
# There are three network based companies which sell router based internet connection to c
# Data shows that
# 10% of Hathway customers will switch to ADN and 10% to Excitel.
# 20% of ADN customers will switch to Hathway and 20% to Excitel.
# 10% of Excitel customers will switch to Hathway and 20% to ADN .
# After significantly long time we will be finding the loyalty % that customers will keep


# P is transition state matrix
P = np.array([[0.8,0.1,0.1],
              [0.2,0.6,0.2],
              [0.1,0.2,0.7]])


state=np.array([[1.0,0.0,0.0]])
stateHist=state
dfStateHist=pd.DataFrame(state)
distr_hist = [[0,0,0]]

for x in range(100):
    state=np.dot(state,P)
    stateHist=np.append(stateHist,state,axis=0)
    dfDistrHist = pd.DataFrame(stateHist)


# Plotting graph
plt.figure(figsize=(10,5))
print(dfDistrHist)
sns.lineplot(data=dfDistrHist)
```

```
            0         1         2
0    1.000000  0.000000  0.000000
1    0.800000  0.100000  0.100000
2    0.670000  0.160000  0.170000
3    0.585000  0.197000  0.218000
4    0.529200  0.220300  0.250500
..        ...       ...       ...
96   0.421053  0.263158  0.315789
97   0.421053  0.263158  0.315789
98   0.421053  0.263158  0.315789
99   0.421053  0.263158  0.315789
100  0.421053  0.263158  0.315789

[101 rows x 3 columns]
<matplotlib.axes._subplots.AxesSubplot at 0x7f191399e190>
```

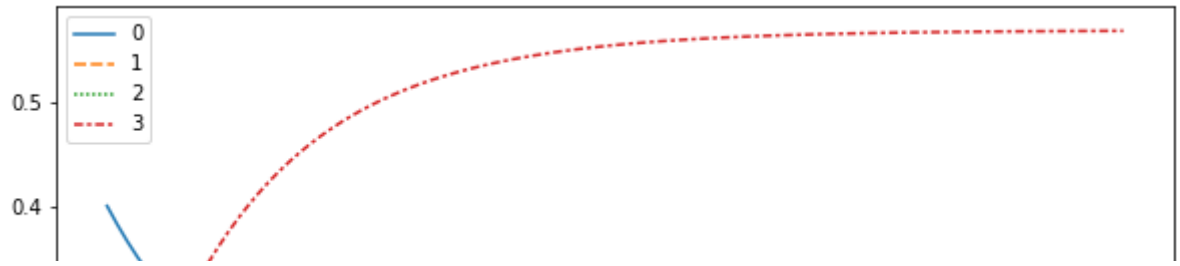
```
# Predicting Market Share
# These are Hypothetical situation, based on real scenerios
# There are three Product based companies Mcaffe , Quickheal and Kaspersky selling antivir
# They designed and implemented marketing strategies that shows it will attract
# 2% of customer base of MCAFFE
# 6% of Quickheal users
# 5% of KASPERSKEY users And will retain 97% of its users.
# Based on this, we will develop a model to predict market share.
```



```
P = np.array([[0.92,0.02,0.01,0.05],
              [0.03,0.94,0.01,0.02],
              [0.02,0.02,0.9,0.06],
              [0.01,0.01,0.01,0.97]])
```

```
state=np.array([[0.4,0.32,0.18,0.10]])
stateHist=state
dfStateHist=pd.DataFrame(state)
distr_hist = [[0,0,0,0]]

for x in range(100):
    state=np.dot(state,P)
    stateHist=np.append(stateHist,state,axis=0)
    dfDistrHist = pd.DataFrame(stateHist)


