CSCI 204: Data Structures & Algorithms

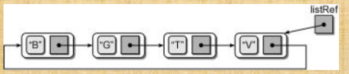
Advanced Linked lists

Circular Lists

Revised based on textbook author's notes.

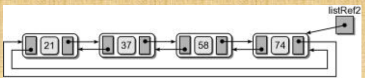
Circular Linked List

- Another variation of the linked list in which the nodes form a continuous circle.
  - Allows for a complete traversal from any initial node.
  - Used with round-robin type applications.
  - The external reference can point to any node in the list. Common to reference "end" of the list.



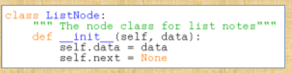
Circular Linked List

- A circular linked list can also be doubly linked.

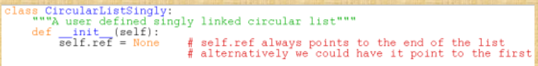


- For now, we will concentrate on singly linked circular lists

ListNode and List

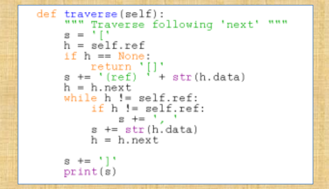


```
class ListNode:
    """ The node class for list notes"""
    def __init__(self, data):
        self.data = data
        self.next = None
```

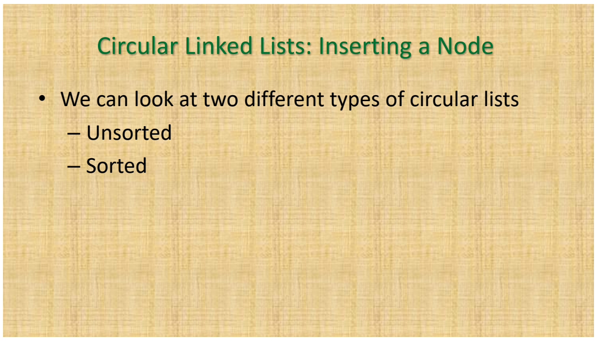


```
class CircularListSingly:
    """A user defined singly linked circular list"""
    def __init__(self):
        self.ref = None     # self.ref always points to the end of the list
                            # alternatively we could have it point to the first
```

Circular Linked List: Traverse

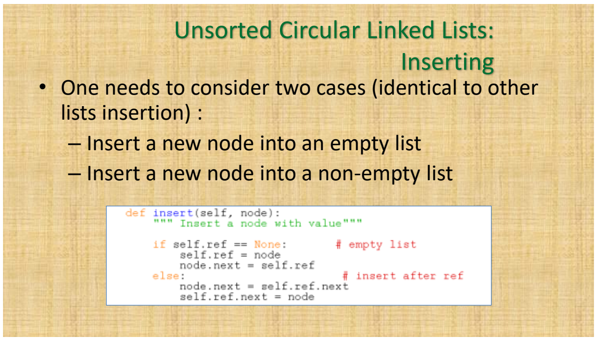


```
    def traverse(self):
        """ Traverse following 'next' """
        s = '['
        h = self.ref
        if h == None:
            return '[]'
        s += '(ref) ' + str(h.data)
        h = h.next
        while h != self.ref:
            if h != self.ref:
                s += ', '
            s += str(h.data)
            h = h.next

        s += ']'
        print(s)
```



Circular Linked Lists: Inserting a Node

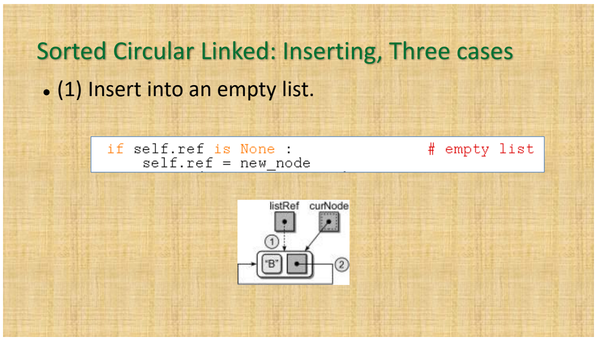
- We can look at two different types of circular lists
  - Unsorted
  - Sorted



Unsorted Circular Linked Lists: Inserting

- One needs to consider two cases (identical to other lists insertion) :
  - Insert a new node into an empty list
  - Insert a new node into a non-empty list

