CSCA48 Winter 2018
Week 3: Priority Queue, Linked Lists

Marzieh Ahmadzadeh, Nick Cheng

University of Toronto Scarborough

Implementing Python's List

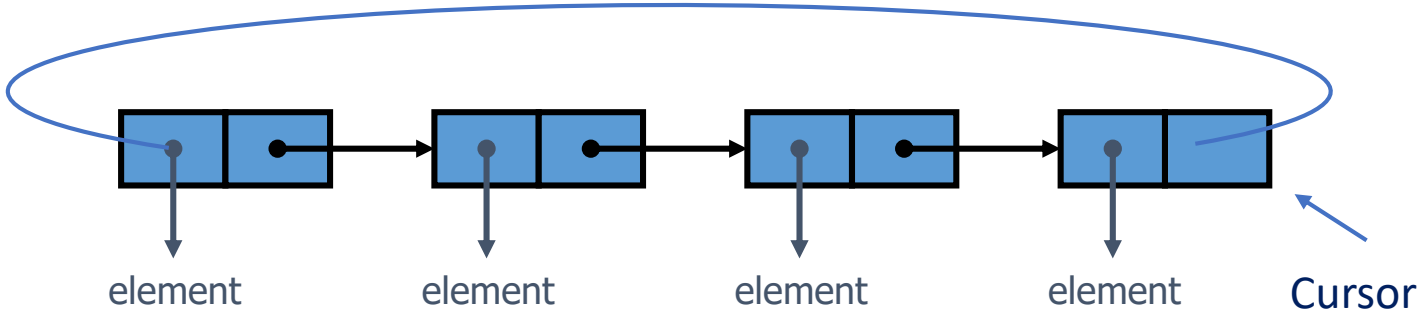
- You've already seen `append()` [i.e. add_first()] and `insert(0, element)` [i.e. add_last()] and `pop(0)` [i.e. remove_first()]

- `clear()`: remove all items from the list
- `copy()`: creates a shallow copy of the list
- `count(item)`: returns the number of occurrence of a the item.
- `index(item)`: returns the index at which the given item first seen. If item is not in the list, raise ValueException
- `insert(index, obj)`: insert the obj at the given index
- `pop(index)`: remove the item at the given index. Raises IndexError if list is empty or index is out of range

Circular Linked List

- An ADT like a queue
- Each item that is removed might be inserted to the back of the queue again.

- Application:
  - Round-Robin CPU scheduler



- For each entry you may need to
  - p=dequeue()
  - Provide service
  - enqueue(p)
- The only data structure that is available is single linked list
- It is a better idea to move the pointer forward instead of repeatedly run dequeue() and enqueue().
- This pointer points to the tail of the queue.

Circular linked lists

- Needs only one pointer, `cursor`, to point to the node that will be processed next.
- Move the pointer forward, when a node receives a service.

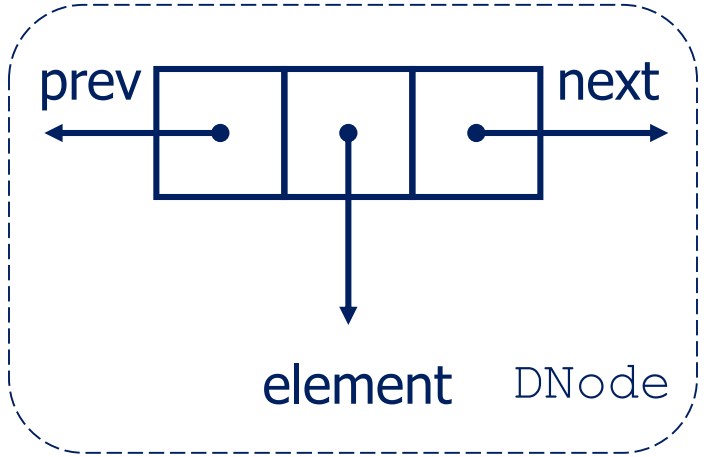
- Methods:
  - add(e): to add a node immediately after the cursor
  - Remove(): remove the node immediately after the cursor
  - advance(): advance the cursor to the next node in the list.

Questions

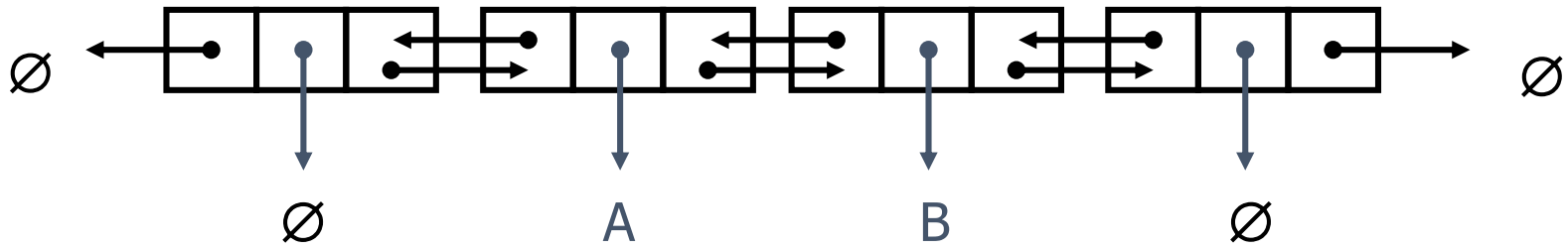
- In CLLs, how do you remove a node, where `cursor` points to?
- In SLL, how do you remove a node at the tail of the list?
- Answer: it's not efficient to remove this node (in both CLL and SLL) as there is no access to previous node.

Double Linked Lists

- A double linked list is a concrete data structure, whose building block is a `DNode`.
- Each node is an object that stores
  - a reference to an element
  - a reference called `next` that points to its next node.
  - a reference called `previous` that points to its previous node



- The first node and last nodes are called `head` and `tail`
- For ease of implementation, head and tail are a dummy node.



DLLs Methods

- `add_first(e)`: add an element to the front of the list.
- `add_last(e)`: add an element to back of the list
- `remove_first()`: remove a node from the front of the list
- `remove_last()`: removes a node from the back of the list

- You can implement a list using a DLL.
- Is it a good idea to implement a Stack using a DLL?