```
# Singly Linked Lists

class SinglyNode:

  def __init__(self, val, next=None):
    self.val = val
    self.next = next

  def __str__(self):
    return str(self.val)


Head = SinglyNode(1)
A = SinglyNode(3)
B = SinglyNode(4)
C = SinglyNode(7)

Head.next = A
A.next = B
B.next = C

print(Head)
```

```
1
```

```
# Traverse the list - O(n)
curr = Head

while curr:
  print(curr)
  curr = curr.next
```

```
1
3
4
7
```

```
# Diplay linked list - O(n)
def display(head):
  curr = head
  elements = []
  while curr:
    elements.append(str(curr.val))
    curr = curr.next
  print(' -> '.join(elements))

display(Head)
```

```
1 -> 3 -> 4 -> 7
```

```
# Search for node value - O(n)
def search(head, val):
```

```
# Search for node value - O(n)
def search(head, val):
  curr = head
  while curr:
    if val == curr.val:
      return True
    curr = curr.next

  return False

search(Head, 7)
```

```
True
```

```
# Doubly Linked Lists

class DoublyNode:
  def __init__(self, val, next=None, prev=None):
    self.val = val
    self.next = next
    self.prev = prev

  def __str__(self):
    return str(self.val)


head = tail = DoublyNode(1)
print(tail)
```

```
1
```

```
# Display - O(n)
def display(head):
  curr = head
  elements = []
  while curr:
    elements.append(str(curr.val))
    curr = curr.next
  print(' <-> '.join(elements))

display(head)
```

```
1
```

```
# Insert at beginning - O(1)
def insert_at_beginning(head, tail, val):
  new_node = DoublyNode(val, next=head)
  head.prev = new_node
  return new_node, tail

head, tail = insert_at_beginning(head, tail, 3)
display(head)
```

```
    curr = curr.next
  print(' <-> '.join(elements))

display(head)
```

```
1
```

```
# Insert at beginning - O(1)
def insert_at_beginning(head, tail, val):
  new_node = DoublyNode(val, next=head)
  head.prev = new_node
  return new_node, tail

head, tail = insert_at_beginning(head, tail, 3)
display(head)
```

```
3 <-> 1
```

```
# Insert at end - O(1)
def insert_at_end(head, tail, val):
    new_node = DoublyNode(val, prev=tail)
    tail.next = new_node
    return head, new_node

head, tail = insert_at_end(head, tail, 7)
display(head)
```

```
3 <-> 1 <-> 7
```

Start coding or generate with AI.