Week-4 Practical

Task 1: (R-6.1) What values are returned during the following series of stack operations, if executed upon an initially empty stack? push(5), push(3), pop(), push(2), push(8), pop(), pop(), push(9), push(1), pop(), push(7), push(6), pop(), pop(), push(4), pop(), pop().

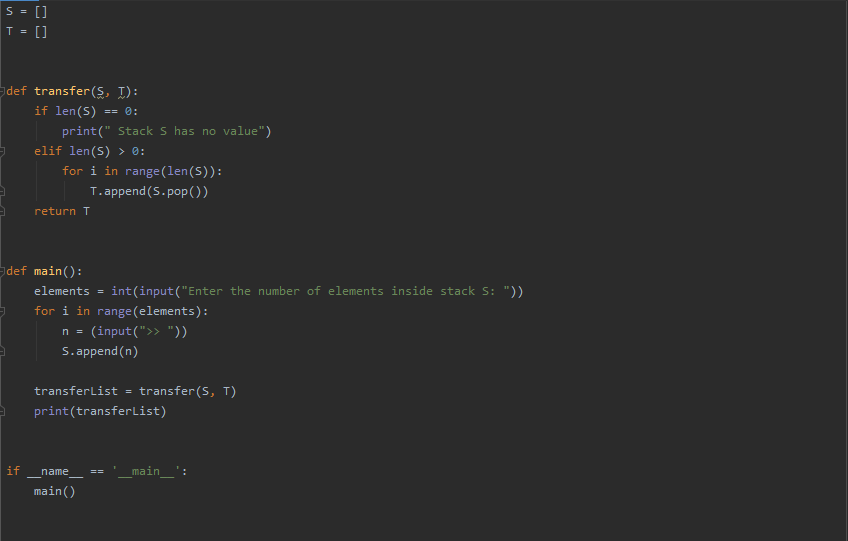
|  |  |  |
| --- | --- | --- |
| Operation | Output | Stack content |
| push(5) | \_ | [5] |
| push(3) | \_ | [5, 3] |
| pop() | 3 | [5] |
| push(2) | \_ | [5, 2] |
| push(8) | \_ | [5, 2, 8] |
| pop() | 8 | [5, 2] |
| pop() | 2 | [5] |
| push(9) | \_ | [5, 9] |
| push(1) | \_ | [5, 9, 1] |
| pop() | 1 | [5, 9] |
| push(7) | \_ | [5, 9, 7] |
| push(6) | \_ | [5, 9, 7, 6] |
| pop() | 6 | [5, 9, 7] |
| pop() | 7 | [5, 9] |
| push(4) | \_ | [5, 9, 4] |
| pop() | 4 | [5, 9] |

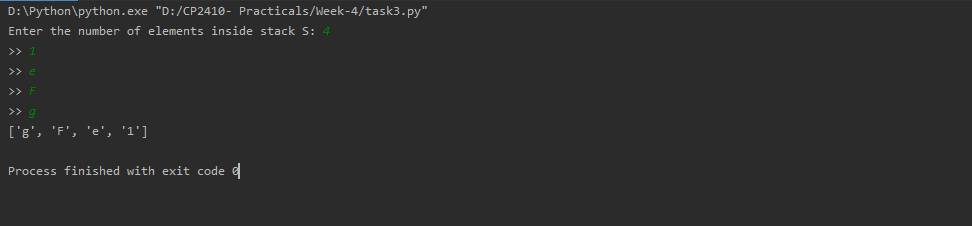
Task 2: (R-6.2) Suppose an initially empty stack S has executed a total of 25 push operations, 12 top operations, and 10 pop operations, 3 of which raised Empty errors that were caught and ignored. What is the current size of S?

* You popped a total number of 10-3 = 7 elements, since 3 of the pops did not change the state (and size) of the stack, so only 7 pops did.
* You pushed a total of 25 elements.
* Top operations do not change the state (and size) of the stack, and can be ignored.

**Total size of the stack is 25-7 = 18 at the end.**

Task 3: (R-6.3) Implement a function transfer(S, T) that transfers all elements from stack S onto stack T, so that the element that starts at the top of S is the first to be inserted onto T, and the element at the bottom of S ends up at the top of T. Use ArrayStack (ch06/array\_stack.py) to test your function.





