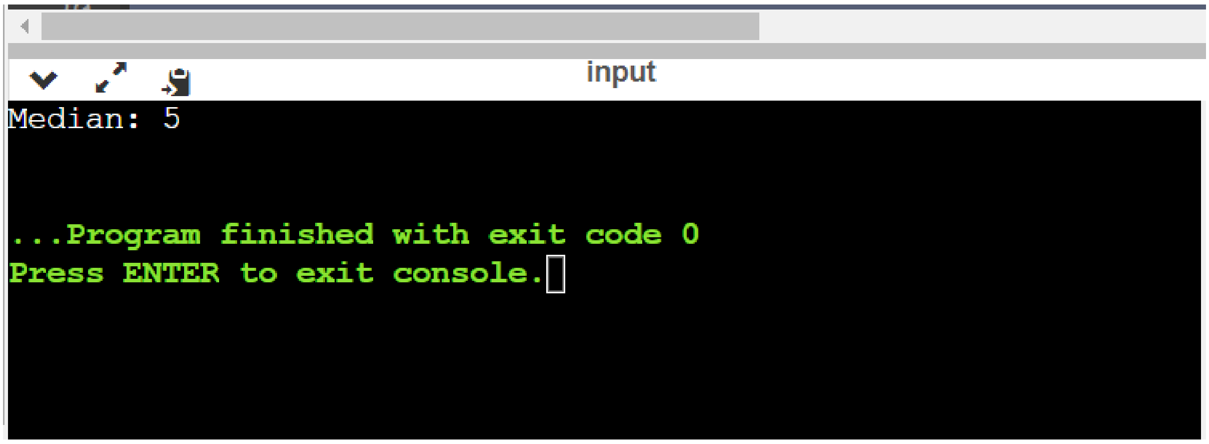
Problem 2: Find median in a stream of running integer

```
    class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left = None
        self.right = None



class MedianFinder:
    def __init__(self):
        self.root = None
        self.count = 0

    def addNum(self, num):
        if self.root is None:
            self.root = TreeNode(num)
        else:
            self._insert(self.root, num)

        self.count += 1

    def _insert(self, node, num):
        if num < node.val:
            if node.left is None:
                node.left = TreeNode(num)
            else:
                self._insert(node.left, num)
        else:
            if node.right is None:
                node.right = TreeNode(num)
            else:
                self._insert(node.right, num)

    def findMedian(self):
        if self.count == 0:
            return None

        if self.count % 2 == 1:
            return self._find_kth_smallest(self.root, (self.count + 1) // 2)
        else:
            left = self._find_kth_smallest(self.root, self.count // 2)
            right = self._find_kth_smallest(self.root, self.count // 2 + 1)
            return (left + right) / 2

    def _find_kth_smallest(self, node, k):
        left_count = self._count_nodes(node.left)

        if k == left_count + 1:
            return node.val
        elif k <= left_count:
            return self._find_kth_smallest(node.left, k)
        else:
            return self._find_kth_smallest(node.right, k - left_count - 1)

    def _count_nodes(self, node):
        if node is None:
            return 0
        return 1 + self._count_nodes(node.left) + self._count_nodes(node.right)

mf = MedianFinder()
mf.addNum(5)
mf.addNum(10)
mf.addNum(1)

mf.addNum(7)

mf.addNum(3)

median = mf.findMedian()

print("Median:", median)
```



```
Median: 5


...Program finished with exit code 0
Press ENTER to exit console.
```