Circular Buffer Code

Introduction:

In this assignment, I implemented a data structure called a Circular Buffer.

A Circular Buffer (also known as a Ring Buffer) is a fixed-size data structure that treats the storage as if it were connected end-to-end in a circle. When the end of the buffer is reached, new elements wrap around to the beginning.

Implemented a circular buffer in Python using object-oriented programming (OOP). Designed a custom class with methods for push, pop, peek, and dynamic resizing.

Functionality Explanation:

- The head points to where the next item will be inserted (youngest).
- The tail points to the oldest item in the buffer.

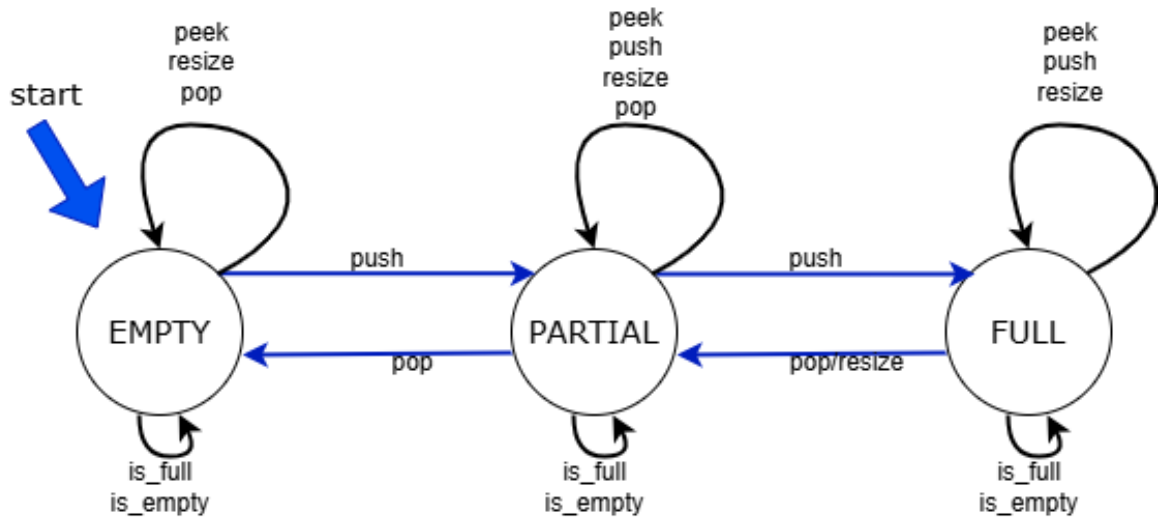
If the circular buffer reaches its maximum capacity and the user continue pushing new values, the oldest value (tail) will be overwritten by the new value.

The head and tail pointers update accordingly to maintain the circular structure.

When the buffer is resized (especially reduced in size), only the most recent **(**youngest**)** values are kept. The older values are discarded to fit the new size.

The __str__ method is used for debugging and testing. It provides a live status of the buffer, including the head, tail, and count values.

More details about the internal flow can be found in the FMS drawing module.



More details can be found at the readme notes