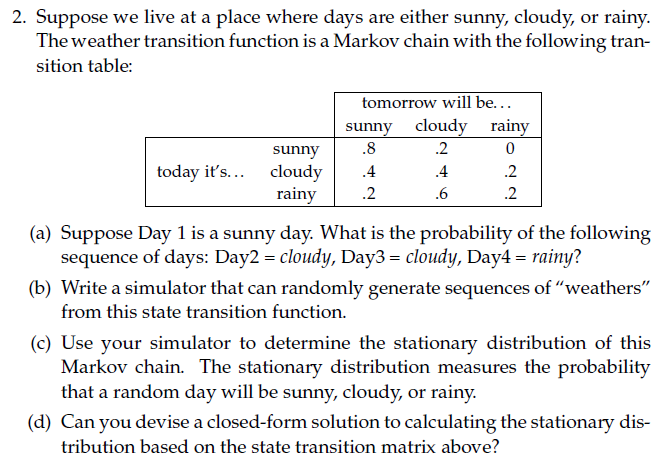
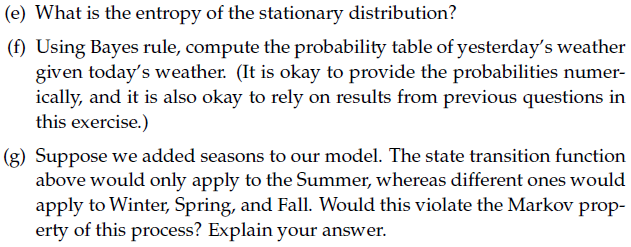
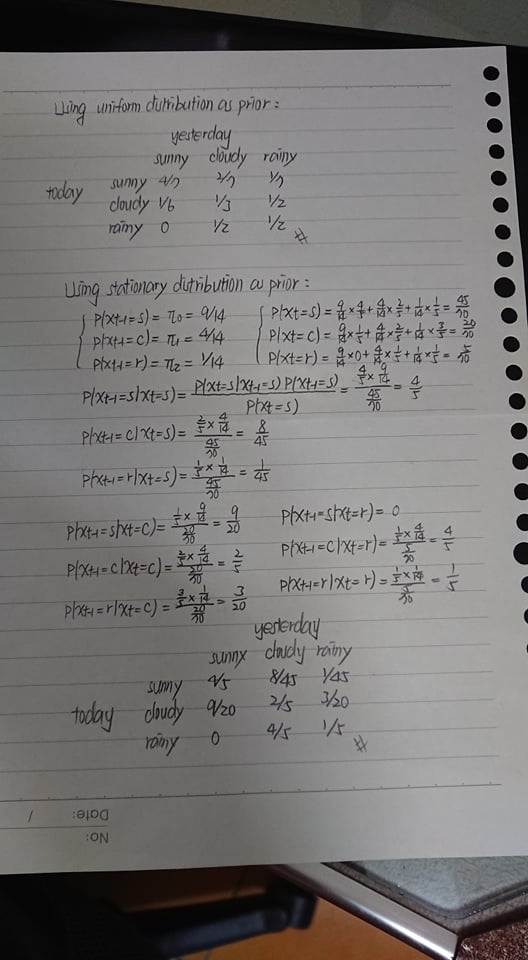
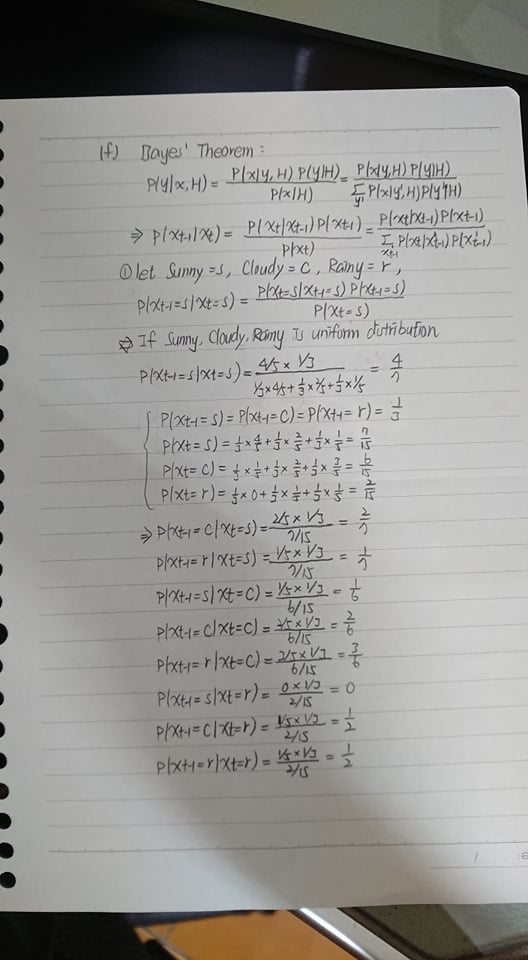
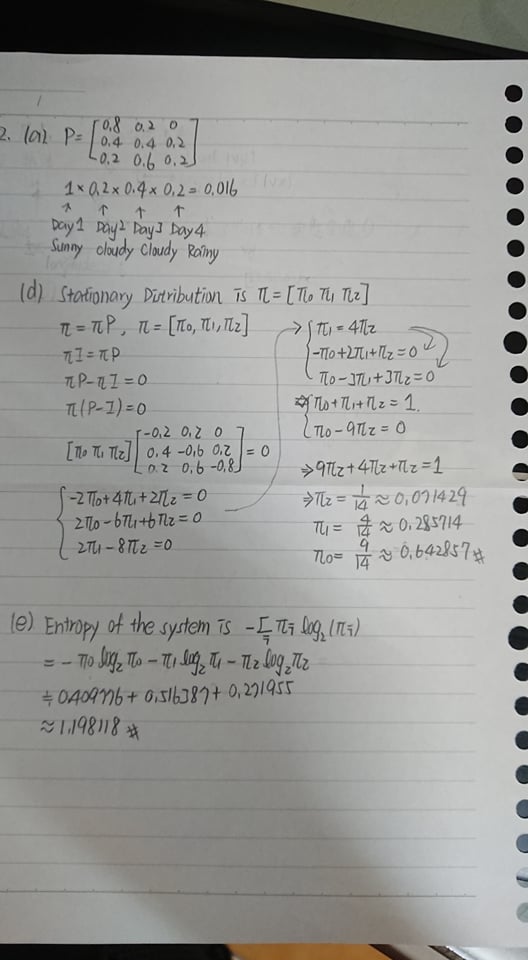
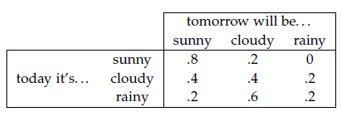
# Excercise2 part a,d,e,f