# Plotting graph
plt.figure(figsize=(10,5))
print(dfDistrHist)
sns.lineplot(data=dfDistrHist)
```

```
             0         1         2         3
0     0.400000  0.320000  0.180000  0.100000
1     0.382200  0.313400  0.170200  0.134200
2     0.365772  0.306986  0.161478  0.165764
3     0.350607  0.300769  0.153715  0.194908
4     0.336605  0.294759  0.146807  0.221830
..         ...       ...       ...       ...
96    0.161254  0.179236  0.090910  0.568599
97    0.161235  0.179211  0.090910  0.568643
98    0.161217  0.179188  0.090910  0.568685
99    0.161200  0.179166  0.090910  0.568723
100   0.161185  0.179146  0.090910  0.568760

[101 rows x 4 columns]
<matplotlib.axes._subplots.AxesSubplot at 0x7f1913a432d0>
```

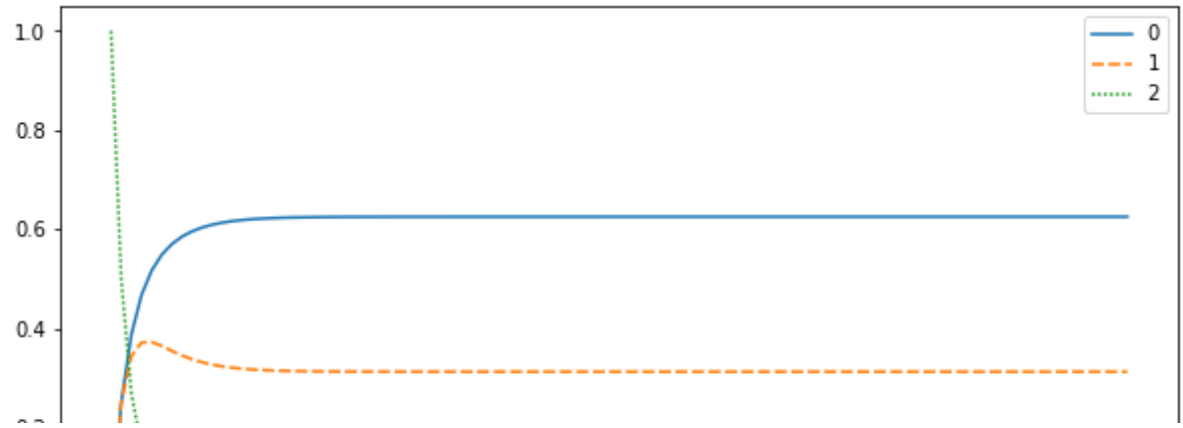



```
# Prediction of Market trend based on transition probabilities
# So we basically have three types of trend in a market. These are
# Bull markets: periods of time where prices generally are rising, due to the actors havin
# Bear markets: periods of time where prices generally are declining, due to the actors ha
# Stagnant markets : periods of time where the market is characterized by neither a declin
# Consider a fair market environment lets suppose a market condition.
# After a week characterized of a bull market trend there is a 90% chance that another bul
```

0 20 40 60 80 100

```
P = np.array([[0.9,0.075,0.025],
              [0.15,0.8,0.05],
              [0.25,0.25,0.5]])


state=np.array([[0.0,0.0,1.0]])
stateHist=state
dfStateHist=pd.DataFrame(state)
distr_hist = [[0,0,0]]

for x in range(100):
    state=np.dot(state,P)
    stateHist=np.append(stateHist,state,axis=0)
    dfDistrHist = pd.DataFrame(stateHist)


plt.figure(figsize=(10,5))
print(dfDistrHist)
sns.lineplot(data=dfDistrHist)
```

```
            0         1         2
0     0.00000  0.000000  1.000000
1     0.25000  0.250000  0.500000
2     0.38750  0.343750  0.268750
3     0.46750  0.371250  0.161250
4     0.51675  0.372375  0.110875
..        ...       ...       ...
96    0.62500  0.312500  0.062500
97    0.62500  0.312500  0.062500
98    0.62500  0.312500  0.062500
99    0.62500  0.312500  0.062500
100   0.62500  0.312500  0.062500

[101 rows x 3 columns]
<matplotlib.axes._subplots.AxesSubplot at 0x7f191375a1d0>
```




```
# Credit risk management the transition matrix represents the likelihood of the future evo
# The transition matrix will describe the probabilities that a certain company, country, e
# current state, or transition into a new state. The following probability transition matr
# of the credit rating agencies such as Standard & Poor, Moody’s and Fitch. Where the tabl
# bonds in the financial and industrial sectors.


P = np.array([[0.9195, 0.0746, 0.0048, 0.0008, 0.0004, 0.0000, 0.0000, 0.0000],
              [0.6400, 0.9181, 0.0676, 0.0060, 0.0008, 0.0012, 0.0003, 0.0000],
              [0.0700, 0.0227, 0.9169, 0.0512, 0.0058, 0.0025, 0.0001, 0.0004],
              [0.0400, 0.0270, 0.0556, 0.8786, 0.0485, 0.0108, 0.0017, 0.0024],
              [0.0600, 0.0010, 0.0061, 0.0779, 0.8148, 0.0790, 0.0111, 0.0101],
              [0.0000, 0.0010, 0.0028, 0.0046, 0.0695, 0.8280, 0.0396, 0.0545],
              [0.1900, 0.0000, 0.0039, 0.0078, 0.0243, 0.1213, 0.6045, 0.2369],
              [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 3.0000]])


financial_data = pd.DataFrame(P, columns=[ 'AAA', 'AA', 'A', 'BBB', 'BB' , 'B' , 'CCC' , '
                           index= [ 'AAA' , 'AA' , 'A' , ' BBB' , ' BB' , ' B' , 'CCC ' ,


print(financial_data)
# Database of credit risk management
```

```
          AAA      AA       A     BBB      BB       B     CCC       D
AAA    0.9195  0.0746  0.0048  0.0008  0.0004  0.0000  0.0000  0.0000
AA     0.6400  0.9181  0.0676  0.0060  0.0008  0.0012  0.0003  0.0000
A      0.0700  0.0227  0.9169  0.0512  0.0058  0.0025  0.0001  0.0004
 BBB   0.0400  0.0270  0.0556  0.8786  0.0485  0.0108  0.0017  0.0024
```

```
BB   0.0600  0.0010  0.0061  0.0779  0.8148  0.0790  0.0111  0.0101
B    0.0000  0.0010  0.0028  0.0046  0.0695  0.8280  0.0396  0.0545
CCC  0.1900  0.0000  0.0039  0.0078  0.0243  0.1213  0.6045  0.2369
D    0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  3.0000
```

```
years = int(input("Enter Years: "))
```

```
Enter Years: 2012
```

```
convergence_matrix = np.power(financial_data,years)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: RuntimeWarning: overf
  """Entry point for launching an IPython kernel.
```

```
financial_data_plt = pd.DataFrame(convergence_matrix, columns=[ 'AAA', 'AA', 'A', 'BBB', '
                          index= [ 'AAA' , 'AA' , 'A' , ' BBB' , ' BB' , ' B' , 'CCC ' ,
```

```
print(financial_data_plt)
```

```
                AAA           AA            A  ...             B  CCC    D
AAA   4.635408e-74  0.000000e+00  0.000000e+00  ...  0.000000e+00  0.0  0.0
AA    0.000000e+00  2.160999e-75  0.000000e+00  ...  0.000000e+00  0.0  0.0
A     0.000000e+00  0.000000e+00  1.555282e-76  ...  0.000000e+00  0.0  0.0
 BBB  0.000000e+00  0.000000e+00  0.000000e+00  ...  0.000000e+00  0.0  0.0
 BB   0.000000e+00  0.000000e+00  0.000000e+00  ...  0.000000e+00  0.0  0.0
 B    0.000000e+00  0.000000e+00  0.000000e+00  ...  0.000000e+00  0.0  0.0
CCC   0.000000e+00  0.000000e+00  0.000000e+00  ...  0.000000e+00  0.0  0.0
D     0.000000e+00  0.000000e+00  0.000000e+00  ...  0.000000e+00  0.0  inf

[8 rows x 8 columns]
```