Task 4: (R-6.7) What values are returned during the following sequence of queue operations, if executed on an initially empty queue? enqueue(5), enqueue(3), dequeue(), enqueue(2), enqueue(8), dequeue(), dequeue(), enqueue(9), enqueue(1), dequeue(), enqueue(7), enqueue(6), dequeue(), dequeue(), enqueue(4), dequeue(), dequeue()

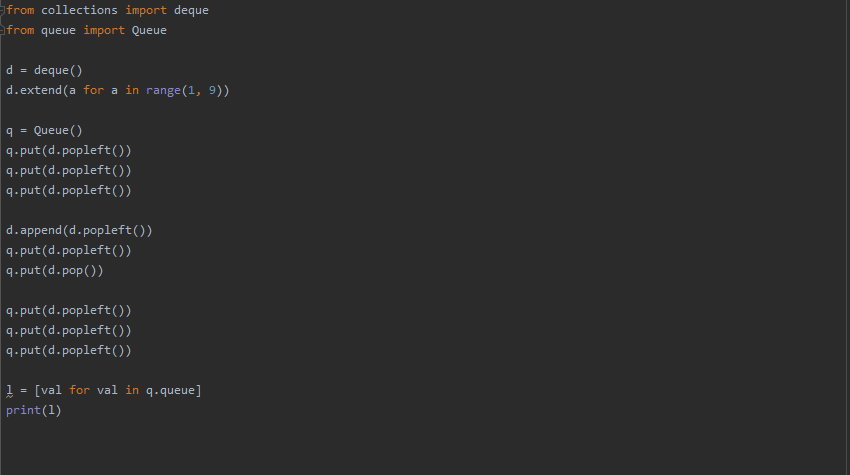
|  |  |  |
| --- | --- | --- |
| Operation | Output | Stack content |
| enqueue(5) | \_ | [5] |
| enqueue(3) | \_ | [5,3] |
| dequeue() | 5 | [3] |
| enqueue(2) | \_ | [3,2] |
| enqueue(8) | \_ | [3,2,8] |
| dequeue() | 3 | [2,8] |
| dequeue() | 2 | [8] |
| enqueue(9) | \_ | [8,9] |
| enqueue(1) | \_ | [8,9,1] |
| dequeue() | 8 | [9,1] |
| enqueue(7) | \_ | [9,1,7] |
| enqueue(6) | \_ | [9,1,7,6] |
| dequeue() | 9 | [1,7,6] |
| dequeue() | 1 | [7,6] |
| enqueue(4) | \_ | [7,6,4] |
| dequeue() | 7 | [6,4] |

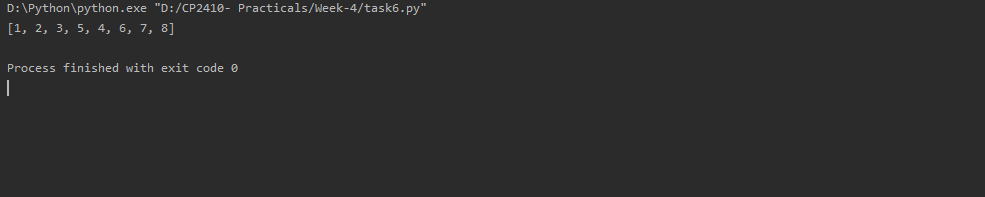
Task 5: (R-6.8) Suppose an initially empty queue Q has executed a total of 32 enqueue operations. 15 dequeue operations were also executed, 5 of which raised Empty errors that were caught and ignored. What is the current size of Q?

* You dequeued a total number of 15-5 = 10 elements, since 5 of the dequeue did not change the state (and size) of the stack, so only 10 dequeue did.
* You enqueued a total of 32 elements.

**Total size of the stack is 32-10 = 22 at the end.**

Task 6: (R-6.13) Suppose you have a deque D containing the numbers (1,2,3,4,5,6,7,8), in this order. Suppose further that you have an initially empty queue Q. Give a code fragment that uses only D and Q (and no other variables) and results in D storing the elements in the order (1,2,3,5,4,6,7,8).





Task 7: (R-6.14) Repeat the previous problem using the deque D and an initially empty stack S.



