For both Stacks and Queue.__str__ have a worst case performance of O(n), since the program has to loop through the structures to get the string representation.

For both Stacks and Queue .__len have a worst case performance of O(1), since the methods just return a parameter.

For Array_Deque, push runs at O(n) since the element gets added to the front of the list, while for the Linked_List_Deque push runs at O(1) since it takes the same time to insert a node at index 0 in all cases.

Pop is an O(n) method for the Array_Deque since we are removing the first element and then the code manually shifts over each element to the left one index, resulting in O(n). Pop for Linked_List_Deque runs at O(1) since removing the first element and returning that element always runs at the same runtime.

Enque is an O(n) method for Array_deque since at worst case, the array has to be grown. Meanwhile enqueue for Linked_list_deque can be done in O(1) time since the push_back method for Linked_list_deque is in constant time.

Enque is an O(n) method for Array_deque since for arrays, each element has to be moved over. Meanwhile dequeue for Linked_list_deque can be done in O(1) time since the pop_back method for Linked_list_deque is in constant time.

Peek for both representations of both Stack of Queue can be done in O(1), since it is a matter of accessing the element at index 0, which can be done in constant time.

Personally, I think the idea to not raise exceptions is a mistake, since users may not realize that they're using the structure wrong if there is no error. This limits functionality because somebody could keep trying to remove elements, while there are none left, causing other errors in the code and confusion over the cause.

My test cases were to create a new stack and queue, and then just adding a few elements and then removing them, printing, and seeing the value of stack, pop, and peek at each step. This checks to see if all the methods work by testing the functionality and showing what the results are for the programmer to see, therefore making up a good set of test cases.