**Part 1: Conceptual Questions**

**Linked Lists:**

1. **linked list is a sequence data structure, similar to lists, but its created by objects that represent each Node, and each node connected to the next node using next pointer and holding a value, these next and values are attributes defined by the class.**

**Linked list can be accessed by head, which represents the first node in list.**

1. doubly linked lists have additional pointer (previous) more over than next, that each Node will have 3 parameters (next, previous, value), so in terms of memory the single list is more efficient, and this previous pointer allows us to traverse the doubly list both forward and backward ways.

**Stacks:**

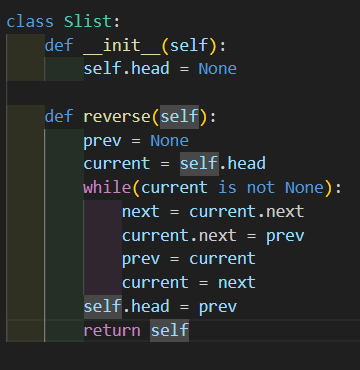
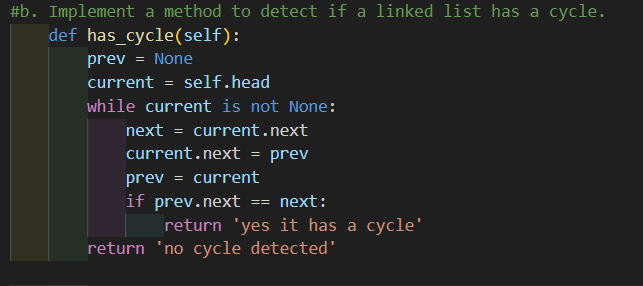
1. in stack data structure last element added to the list is the first element that we can get to do some processing on it, on windows (**ctrl + z**) holds a stack of last edit changes, so can get back to the most recent change using this feature.
2. Operations on stacks are (**POP, PUSH**), for history of calls you can use pop to get the last call you did, use push after ending current call to add to top of history stack.

**Queues:**

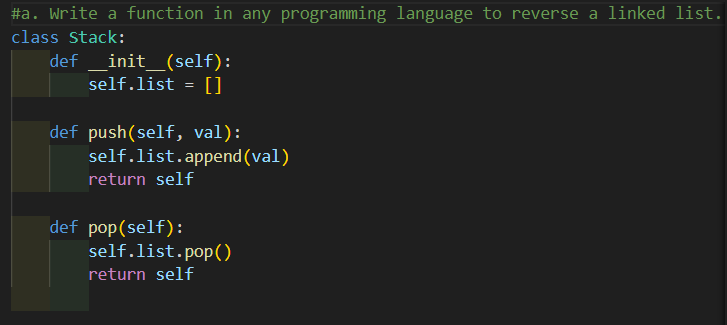
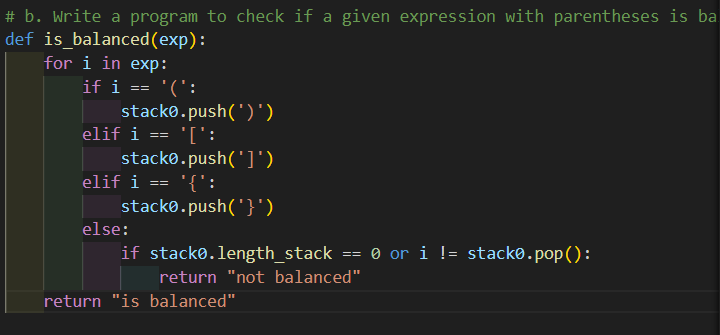
1. In a simple way we can say that CPU uses Queue to manage the all the processes in waiting list, this can be done using queue data structure so that the first activity added to the waiting list must be done firstly, then second one and so on.
2. Priority queue have the same interfaces (**Rear, Front**) as normal queue, but it follows different rule at which element can out first, these rules depend on priority of this node.

**Part 2: Practical Questions**

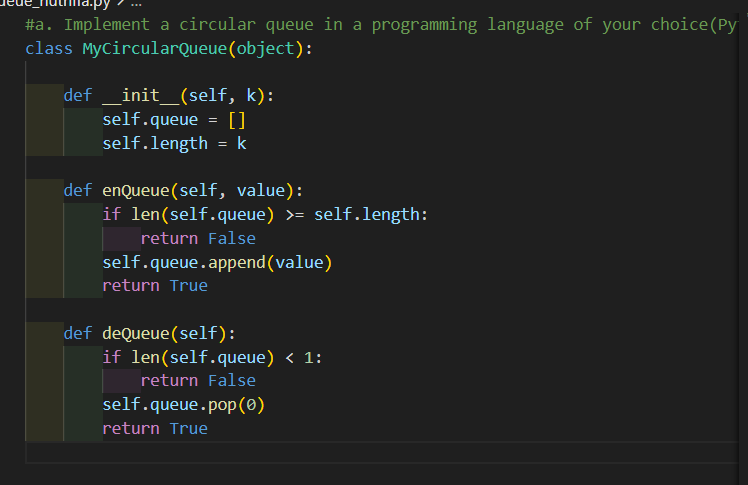
**Linked Lists:**

1. ****
2. ****

**Stacks:**

1. 
2. 

**Queues:**

1. 

**Part 3: Analysis**

1. Adding a new node at the end of linked list is O(1) complexity, no need for any shifting.

Also adding a new node at the beginning of linked list has no affect to all nodes, only the first node will no longer be the head, its also O(1).

1. Using my approach up here, **enqueue will results to increasing the array length with no complexity, but in the other hands dequeue an element causes the whole array to be shifted back (rearrange element with indexes), and this is O(n) logarithmic time complexity.**