# Excercise2 part g

馬可夫性質的定義是：未來狀態的條件機率分佈只依賴於當前狀態。我認為加入季節的特徵只是令馬可夫鏈多了一個特徵(條件)而已，也就是說當考慮明天的天氣時，也要考慮到當前的季節是什麼。以前面的表格舉例：



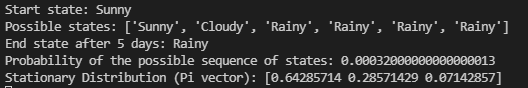
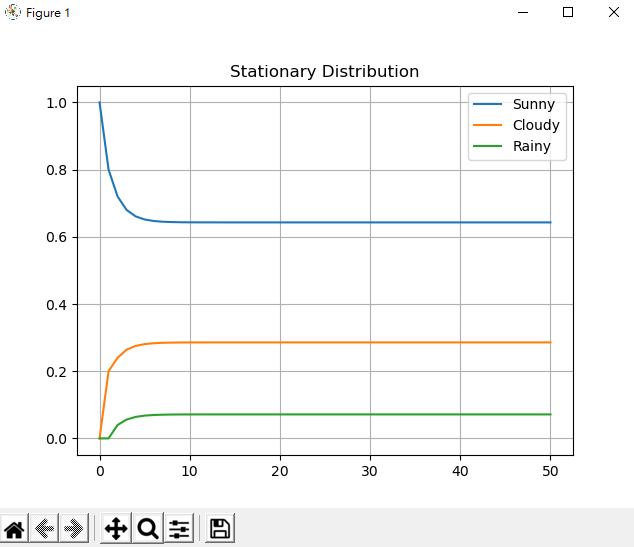
P(Xt=sunny| Xt-1=sunny) = 0.8若考慮季節，就會寫成：

