FE-520 Assignment 3

Zhiyuan Yao, Zhi Chen

Spring 2022

Submission Requirement:

For all the problems in this assignment you need to design and use Python 3, output and present the results in nicely format. Please submit a .py file, if you prefer to use jupyter notebook, please download your .ipynb file as .html file to submit. You are strongly encouraged to write comment for your code, because it is a convention to have your code documented all the time. In your python file, you need contain both function and test part of function. Python script must be a '.py' script, Jupyter notebook '.ipynb is not allowed. Do NOT copy and paste from others, all homework will be firstly checked by plagiarism detection tool.

1 Moving Average 30 Pts

Given a stream of integers and a window size, calculate the moving average of all integers in the sliding window.

Implement the MovingAverage class:

1. MovingAverage(int size) Initializes the object with the size of the window size.
2. next(int val) Returns the moving average of the last size values of the stream.

Example:
Input
["MovingAverage", "next", "next", "next", "next"]
[[3], [1], [10], [3], [5]]
Output
[null, 1.0, 5.5, 4.66667, 6.0]
Explanation MovingAverage movingAverage = new MovingAverage(3);
movingAverage.next(1); // return 1.0 = 1 / 1
movingAverage.next(10); // return 5.5 = (1 + 10) / 2
movingAverage.next(3); // return 4.66667 = (1 + 10 + 3) / 3
movingAverage.next(5); // return 6.0 = (10 + 3 + 5) / 3

2 Implement the Subway Class 40 pts

Implement a class for subway system with following requirement:

1. checkIn(int id, string stationName, int t)
    - A customer with a card id equal to id, gets in the station stationName at time t.
    - A customer can only be checked into one place at a time.
2. checkOut(int id, string stationName, int t)
    - A customer with a card id equal to id, gets out from the station stationName at time t.
3. getAverageTime(string startStation, string endStation)
    - Returns the average time to travel between the startStation and the endStation.
    - The average time is computed from all the previous traveling from startStation to endStation that happened directly.
    - Call to getAverageTime is always valid.

You can assume all calls to checkIn and checkOut methods are consistent. If a customer gets in at time t1 at some station, they get out at time t2 with t2 > t1. All events happen in chronological order.

Example 1:

Input

[”Subway”,”checkIn”,”checkOut”,”getAverageTime”,
”checkIn”,”checkOut”,”getAverageTime”,”checkIn”,”checkOut”,”getAverageTime”]
[[],[10,”Leyton”,3],[10,”Paradise”,8],[”Leyton”,”Paradise”],[5,”Leyton”,10],
[5,”Paradise”,16],[”Leyton”,”Paradise”],[2,”Leyton”,21],[2,”Paradise”,30],
[”Leyton”,”Paradise”]]

Output

[null,null,null,5.00000,null,null,5.50000,null,null,6.66667]

Explanation

Subway Subway = new Subway();
Subway.checkIn(10, ”Leyton”, 3);
Subway.checkOut(10, ”Paradise”, 8);
Subway.getAverageTime(”Leyton”, ”Paradise”); // return 5.00000
Subway.checkIn(5, ”Leyton”, 10);
Subway.checkOut(5, ”Paradise”, 16);
Subway.getAverageTime(”Leyton”, ”Paradise”); // return 5.50000
Subway.checkIn(2, ”Leyton”, 21);
Subway.checkOut(2, ”Paradise”, 30);
Subway.getAverageTime(”Leyton”, ”Paradise”); // return 6.66667

Example 2:

Input
["Subway","checkIn","checkIn","checkIn","checkOut",
"checkOut","checkOut","getAverageTime","getAverageTime","checkIn",
"getAverageTime","checkOut","getAverageTime"]
[[],[45,"Leyton",3],[32,"Paradise",8],[27,"Leyton",10],
[45,"Waterloo",15],[27,"Waterloo",20],[32,"Cambridge",22],
["Paradise","Cambridge"],["Leyton","Waterloo"],[10,"Leyton",24],
["Leyton","Waterloo"],[10,"Waterloo",38],["Leyton","Waterloo"]]
Output
[null,null,null,null,null,null,null,14.00000,11.00000, null,11.00000,null,12.00000]
Explanation
Subway Subway = new Subway();
Subway.checkIn(45, "Leyton", 3);
Subway.checkIn(32, "Paradise", 8);
Subway.checkIn(27, "Leyton", 10);
Subway.checkOut(45, "Waterloo", 15);
Subway.checkOut(27, "Waterloo", 20);
Subway.checkOut(32, "Cambridge", 22);
Subway.getAverageTime("Paradise", "Cambridge"); // return 14.00000. There was only one travel from "Paradise" (at time 8) to "Cambridge" (at time 22)
Subway.getAverageTime("Leyton", "Waterloo"); // return 11.00000. There were two travels from "Leyton" to "Waterloo", a customer with id=45 from time=3 to time=15 and a customer with id=27 from time=10 to time=20. So the average time is ((15-3) + (20-10)) / 2 = 11.00000
Subway.checkIn(10, "Leyton", 24);
Subway.getAverageTime("Leyton", "Waterloo"); // return 11.00000
Subway.checkOut(10, "Waterloo", 38);
Subway.getAverageTime("Leyton", "Waterloo"); // return 12.00000

