class Node:

    def \_\_init\_\_(self, data):

        self.data = data

        self.next = None

class Queue:

    def \_\_init\_\_(self):

        self.rear = None

        self.front = None

        self.count = 0

    def Enqueue(self, data):

        # Insert from rear

        new\_node = Node(data)

        #case 1: Queue is empty

        if self.rear is None:

            self.front = new\_node

        else:

            # case 2: Queue has some nodes

            # current node next is rear

            self.rear.next = new\_node

        self.rear = new\_node

        self.count += 1

    def Dequeue(self) -> int:

        if (self.front == None):

            print("Queue is Empty")

            return

        # take a copy of front value

        return\_data = self.front.data

        # move next node

        self.front = self.front.next

        # if there was only one elements , queue becomes empty

        if self.front == None:

            self.rear = None

        self.count -= 1

        print(f"Removed element: {return\_data}")

        return return\_data

    def peek(self):

        if self.front == None:

            print("Queue is Empty")

            return

        return self.front.data

    def get\_count(self) -> int:

        return self.count

    def print\_all\_elements(self):

        if self.front == None:

            print("Queue is Empty")

            return

        current\_node = self.front

        while current\_node is not None:

            print(f"{current\_node.data} --> ",end=" ")

            current\_node = current\_node.next

        print()

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

    queue = Queue()

    queue.Dequeue()

    queue.print\_all\_elements()

    queue.get\_count()

    queue.Enqueue(1)

    queue.print\_all\_elements()

    queue.get\_count()

    queue.Dequeue()

    queue.print\_all\_elements()

    queue.Enqueue(1)

    queue.Enqueue(2)

    queue.Enqueue(3)

    queue.Enqueue(4)

    queue.Enqueue(5)

    queue.print\_all\_elements()

    queue.get\_count()

    value = queue.Dequeue()

    print(f"Removed value {value} from Queue")

    queue.print\_all\_elements()