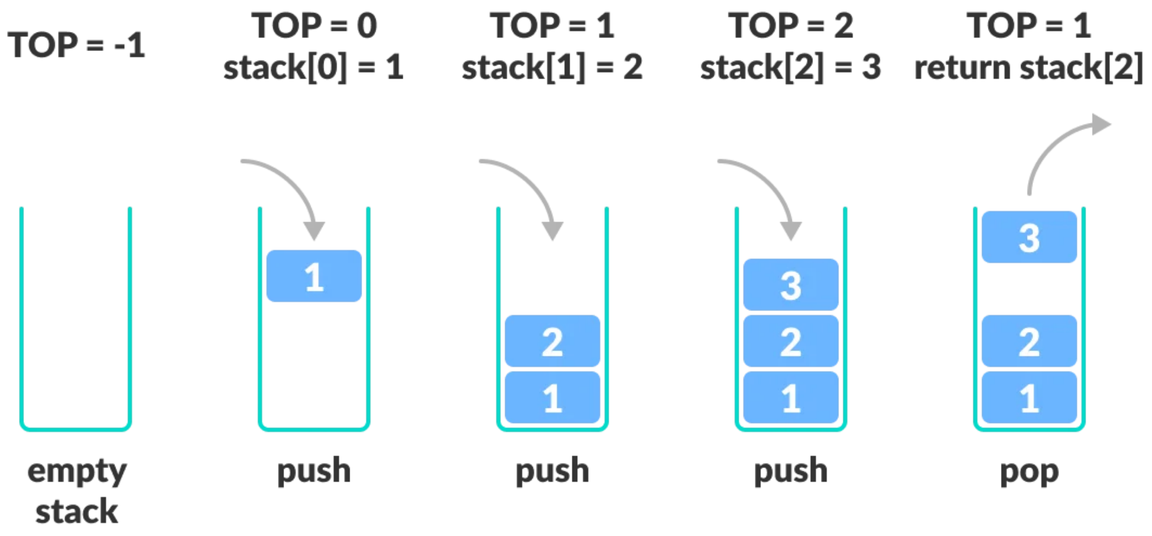
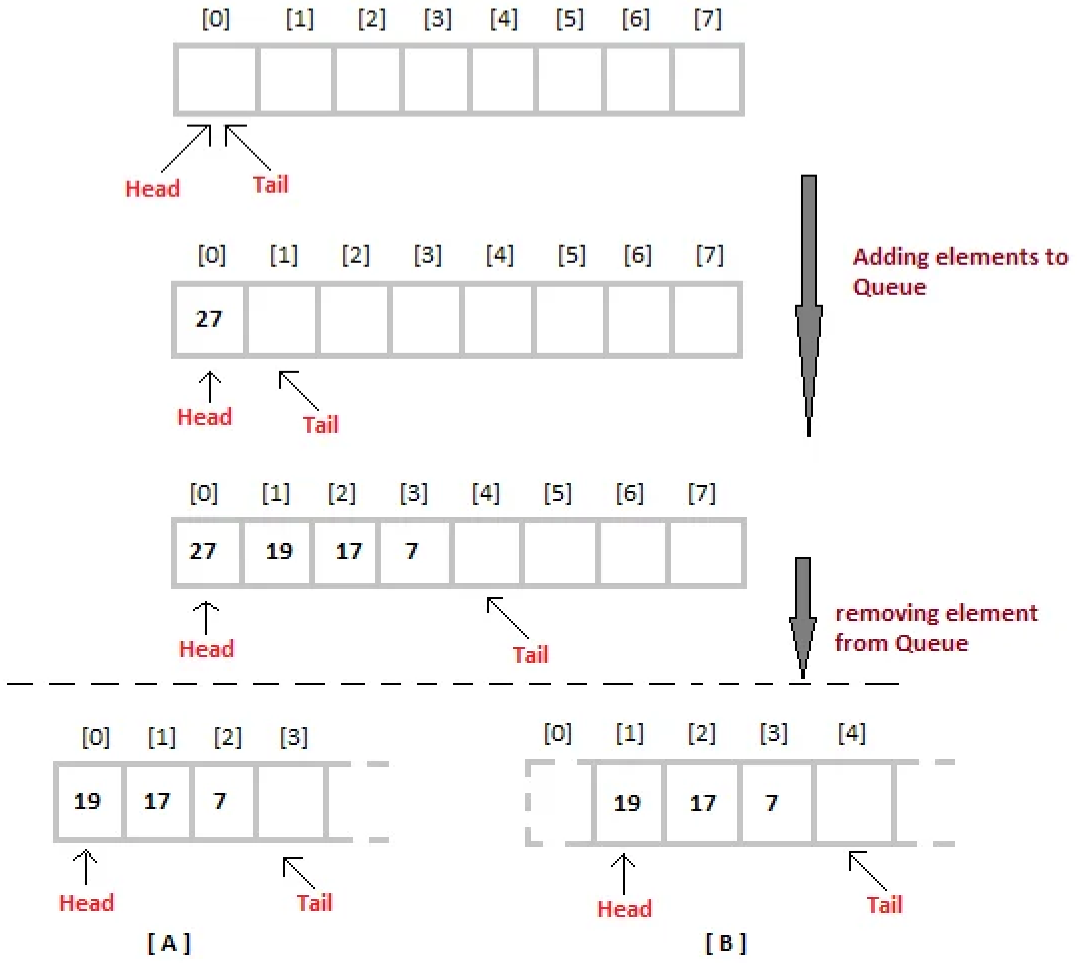
1. Describe Stack and Queue with illustrations.

