Moving Average from Data Stream

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

For example,

```
MovingAverage m = new MovingAverage(3);
m.next(1) = 1
m.next(10) = (1 + 10) / 2
m.next(3) = (1 + 10 + 3) / 3
m.next(5) = (10 + 3 + 5) / 3
```

Solution 1

```
import collections

class MovingAverage(object):

    def __init__(self, size):
        """
        Initialize your data structure here.
        :type size: int
        """
        self.queue = collections.deque(maxlen=size)


    def next(self, val):
        """
        :type val: int
        :rtype: float
        """
        queue = self.queue
        queue.append(val)
        return float(sum(queue))/len(queue)



# Your MovingAverage object will be instantiated and called as such:
# obj = MovingAverage(size)
# param_1 = obj.next(val)
```

written by microleo720 (https://leetcode.com/discuss/user/microleo720) original link here (https://leetcode.com/discuss/100373/4-line-python-solution-using-deque)

Solution 2

```
class MovingAverage {
private:
    queue<int> qu;
    int avergeSize;
    double sum;
public:
    MovingAverage(int size):avergeSize(size),sum(0) {}

    double next(int val) {
        sum += val;
        qu.push(val);
        int queueSize = (int)qu.size();
        if(queueSize <= avergeSize){
            return sum / queueSize;
        }
        else{
            sum -= qu.front();
            qu.pop();
            return sum / avergeSize;
        }
    }
};
```

written by sxycwzwzq (https://leetcode.com/discuss/user/sxycwzwzq) original link here (https://leetcode.com/discuss/100352/c-easy-solution-using-queue)

Solution 3

Fixed size array is enough for this problem.

Java:

```
public class MovingAverage {
    int[] window;
    int index = 0;
    /** Initialize your data structure here. */
    public MovingAverage(int size) {
        this.window = new int[size];
    }

    public double next(int val) {
        this.window[index] = val;
        index = (index+1)%this.window.length;
        double ans = 0.0;
        for (int i:window) {
            ans += i;
        }
        return ans/this.window.length;
    }
}
```

Python:

```
class MovingAverage(object):

    def __init__(self, size):
        """
        Initialize your data structure here.
        :type size: int
        """
        self.size = size;
        self.window = [0]*size;
        self.index = 0;
    def next(self, val):
        """
        :type val: int
        :rtype: float
        """
        self.window[self.index] = val
        self.index = (self.index+1)%self.size
        return float(sum(self.window))/self.size
```

written by sirxudi (https://leetcode.com/discuss/user/sirxudi) original link here (https://leetcode.com/discuss/101057/java-and-python-fixed-size-array)