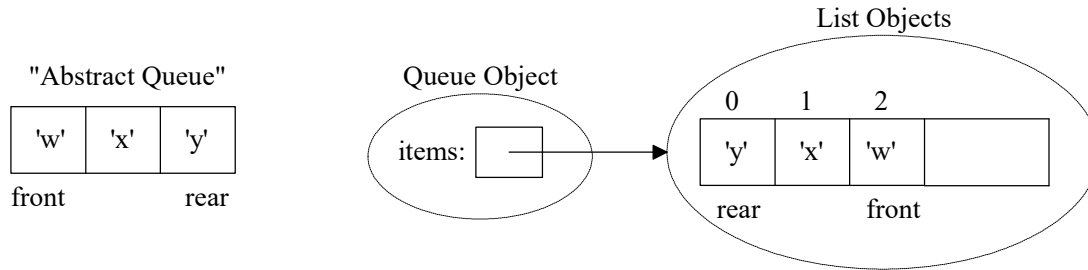


**Objective:** To understand FIFO (First-In-First-Out) queue implementations in Python including being able to determine the big-oh of each operation.

**To start the lab:** Download and unzip the file at: [www.cs.uni.edu/~fienup/cs1520s19/labs/lab3.zip](http://www.cs.uni.edu/~fienup/cs1520s19/labs/lab3.zip)

**Part A:** The textbook's QueueText implementation in lab3/queue\_text.py uses a Python list



a) Complete the big-oh notation for the above QueueText implementation: ("n" is the # items)

	<code>__init__</code>	<code>enqueue(item)</code>	<code>dequeue()</code>	<code>peek()</code>	<code>size()</code>	<code>isEmpty()</code>	<code>__str__</code>
Big-oh							

b) Explain your big-oh answer for enqueue(item).

c) Explain your big-oh answer for dequeue()

d) Run the timeQueue.py file which times 100,000 enqueues followed by 100,000 dequeues.

Time for 100,000 enqueues:

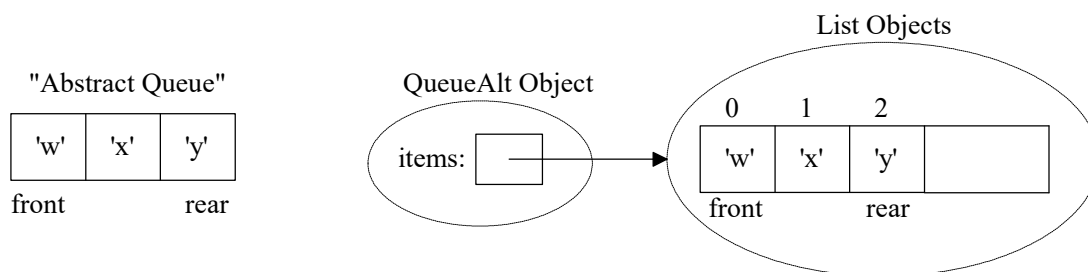
Time for 100,000 dequeues:

e) Why do the enqueues take so much more time?

**After answering the above questions, raise you hand and explain your answers.**

**Part B:**

a) Complete the QueueAlt implementation in lab3/queue\_alt.py uses a Python list



b) Complete the big-oh notation for the above QueueAlt implementation: ("n" is the # items)

	<code>__init__</code>	<code>enqueue(item)</code>	<code>dequeue()</code>	<code>peek()</code>	<code>size()</code>	<code>isEmpty()</code>	<code>__str__</code>
Big-oh							

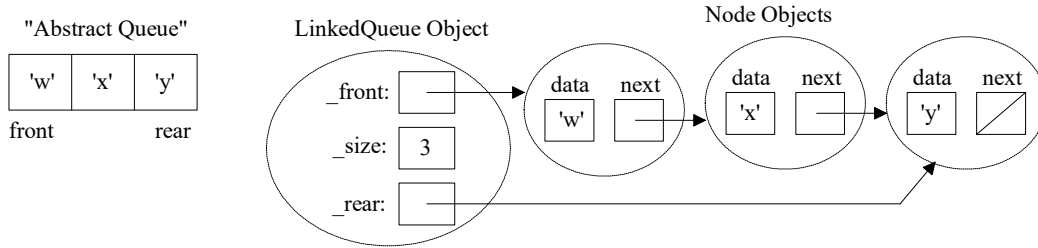
c) Run the timeQueueAlt.py file which times 100,000 enqueues followed by 100,000 dequeues.

Time for 100,000 enqueues:

Time for 100,000 dequeues:

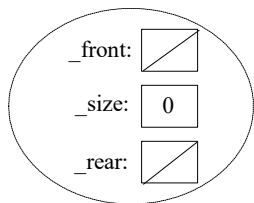
**After completing the QueueAlt class, answering the above questions, raise you hand and demonstrate your code.**

**Part C:** Consider the `LinkedListQueue` implementation in `lab3/linked_queue.py` which uses a linked structure that looks like:

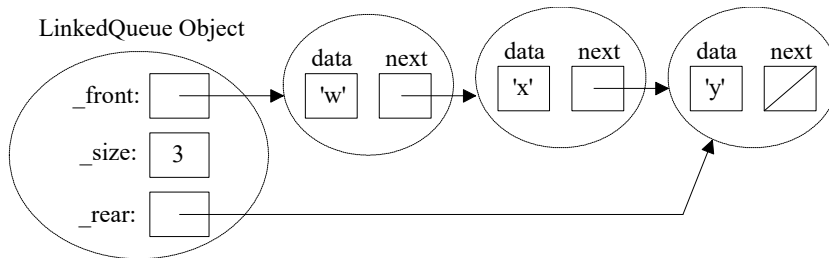


- a) Modify the above picture and number the steps for the `enqueue` method's "normal" case (non-empty queue)
- b) Complete the `enqueue` method code for the "normal" case in the `lab3/linked_queue.py` file
- c) Starting with the empty queue below, draw the resulting picture after your "normal" case code executes.

empty `LinkedListQueue` Object



- d) Fix your "normal" case code to handle the "special case" of an empty queue.



- e) Modify the above picture and number the steps for the `dequeue` method's "normal" case (non-empty queue)
- f) Complete the `dequeue` method code for the "normal" case in the `lab3/linked_queue.py` file
- g) What "special case(s)" does the `dequeue` method code need to handle?

h) Complete the big-oh notation for the `LinkedListQueue` methods: ("n" is the # items)

	<code>__init__</code>	<code>enqueue(item)</code>	<code>dequeue()</code>	<code>peek()</code>	<code>size()</code>	<code>isEmpty()</code>	<code>__str__</code>
Big-oh							

- i) Run the `timeLinkedListQueue.py` file which times 100,000 enqueues followed by 100,000 dequeues.  
 Time for 100,000 enqueues: \_\_\_\_\_ Time for 100,000 dequeues: \_\_\_\_\_

**After thoroughly testing your linked implementation, raise you hand and demonstrate your queue.**