1a) A stack is a data structure that is a FILO (first in last out) / LIFO (last in first out) implementation of a linked list. This means that each element appended onto the stack [push() function] will then be the first element in line to be removed from the structure [pop() function]. In other words pop() operates on the tail of the linked list, while the head of the linked list remains in place and untouched unless all other elements in the stack are untouched.

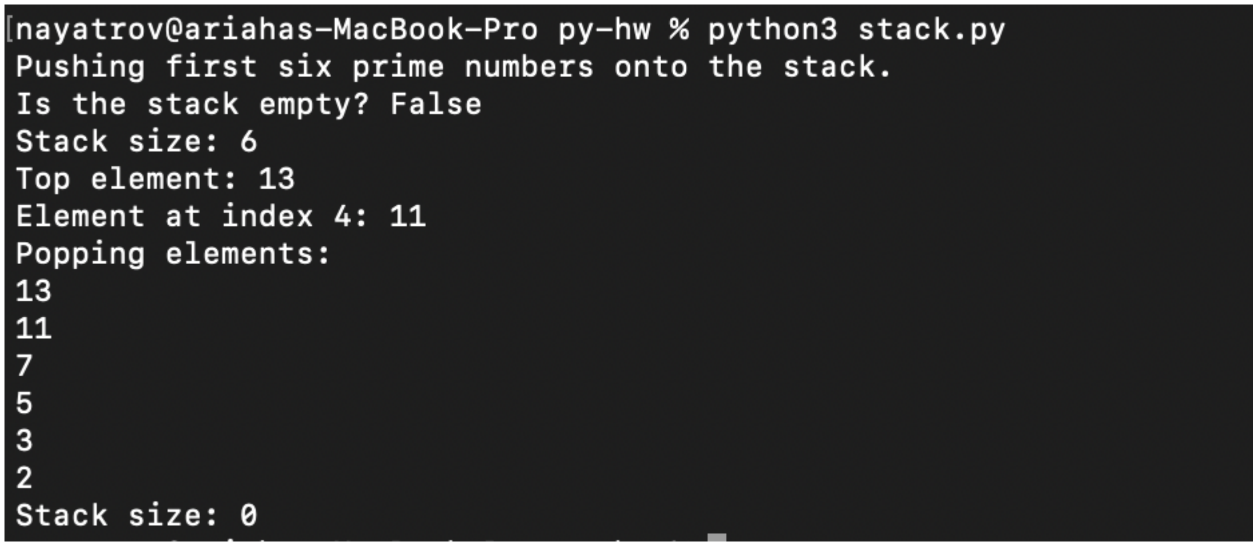


1b) A queue is a data structure that is a FIFO (first in first out) / LILO (last in last out) implementation of a linked list. This means that each element appended onto the queue [push() function] will then be the element last in line to be removed from the queue [via pop() function]. In other words pop() operates on the head of the linked list, removing it and setting the next element which the current head points to as the new head. It is the opposite of a stack in which the head remains unchanged unless the head is the only element in the structure.



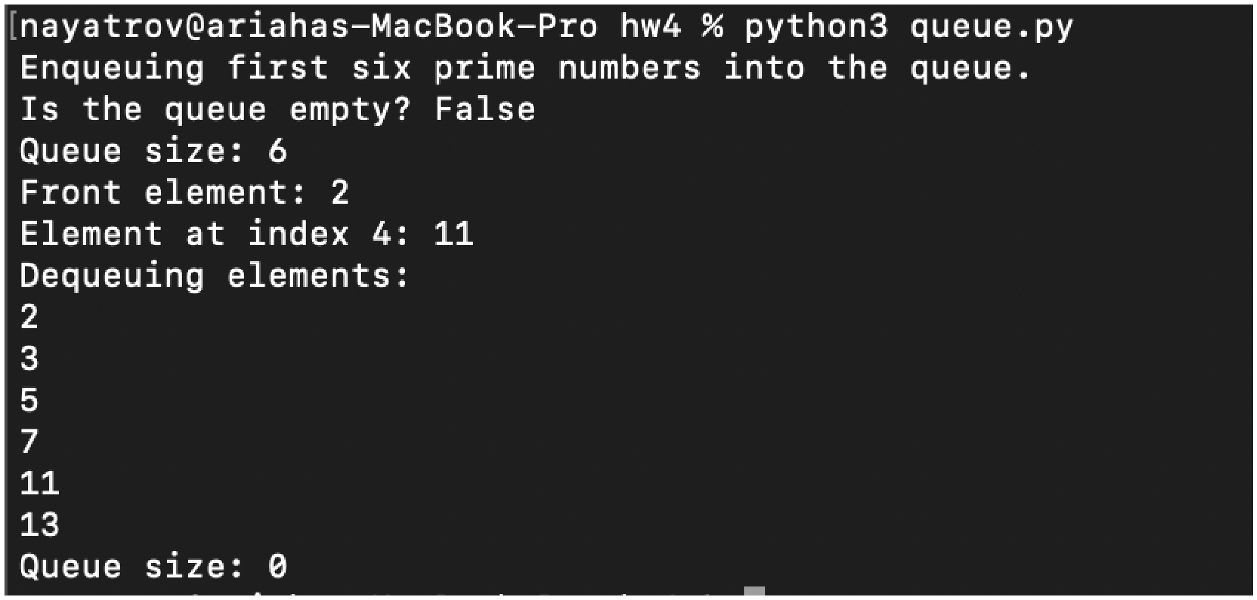
[0] [1] [2] [3] [4] [5] [6] [7]
Head
Tail
Adding elements to Queue
[0] [1] [2] [3] [4] [5] [6] [7]
27
Head
Tail
[0] [1] [2] [3] [4] [5] [6] [7]
27 19 17 7
Head
Tail
removing element from Queue
[0] [1] [2] [3]
19 17 7
Head
Tail
[A]
[0] [1] [2] [3] [4]
19 17 7
Head
Tail
[B]

2. Write code to implement a stack and its basic operations. (Code included in stack.py)
   [output for stack.py below]



```
[nayatrov@ariahas-MacBook-Pro py-hw % python3 stack.py
Pushing first six prime numbers onto the stack.
Is the stack empty? False
Stack size: 6
Top element: 13
Element at index 4: 11
Popping elements:
13
11
7
5
3
2
Stack size: 0
```

3. Write code to implement a queue and its basic operations. (Code included in queue.py)
   [output for queue.py below]



```
[nayatrov@ariahas-MacBook-Pro hw4 % python3 queue.py
Enqueuing first six prime numbers into the queue.
Is the queue empty? False
Queue size: 6
Front element: 2
Element at index 4: 11
Dequeuing elements:
2
3
5
7
11
13
Queue size: 0
```