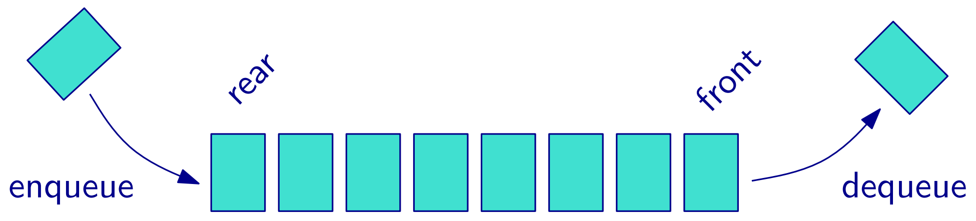
Queues (First In First Out: FIFO)

- A queue is a collection whose elements are added on one end and removed from the other.
- Therefore a queue is processed in a FIFO fashion: first in, first out.
- Elements are removed in the same order they arrive.
- Any waiting line is a queue:
  - the check out line at a grocery store,
  - the cars at a stop light,
  - an assembly line.
- Queue = FIFO, Stack = LIFO

- A queue is usually depicted horizontally.
- One end of the queue is the rear (or tail), where elements are added (enqueued).
- The other end is the front (or head), from which elements are removed (dequeued).
- Unlike a stack, which operates on one end of the collection, a queue operates on both ends.
- Like a stack, a pure queue does not allow the user to access the elements in the middle of the queue.



Queues in the computing environment

- Email is queued
- Graphical User Interfaces (GUIs) depend upon event queues.
- Documents sent to the printer are spooled (queued).
- Data transferred to a stream are buffered (queued).
- Machine instructions are executed using a sophisticated queue, known as a pipeline.

Computing capital gains

When selling shares, one must pay tax on the capital gains, the difference between the price for which the stock is sold and the price for which it was bought.

In reality, the situtation is a bit more complicated:

Date		Number of shares	Price KRW
Mar 15	buy	10	20000
Apr 2	buy	5	21000
Apr 20	buy	20	19000
May 15	sell	5	23000
June 3	sell	12	22000
July 15	buy	10	21000
Aug 15	sell	28	22000

Computing capital gains

The standard accounting principle for capital gains valuation is first-in-first-out (FIFO).

Algorithm:

- When buying shares, enqueue number of shares and cost.
- When selling n shares:
  - Look at oldest shares (front of the queue). Let their number be m.
  - If $n < m$, then compute profit for n shares based on price difference. Decrease number of oldest shares.
  - Otherwise, compute profit for m shares based on price difference. Dequeue oldest shares. Let $n \leftarrow n - m$, and repeat.

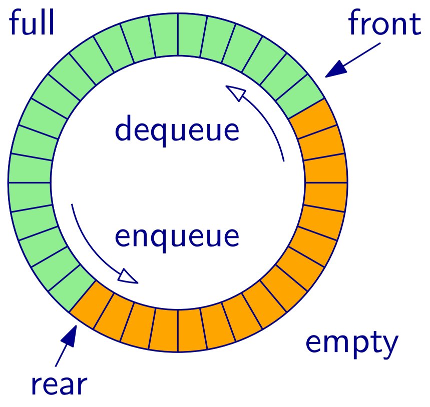
Implementing a Queue

- As a Python list:
  `is_empty` and `front` constant time. `enqueue` constant time on average.
  But `dequeue` takes time linear in the size of the queue!
- Can we make a "linked queue" with constant time per operation, similar to our `linkedstack`?
  The problem is that we need to modify the linked structure on both ends. We'll look at this later!
- Using two Python lists: All operations take constant time on average.
- Can we guarantee constant time per operation if the maximum queue size is known in advance?

Circular buffers

Implementing a stack using an array is easy—just keep an index to remember the top of the stack inside the array.

