The LastK ADT

The following abstract data type will be used in the next homework.
Note: It doesn't use a mapping data structure.

A `LastK` data structure is a collection that only keeps the last `k` items added, where `k` is a parameter specified when initializing a new instance. It should support the following operations.

- `__init__(self, k)` - Initialize a new `LastK` data structure to store the last `k` items.
- `add(self, item)` - add `item` to the collection. If there are already `k` items in the collection, then the oldest one will be evicted.
- `__getitem__(self, i)` - returns the item with index `i` in the sorted list of the last `k` items added. For example, if we added 1,2,3,4,5 in order to a `LastK` object `C` and `k = 4`, then `C[0] = 2`, `C[1] = 3`, `C[2] = 4`, and `C[3] = 5`. If the index is negative or is greater than or equal to the number of items in the collection, then `__getitem__` should raise an `IndexError`.
- `first(self)` - returns the oldest item in the collection. Raise an `IndexError` if the collection is empty.
- `last(self)` - returns the newest item in the collection. Raise an `IndexError` if the collection is empty.
- `clear` - resets back to an empty collection.
- `__len__` - return the number of items currently stored. This should be a number from 0 to k.

What to do

Implement a class called `LastK` that implements the LastK ADT. Put it in a file called `lastk.py`. The goal is to implement `add` and `__getitem__` in $O(1)$ time and $O(k)$ space. The best way to implement this is with a so-called circular list. Just keep a single list of length `k`, keep track of which index is the start, and use modular arithmetic to "*wrap*" the indices back around to the beginning of the list. This way, when it is full, new entries overwrite the one that is no longer needed automatically.