slip13

```
class Queue:
    def __init__(self):
            self.queue = []

    def enqueue(self, item):
        self.queue.append(item)
        print(f"Enqueued: {item}")

    def dequeue(self):
        if not self.is_empty():
            item = self.queue.pop(0)
            print(f"Dequeued: {item}")
            return item
        else:
            print("Queue is empty. Cannot dequeue.")
            return None

    def is_empty(self):
         return len(self.queue) == 0

    def display(self):
        if self.is_empty():
            print("Queue is empty.")
        else:
            print("Current Queue:", self.queue)
q = Queue()
q.enqueue(10)
```

```
q.enqueue(20)

q.enqueue(30)

q.display()  # Display the queue

q.dequeue()  # Remove the front element

q.display()  # Display the queue

q.dequeue()  # Remove another element

q.dequeue()  # Remove the last element

q.dequeue()  # Try to dequeue from an empty queue
```

```
import java.util.Scanner;

public class s13q2 {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);


      System.out.print("Enter your name: ");

      String name = scanner.nextLine();


      String upperCaseName = name.toUpperCase();

      System.out.println("Hello, " + upperCaseName + ", nice to meet you!");

      scanner.close();
   }
}
```

```
def povi():
    while True:
        try:
            n = int(input("Enter a positive number: "))
            if n < 0:
                print("The number is negative. Please enter a positive number.")
            else:
                print("Correct input:", n)
                break

        except ValueError:

            print("Invalid input. Please enter a valid integer.")
povi()
```

```
class Queue:
    def __init__(self):
            self.queue = []

    def enqueue(self, item):
        self.queue.append(item)
        print(f"Enqueued: {item}")

    def dequeue(self):
        if not self.is_empty():
            item = self.queue.pop(0)
            print(f"Dequeued: {item}")
```

```
            return item
        else:
            print("Queue is empty. Cannot dequeue.")
            return None

    def is_empty(self):
        return len(self.queue) == 0

    def display(self):
        if self.is_empty():
            print("Queue is empty.")
        else:
            print("Current Queue:", self.queue)
q = Queue()
q.enqueue(10)
q.enqueue(20)
q.enqueue(30)

q.display()  # Display the queue

q.dequeue()  # Remove the front element
q.display()  # Display the queue

q.dequeue()  # Remove another element
q.dequeue()  # Remove the last element
q.dequeue()  # Try to dequeue from an empty queue
```