Tutorial 02 CS384 - Memory Game

Dr. Mayank Agarwal

Assignment Given: 20th Aug 2021,
Deadline 23th August 2021, 23:59
Submission: GitHub

Things to be kept in mind

1. You should first check that the list does not contain any floating numbers or characters. The list should have only integers as input. If a non int is present, just output: "Please enter a valid input list" and show all the elements that was invalid. E.g, if input_nums = [1,4, "a", 3, 5, "Star", 7.5], then output will be: "Please enter a valid input list". Invalid inputs detected": ["a", "Star", 7.5] and exit the code. You can use isdigit() function also.
2. While evaluating your program, the TA we will modify the input list (input_nums), and check for correctness. End user will not enter any input via keyboard, the input will be static in nature, input_nums = []
3. Program will be checked for plagiarism.

Returns the score from a memory game. The strategy is to remove the number that has been in the memory the longest time.

A memory game is played (and scored) as follows: Random numbers between 0 and 10 (zero inclusive) are called out one at a time. In this memory game the player can remember a maximum of 5 previously called out numbers. If the called number is already in the player's memory, a point is added to the player's score. If the called number is not in the player's memory, the player adds the called number to his memory, first removing another number if his memory is full. In our simulation of this game, the number which is removed from the player's memory is the number that has been in the player's memory the longest time. For example, if the random numbers are [3, 4, 3, 0, 7, 4, 5, 2, 1, 3], the game proceeds as follows:
Called number 3: Score: 0, Numbers in memory: [3]
Called number 4: Score: 0, Numbers in memory: [3, 4]
Called number 3: Score: 1, Numbers in memory: [3, 4]
Called number 0: Score: 1, Numbers in memory: [3, 4, 0]
Called number 7: Score: 1, Numbers in memory: [3, 4, 0, 7]
Called number 4: Score: 2, Numbers in memory: [3, 4, 0, 7]
Called number 5: Score: 2, Numbers in memory: [3, 4, 0, 7, 5]
Called number 2: Score: 2, Numbers in memory: [4, 0, 7, 5, 2]
Called number 1: Score: 2, Numbers in memory: [0, 7, 5, 2, 1]
Called number 3: Score: 2, Numbers in memory: [7, 5, 2, 1, 3]

Complete the get_memory_score() function which is passed a list of random numbers as a parameter and returns the final score using the algorithm described above. For example, the following code:

```
print("1. Score:", get_memory_score([3, 4, 1, 6, 3, 3, 9, 0, 0, 0]))
print("2. Score:", get_memory_score([1, 2, 2, 2, 2, 3, 1, 1, 8, 2]))
print("3. Score:", get_memory_score([2, 2, 2, 2, 2, 2, 2, 2, 2]))
print("4. Score:", get_memory_score([1, 2, 3, 4, 5, 6, 7, 8, 9]))
input_nums = [7, 5, 8, 6, 3, 5, 9, 7, 9, 7, 5, 6, 4, 1, 7, 4, 6, 5, 8, 9, 4, 8, 3, 0, 3]
print("5. Score:", get_memory_score(input_nums))
```

prints:

1. Score: 4
2. Score: 6
3. Score: 8
4. Score: 0
5. Score: 10

Sample input output

Input

```
input_nums = [3, 4, 1, 6, 3, 3, 9, 0, 0, 0]
```

Output

```
Score: 4
```