3 Linear Regression 30 pts

Continue with the last problem in assignment 2, in this problem you should do the same thing to implement linear regression in class.

1. Initialize this class with x and y as input.
2. Constructor should also take m, c, epochs(number of iterations), L(learning rate) as input, and set default values.
3. Define a gradient descent function to find m and c
4. Write a predictive function to predict y based on new input x (where x is a 2D list).

4 Bonus: Uniform Distributed Random Number Generator (20pt)

You can not use any random number generators in packages like Numpy, because you are required to build your own generator in this question.

Almost all the random numbers generated by computer are pseudo random numbers, because they are generated by a formula. A good random number generator is very important for financial simulations, such as Monte-Carlo method.

The goal of this assignment is to let you understand how computer generate random numbers and create pseudo random number generators which can generate 0-1 uniform-distributed random numbers by yourselves.

The probability density function (PDF) of 0-1 uniform distribution is

$$f(x) = \begin{cases} 1 & for\ x \in [0,1] \\ 0 & otherwise \end{cases}$$

It means each point in interval $[0, 1]$ has the same probability to be chosen.

1. For implementation purpose, people use different algorithms to generate pseudo-random numbers. One of the most famous algorithms is called **Linear Congruential Generator**(LCG). This generator can yield a sequence of random numbers $\{X_i\}$. The recurrence relation of this algorithm is:

   $$X_{n+1} = (aX_n + c)\ mod\ M$$

   here a, c, M are all parameters. a is called multiplier which satisfies $0 < a < M$, c is called increment which satisfy $0 \leqslant c < M$, M is the modulus which is also stands for the maximum range of this sequence. You can find more information about the common parameters setting in the link above. There is another parameter X_0 which is so called **seed**. You can generate totally different sequence by different seeds. In the formula above *mod* stands for the modulus operation, for example: $7\ mod\ 3 = 1$ due to $7 = 2 \times 3 + 1$.
   As you can guess, this generator can generate a sequence of integers within $[0, M)$. To create a uniform distributed sequence, you only need to divide the sequence above by M.
   For example, now I have a setting: $X_0 = 1$, and $a = 1103515245,\ M = 2^{32},\ c = 12345$.

   $$\begin{aligned} X_1 &= (a \cdot X_0 + c)\ mod\ M = 1103527590 \\ X_2 &= (a \cdot X_1 + c)\ mod\ M = 25248852323 \\ &\cdots \end{aligned}$$

   Then the uniform distributed sequence will be

   $$\{\frac{1}{2^{32}}, \frac{1103527590}{2^{32}}, \frac{25248852323}{2^{32}}, \cdots\} \sim \{2.328 \times 10^{-10}, 0.2569, 0.5879, \cdots\}$$

There are some variants of LCG, one of them has such recurrence relation:

$$X_{n+1} = (X_n(X_n + 1)) \ mod \ M$$

The seed of this generator has to satisfy $X_0 \ mod \ 4 = 2$.
Now it's your turn to implement it in Python 3.

- Create a file called generator.py. In this file, create a class called **LCG** whose instance can generate random numbers by Linear Conguential Generator algorithm. It should have these data attributes: self.seed, self.multiplier, self.increment, self.modulus which are assigned in initializer (__init__()). The parameters should be passed by argument from outside of the class. This class's instance should also at least have function attributes allow me: 1) get the seed; 2) set the seed; 3) initialize the generator (start from the seed); 4) give me the next random number; 5) give me a sequence (list) of random number, the length should be passed by argument.

- A good code style like following good conventions, writing comment and doc string for your code will earn additional bonus.