P(Xt=sunny| Xt-1=sunny, Summer) = 0.8

因此加入季節特徵不違背馬可夫性質。

# Excercise2 part b,c

**Result:**

**** ****

**CODE:**

import numpy as np

import pandas as pd

import random as rm

import matplotlib.pyplot as plt

# The statespace

states = ["Sunny","Cloudy","Rainy"]

# Possible sequences of events

transitionName = [["SS","SC","SR"],["CS","CC","CR"],["RS","RC","RR"]]

# Probabilities matrix (transition matrix)

transitionMatrix = [[0.8, 0.2, 0.0],[0.4, 0.4, 0.2],[0.2, 0.6, 0.2]]

# # Check transition matrix's probibilities

# if sum(transitionMatrix[0])+sum(transitionMatrix[1])+sum(transitionMatrix[1]) != 3:

#     print("Somewhere, something went wrong. Transition matrix, perhaps?")

# else: print("All is gonna be okay, you should move on!! ;)")

# A function that implements the Markov model to forecast the state/mood.

def weather\_forecast(days, startWeather):

    # Choose the starting state

    weatherToday = startWeather

    print("Start state: " + weatherToday)

    # Shall store the sequence of states taken. So, this only has the starting state for now.

    weatherList = [weatherToday]

    # To calculate the probability of the weatherList

    prob = 1

    for i in range(days):

        if weatherToday == "Sunny":

            change = np.random.choice(transitionName[0],replace=True,p=transitionMatrix[0])

            if change == "SS":

                prob = prob \* 0.8

                weatherList.append("Sunny")

                pass    # 繼續執行迴圈剩下的程式碼

            elif change == "SC":

                prob = prob \* 0.2

                weatherToday = "Cloudy"

                weatherList.append("Cloudy")

            else:

                prob = prob \* 0.0

                weatherToday = "Rainy"

                weatherList.append("Rainy")

        elif weatherToday == "Cloudy":

            change = np.random.choice(transitionName[1],replace=True,p=transitionMatrix[1])

            if change == "CS":

                prob = prob \* 0.4

                weatherList.append("Sunny")

                pass    # 繼續執行迴圈剩下的程式碼

            elif change == "CC":

                prob = prob \* 0.4

                weatherToday = "Cloudy"

                weatherList.append("Cloudy")

            else:

                prob = prob \* 0.2

                weatherToday = "Rainy"

                weatherList.append("Rainy")

        else:

            change = np.random.choice(transitionName[0],replace=True,p=transitionMatrix[0])

            if change == "RS":

                prob = prob \* 0.2

                weatherList.append("Sunny")

                pass    # 繼續執行迴圈剩下的程式碼

            elif change == "RC":

                prob = prob \* 0.6

                weatherToday = "Cloudy"

                weatherList.append("Cloudy")

            else:

                prob = prob \* 0.2

                weatherToday = "Rainy"

                weatherList.append("Rainy")

    print("Possible states: " + str(weatherList))

    print("End state after "+ str(days) + " days: " + weatherToday)

    print("Probability of the possible sequence of states: " + str(prob))

def Markov\_chain\_stationary\_distribution(P):

    state = np.array([[1.0, 0.0, 0.0]])

    stateHist = state

    dfStateHist = pd.DataFrame(state)

    distr\_hist = [[0 ,0, 0]]

    for x in range(50):

        state = np.dot(state, P)

        # print(state)

        stateHist = np.append(stateHist, state, axis=0)

        dfDistrHist = pd.DataFrame(stateHist)

    print("Stationary Distribution (Pi vector):", state[0])

    dfDistrHist.plot()

    plt.grid()

    plt.title('Stationary Distribution')

    plt.legend(['Sunny', 'Cloudy', 'Rainy'])

    plt.show()

### Main Function ###

# inputWeather = input("Please input the weather today, type in \"Sunny\", \"Cloudy\", or \"Rainy\": ")

# days = int(input("Please input how many days you want to forecast: "))

weather\_forecast(days = 5, startWeather = "Sunny")

### 判斷穩態分佈存在的條件

# 1. 不可約性 Irreducible: 任何狀態都可能轉移到任意狀態

# 2. 非週期性 Aperiodic: 才會有穩態存在

# 3. 時間均匀性 Homogeneous

# 4. 有限狀態 Finite States

Markov\_chain\_stationary\_distribution(P = transitionMatrix)

# Excercise3

