In the previous example, different data structures are used to implement a priority queue with customized priority. Let's break down the usage of each data structure:

1. **Class CustomItem:** This is a custom class created to represent the items that will be enqueued in the priority queue. It contains two attributes: item to store the actual item being enqueued, and priority to store the priority value associated with the item. By creating this custom class, we can define our own comparison logic for prioritization.
2. **queue.PriorityQueue:** This is a built-in class provided by the queue module in Python's standard library. It implements a priority queue data structure. The PriorityQueue class is used to create an instance of the priority queue. It automatically orders the items based on their priorities when enqueued and dequeues them in the order of priority.
3. **priority\_queue.put():** The put() method of the PriorityQueue class is used to enqueue items into the priority queue. In this example, instances of the CustomItem class are enqueued, and each item is associated with a priority value. The priority queue uses the defined comparison logic in the CustomItem class to order the items based on their priorities.
4. **priority\_queue.get():** The get() method of the PriorityQueue class is used to dequeue items from the priority queue. It removes and returns the item with the highest priority (i.e., the item with the lowest priority value). In this example, the items are dequeued one by one in ascending order of priority, and the associated item is printed.

By combining these data structures and methods, we create a priority queue where items are enqueued with custom priorities and dequeued based on their priority order. This allows us to process items in the priority order specified by the CustomItem class.