```python
#!/usr/bin/env python3


import numpy as np

import pandas as pd

import matplotlib.pyplot as plt


DGS1 = pd.read_csv('DGS1.csv')

DGS1['DATE'] = pd.to_datetime(DGS1['DATE'])

DGS1.replace('.',np.nan,inplace=True)

DGS1 = DGS1.dropna(subset=['DGS1'])

DGS1['DGS1'] = DGS1['DGS1'].astype(float)

#DGS1.set_index('DATE',inplace=True)


DGS10 = pd.read_csv('DGS10.csv')

DGS10['DATE'] = pd.to_datetime(DGS10['DATE'])

DGS10.replace('.',np.nan,inplace=True)

DGS10 = DGS10.dropna(subset=['DGS10'])

DGS10['DGS10'] = DGS10['DGS10'].astype(float)


USREC = pd.read_csv('USREC.csv')

USREC['DATE'] = pd.to_datetime(USREC['DATE'])

USREC.replace('.',np.nan,inplace=True)

USREC = USREC.dropna(subset=['USREC'])

USREC['USREC'] = USREC['USREC'].astype(float)


DGS1 = DGS1.groupby([DGS1.DATE.dt.year,DGS1.DATE.dt.month])['DGS1'].mean()
```

```
#DGS1 = DGS1.reset_index()

DGS10 = DGS10.groupby([DGS10.DATE.dt.year,DGS10.DATE.dt.month])['DGS10'].mean()

#DGS10 = DGS10.reset_index()

USREC = USREC.groupby([USREC.DATE.dt.year,USREC.DATE.dt.month])['USREC'].mean()

#USREC = USREC.reset_index()

#print(DGS1,'\n',DGS10,'\n',USREC)

shape = USREC.index



merged = pd.merge(USREC,DGS1,left_index=True,right_index=True)

df = pd.merge(merged,DGS10,left_index=True,right_index=True)

#print(df)

#adding column for if short rate higher than long rate, 1 = short higher

newcol = []

for col,i in df.iterrows():

    if i['DGS1'] > i['DGS10']:

        newcol.append(1)

    else:

        newcol.append(0)

df.insert(3,"Short_High",newcol,True)




#df.to_csv("JoinedColumns.csv")



df.index = pd.to_datetime([f'{a}-{b}' for a,b in df.index],errors='coerce')

#print(df)

Dates=df.index
```

```
fig = plt.subplots(figsize=(16,10))

plt.plot(Dates,df['USREC'],label="USREC")

plt.plot(Dates,df['DGS10'],label="DGS10",color="forestgreen")

plt.plot(Dates,df['DGS1'],label="DGS1",color="rebeccapurple")

plt.plot(Dates,df['Short_High'],label="Short Greater than Long",color="red")

plt.xlabel("Date")

plt.ylabel("Rate")

plt.title("Time Series Of DGS1, DGS10, and USREC")

plt.legend()

plt.fill_between(Dates,-.15,17.75,where=df['USREC']==1,color="blue",alpha=.2)

plt.fill_between(Dates,-.15,17.75,where=df['Short_High']==1,color="red",alpha=.2)

plt.grid(axis="y",alpha=.75)

plt.xlim(df.index.min(),df.index.max())

plt.ylim(-0.15,17.75)

plt.show()


#Creating column for markov chain state

#short > Long, in recession = 1

#long > short, in recession = 2

#short > long, not in recession = 3

#long > short, not in recession = 4

newcol = []

for col,i in df.iterrows():

    if (i['Short_High'] == 1) and (i['USREC'] == 1):

        newcol.append(1)

    elif (i['Short_High'] == 0) and (i['USREC'] == 1):
```

```
        newcol.append(2)

    elif (i['Short_High'] == 1) and (i['USREC'] == 0):

        newcol.append(3)

    else:

        newcol.append(4)



df.insert(4,"State",newcol,True)



counts = np.array([[0,0,0,0],

                  [0,0,0,0],

                  [0,0,0,0],

                  [0,0,0,0]])



#determining number of transitions

for i in range(len(newcol)-1):

