Control priority in priority queue

If you want a queue that returns items in insertion order, use a regular Queue implementation like LinkedList.

**class** PriorityQueueNode:

**def** \_\_init\_\_(self, value, pr):

    self.data **=** value

    self.priority **=** pr

    self.next **=** None

**class** PriorityQueue:

**def** \_\_init\_\_(self):

        self.front **=** None

**def** isEmpty(self):

**return** True **if** self.front **==** None **else** False

**def** push(self, value, priority):

**if** self.isEmpty() **==** True:

            self.front **=** PriorityQueueNode(value,

                                           priority)

**return** 1

**else**:

**if** self.front.priority > priority:

                newNode **=** PriorityQueueNode(value,

                                            priority)

                newNode.next **=** self.front

                self.front **=** newNode

**return** 1

**else**:

                temp **=** self.front

**while** temp.next:

**if** priority <**=** temp.next.priority:

**break**

                    temp **=** temp.next

                newNode **=** PriorityQueueNode(value,

                                            priority)

                newNode.next **=** temp.next

                temp.next **=** newNode

**return** 1

    # Method to remove high priority item

    # from the Priority Queue

**def** pop(self):

**return**

**else**:

            self.front **=** self.front.next

**return** 1

**def** peek(self):

**if** self.isEmpty() **==** True:

**return**

**else**:

**return** self.front.data

**def** traverse(self):

**if** self.isEmpty() **==** True:

**return** "Queue is Empty!"

**else**:

            temp **=** self.front

**while** temp:

                print(temp.data, end **=** " ")

                temp **=** temp.next

**if** \_\_name\_\_ **==** "\_\_main\_\_":

    pq **=** PriorityQueue()

    pq.push(4, 1)

    pq.push(5, 2)

    pq.push(6, 3)

    pq.push(7, 0)

    pq.traverse()

    pq.pop()