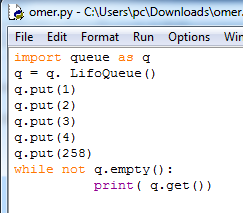
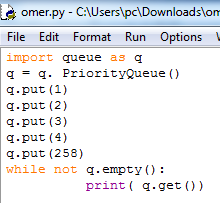
LAB # 04

**IMPLEMENTING PRIORITY QUEUE**

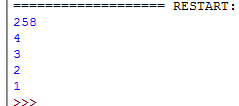
1.In contrast to the standard FIFO implementation of Queue, the LifoQueue uses last-in, first-out ordering (normally associated with a stack data structure). Implement LIFO queue using queue module

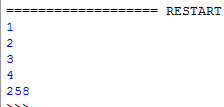
SOURCE CODE:





OUTPUT:

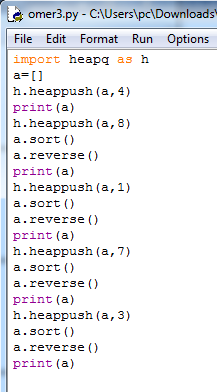




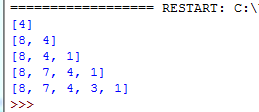
2.We have an array of 5 elements: [4, 8, 1, 7, 3] and we have to insert all the elements in the max-priority queue. First as the priority queue is empty, so 4 will be inserted initially. Now when 8 will be inserted it will move to front as 8 is greater than 4. While inserting 1, as it is the current minimum element in the priority queue, it will remain in the back of priority queue. Now 7 will be inserted between 8 and 4 as 7 is smaller than Now 3 will be inserted before 1 as it is the 2nd minimum element in the priority queue. All the steps are represented in the diagram below.



SOURCE CODE:

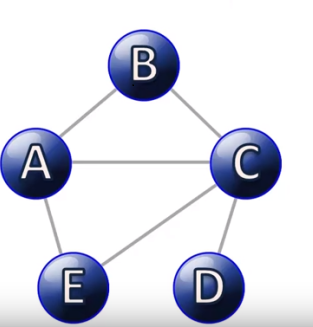


OUTPUT:

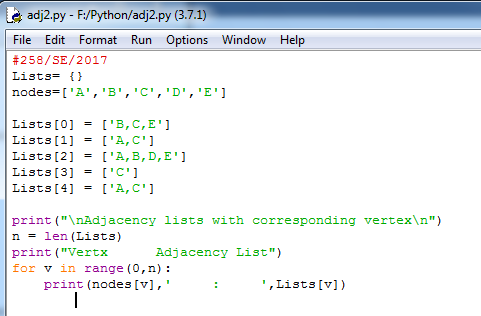
****

**Home Assignment:**

Implement graph using adjacency list using list or dictionary , make a class such as Vertex and Graph then make some function such as add\_nodes , add\_edges, add\_neighbors, add\_vertex, add\_vertices and suppose whatever you want to need it



SOURCE CODE:



OUTPUT:

