Scientific Computing with Python Lab

14th Session(April 30th)

Kaheon Kim : https://github.com/kaheonkim/Scientific-computing-with-python-lab-material

Notice

- No office hour on this Thursday
- Because it’s final, I cannot answer to the details of your python script.
- But I can give you some brief introduction of the problem or provide some reference of it.

Things to do today(Final session)

Final : Due May 7th(Tue)

- Explain about the packages : numpy, matplotlib, csv

- Go through some of the problems : just give brief explanation

Numpy

import numpy as np

- I can tell it’s most important package in python!
- If you are continuing programming(in other courses or in your future career), you may be find yourself starting your code with stating : import numpy as np
- There are many important functions of numpy:

np.linspace, np.array, np.dot …

- Review the example code distributed on the canvas!

Stochastic Simulation

Random function : import numpy as np

- np.random.random() : generate number between 0 and 1
- np.random.choice(list) : choose 1 of the elements from the list uniformly

⇒ The output varies in each simulation as it’s stochastic(not deterministic)

Stochastic Simulation

Monte Carlo

- Stochastic Simulation :

After assuming the probability, we simulate based on the distribution → It bears uncertainty!

Ex. Let’s say A beats B with probability 0.7. If they fight for 10000 times, How much times A beats B?

Visualization

matplotlib -> import matplotlib.pyplot as plt

plt.(x_values,y_values, other_options)

- length of x_values and y_values should be same
- several options for plotting : shape, color, label, legend, title…
- ends with plt.show()

Visualization

matplotlib -> import matplotlib.pyplot as plt

plt.(x_values,y_values, other_options)

- length of x_values and y_values should be same
- several options : grid, labels, legend, drawing_options

CSV

import csv

Similar to the txt case, but has some difference in processing them

Like csv file,

- You have to accept variables in a row wise manner

Different from txt file,

- It doesn't have some symbols locating between numbers
- It may have the head row to signify the elements

Let’s go into the Problems of Finals

Problem 1

1. In this problem, we will produce a plot of a function on the interval $[-1, 1]$ given as an expansion in terms of Legendre polynomials.

   The Legendre polynomials are a sequence of polynomials satisfying:

$$\begin{aligned}
P_0(x) &= 1 \\
P_1(x) &= x \\
P_2(x) &= \frac{3}{2}x^2 - \frac{1}{2} \\
P_3(x) &= \frac{5}{2}x^3 - \frac{3}{2}x \\
&\vdots \\
P_n(x) &= 2^n \sum_{k=0}^{n} \binom{n}{k} \binom{\frac{n+k-1}{2}}{n} x^k
\end{aligned}$$

   (where the generalized binomial coefficients have been used in the above expansion).

   For this problem, write a Python script to open a file named "final_exam_problem_1_input.txt". The first line of the file consists of a single integer M representing the number of equally-spaced points over the interval $[-1, 1]$ at which to plot the function. The second line is blank. The remaining lines consist, in order, of the coefficients b_n (as floats, with one per line, beginning with b_0) of the expansion of a function $f(x) = \sum_{n=0}^{N} b_n P_n(x)$.

   Your task is to read from this file, and use it to plot the function $f(x)$ defined by the coefficients appearing in the file, over the interval $[-1, 1]$ (using equally-spaced x values). See Canvas for examples of what the input and the produced graph should look like.

Problem 2

2. In this problem, we will approximate the expected number of flips of a coin until a desired pattern (approximately) appears.

   In a file named "final_exam_problem_2_input.txt", the first line consists of the desired pattern, with heads and tails represented by H and T respectively, with no space between. The second line contains a float representing the probability p of flipping heads on a given flip (so that $1 - p$ is the probability of flipping tails). The third line consists of an integer representing the threshold of the number of agreeing flips (see below) to constitute an approximate match, and the fourth line consists of the number of trials to perform in your simulation.

   As an example, suppose the desired approximate pattern were $HHTH$, and the last four flips of the coin were $HHTT$. Then this would constitute a match if the threshold for matching were 0, 1, 2 or 3, (since three out of four flips agree in order), but would not constitute a match if the threshold were 4 (since at least one flip disagrees). Similarly, for the same desired pattern $HHTH$, the flips $THTH$ would constitute a match if the threshold for matching were 0, 1, or 2, but not if the threshold were 3 or 4 (since only two of the flips agree).

   Now, suppose the coin is repeatedly flipped until the desired pattern is (approximately) matched among the most recent flips (given the threshold for constituting an approximate match). For example, if the pattern were $HHTH$, with a threshold of 3 (requiring at least 3 out of four flips to match), and the coin were repeatedly flipped, obtaining the sequence of flips $TTTTHTH$, then 7 total flips were required until an approximate match was obtained in the last four flips (at no prior point in this sequence was an approximate match obtained sooner, as none of $TTTT$, $TTTH$, $TTHT$ constitute an approximate match). Note that the coin must be flipped at least 4 times in this scenario before it is possible to obtain an approximate match, as we will require an approximate match to have the same length as the desired pattern.

   Your task is to write a simulation to determine the expected number of flips until an approximate match to the desired pattern is obtained. Run your simulation for the indicated number of trials, and report the average number of trials needed to obtain the pattern to the user (as a float, do not round the calculated average). See Canvas for examples of what the input and results might look like.

Problem 4

4. In this problem, we will simulate the random movement of a king from the game of chess. A king starts in the lower left corner of an otherwise empty $M \times M$ chessboard. The king moves in turns, and, on each turn, may move to any adjacent space (horizontally, vertically, or diagonally as in the usual rules of chess), with the move selected uniformly at random from all available legal moves.

   Write a Monte Carlo simulation to determine the expected number of spaces visited by the king if it moves for N total turns. Note that the lower left corner, where the king starts, counts as having been visited, so that after the king has moved once, it will have visited exactly two spaces (the starting space, and whichever space it moved to).

   Your simulation should read data from a file named "final_exam_problem_4_input.txt". The first line of the file will specify the value M indicating the size of the board, the second line will specify the number N of moves to be made by the king, and the final line will specify the number of trials to conduct. Report to the user the approximate expected number of distinct spaces visited (as a float, do not round the calculated average).