    for j in [1,2,3,4]:

        if newcol[i] == j:

            for k in [1,2,3,4]:

                if newcol[i+1] == k:

                    counts[j-1,k-1] = counts[j-1,k-1] + 1



total = np.sum(counts,axis=1)

print(counts)

print(total)

markov = np.zeros((4,4),dtype=float)

for i in range(4):
```

```
    for j in range(4):
        markov[i,j] = counts[i,j]/total[i]
np.set_printoptions(precision=3)
print(markov)
#print(df)
future1 = list(df['State'])
future2 = list(df['State'])
future3 = list(df['State'])
future4 = list(df['State'])
state = df.loc['2024-03-01','State']
#print(state)
state = int(state)
N = 240
def chain(Markov,N,array,X0):
    state = X0
    for i in range(N):
            state = state-1
            state = np.random.choice(len(Markov), p = Markov[state])
            state = state+1
            array.append(state)
    return(array)


enddate = max(Dates)
#print(enddate)
future1 = chain(markov,N,future1,state)
future2 = chain(markov,N,future2,state)
```

```
future3 = chain(markov,N,future3,state)

future4 = chain(markov,N,future4,state)

#print(future1,'\n',future2,'\n',future3,'\n',future4)

#print(len(future1))

dates = pd.date_range(start = pd.to_datetime('1/1/1962'), periods = len(future1), freq = 'MS')

#print(dates)

projected_states = pd.DataFrame(

    {'Dates': dates,

     'Sim 1': future1,

     'Sim 2': future2,

     'Sim 3': future3,

     'Sim 4': future4}

)

#print(projected_states)

fig = plt.subplots(figsize=(16,10))

plt.plot(dates,projected_states['Sim 1'], label = "Sim 1")

plt.axvline(x = enddate, color='gray',linestyle='dashed')

plt.xlabel("Dates")

plt.ylabel("State")

plt.title("States of Markov Chain Projected +20 years")

plt.grid(axis = 'y',alpha = .75)

plt.legend()

plt.xlim(dates.min(),dates.max())

plt.ylim(0.5,4.5)

plt.yticks([1,2,3,4])

plt.show()
```

```
fig = plt.subplots(figsize=(16,10))

plt.plot(dates,projected_states['Sim 2'], label = "Sim 2", color = "red")

plt.axvline(x = enddate, color='gray',linestyle='dashed')

plt.xlabel("Dates")

plt.ylabel("State")

plt.title("States of Markov Chain Projected +20 years")

plt.grid(axis = 'y',alpha = .75)

plt.legend()

plt.xlim(dates.min(),dates.max())

plt.ylim(0.5,4.5)

plt.yticks([1,2,3,4])

plt.show()


fig = plt.subplots(figsize=(16,10))

plt.plot(dates,projected_states['Sim 3'],label = "Sim 3",color = "forestgreen")

plt.axvline(x = enddate, color='gray',linestyle='dashed')

plt.xlabel("Dates")

plt.ylabel("State")

plt.title("States of Markov Chain Projected +20 years")

plt.grid(axis = 'y',alpha = .75)

plt.legend()

plt.xlim(dates.min(),dates.max())

plt.ylim(0.5,4.5)

plt.yticks([1,2,3,4])

plt.show()
```

```
fig = plt.subplots(figsize=(16,10))

plt.plot(dates,projected_states['Sim 4'],label = "Sim 4", color = "rebeccapurple")

plt.axvline(x = enddate, color='gray',linestyle='dashed')

plt.xlabel("Dates")

plt.ylabel("State")

plt.title("States of Markov Chain Projected +20 years")

plt.grid(axis = 'y',alpha = .75)

plt.legend()

plt.xlim(dates.min(),dates.max())

plt.ylim(0.5,4.5)

plt.yticks([1,2,3,4])

plt.show()


X = len(future1)
```