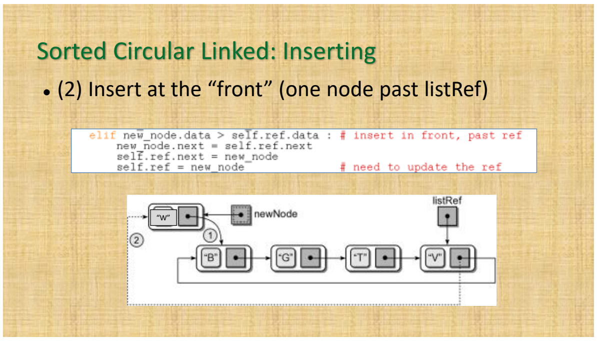
```
    def insert(self, node):
        """ Insert a node with value"""

        if self.ref == None:        # empty list
            self.ref = node
            node.next = self.ref
        else:                        # insert after ref
            node.next = self.ref.next
            self.ref.next = node
```



Sorted Circular Linked: Inserting, Three cases

- (1) Insert into an empty list.

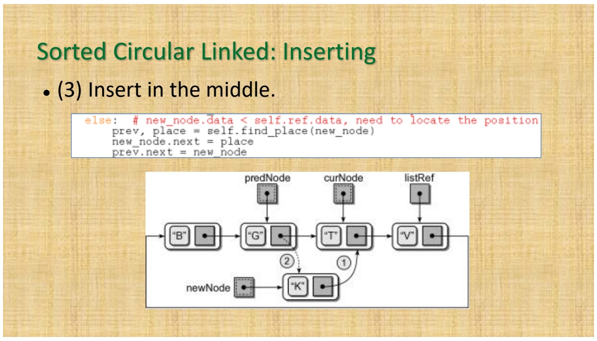
```
  if self.ref is None :                         # empty list
     self.ref = new_node
```



Sorted Circular Linked: Inserting

- (2) Insert at the “front” (one node past listRef)

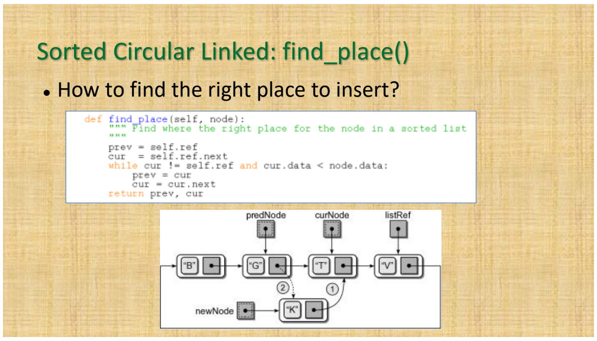
```
    elif new_node.data > self.ref.data : # insert in front, past ref
        new_node.next = self.ref.next
        self.ref.next = new_node
        self.ref = new_node                    # need to update the ref
```



Sorted Circular Linked: Inserting

- (3) Insert in the middle.

```
  else:  # new_node.data < self.ref.data, need to locate the position
      prev, place = self.find_place(new_node)
      new_node.next = place
      prev.next = new_node
```

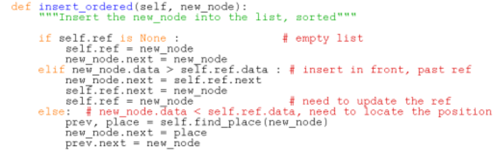


Sorted Circular Linked: find_place()

- How to find the right place to insert?

```
    def find_place(self, node):
        """ Find where the right place for the node in a sorted list
        """
        prev = self.ref
        cur  = self.ref.next
        while cur != self.ref and cur.data < node.data:
            prev = cur
            cur = cur.next
        return prev, cur
```

Circular Singly Linked: Inserting



```
def insert_ordered(self, new_node):
    """Insert the new_node into the list, sorted"""

    if self.ref is None :                   # empty list
        self.ref = new_node
        new_node.next = new_node
    elif new_node.data > self.ref.data : # insert in front, past ref
        new_node.next = self.ref.next
        self.ref.next = new_node
        self.ref = new_node                 # need to update the ref
    else:  # new_node.data < self.ref.data, need to locate the position
        prev, place = self.find_place(new_node)
        new_node.next = place
        prev.next = new_node
```

Try out `test_singly_circular_list.py`

How to remove a node?

- Your exercise