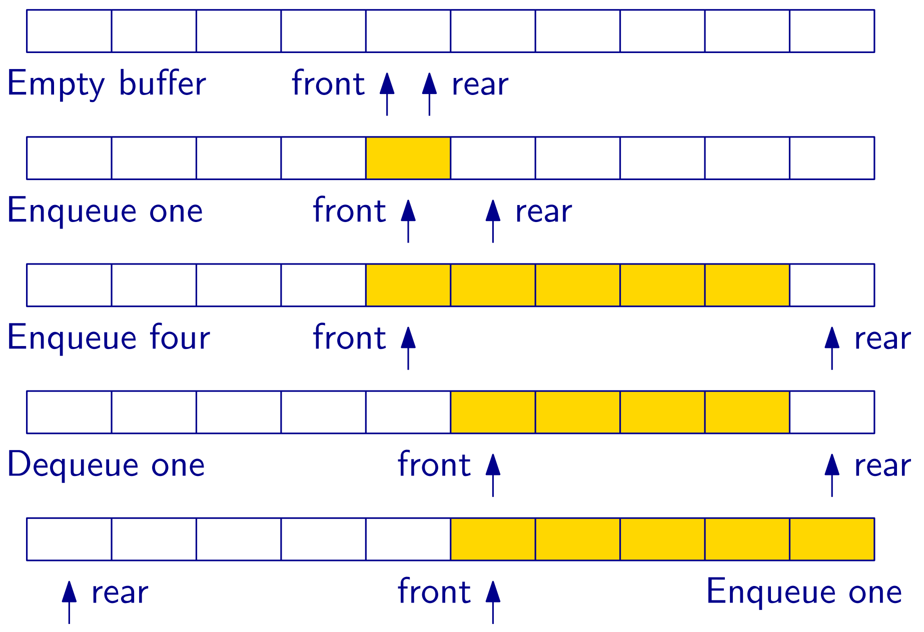
Implementing a queue is harder, because we insert and remove elements at two different places. The normal way to do this is using a circular buffer.



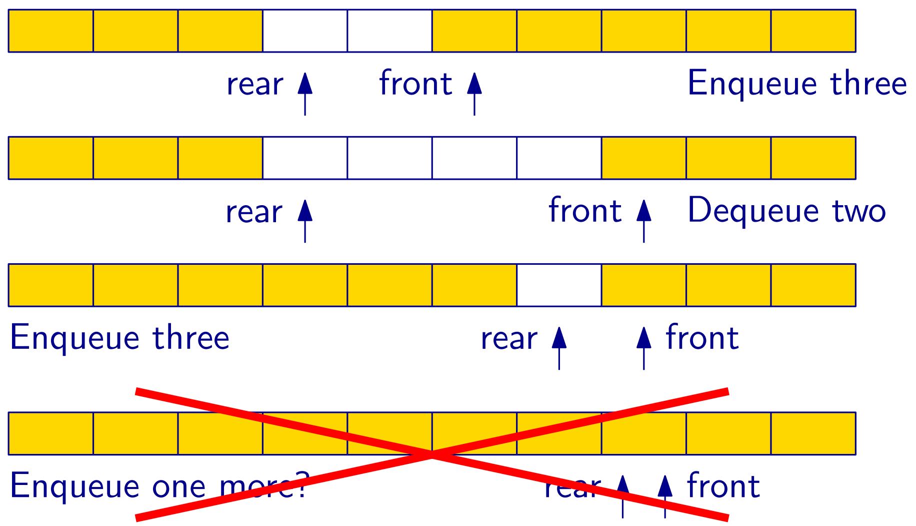
Since we do not have circular memory, we have to simulate that using a linear array, and two index pointers.

When the index reaches the end, it "wraps around" to the beginning.

Circular buffers



Circular buffers



What does `front == rear` mean? Full buffer or empty buffer?

Typical solution: Forbid filling buffer completely, always keep one slot free. (See Circular Buffers on Wikipedia for other solutions.)