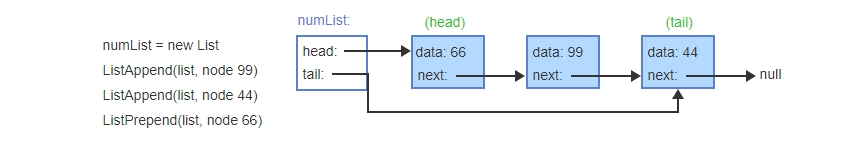
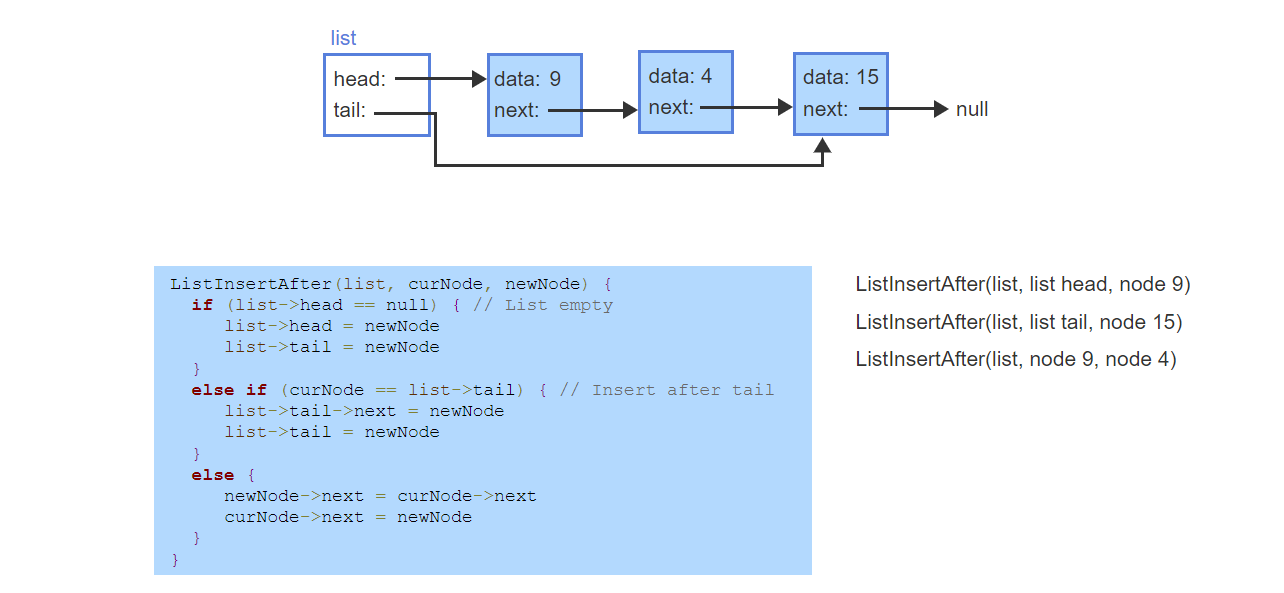
# 12.1 Singly-linked lists

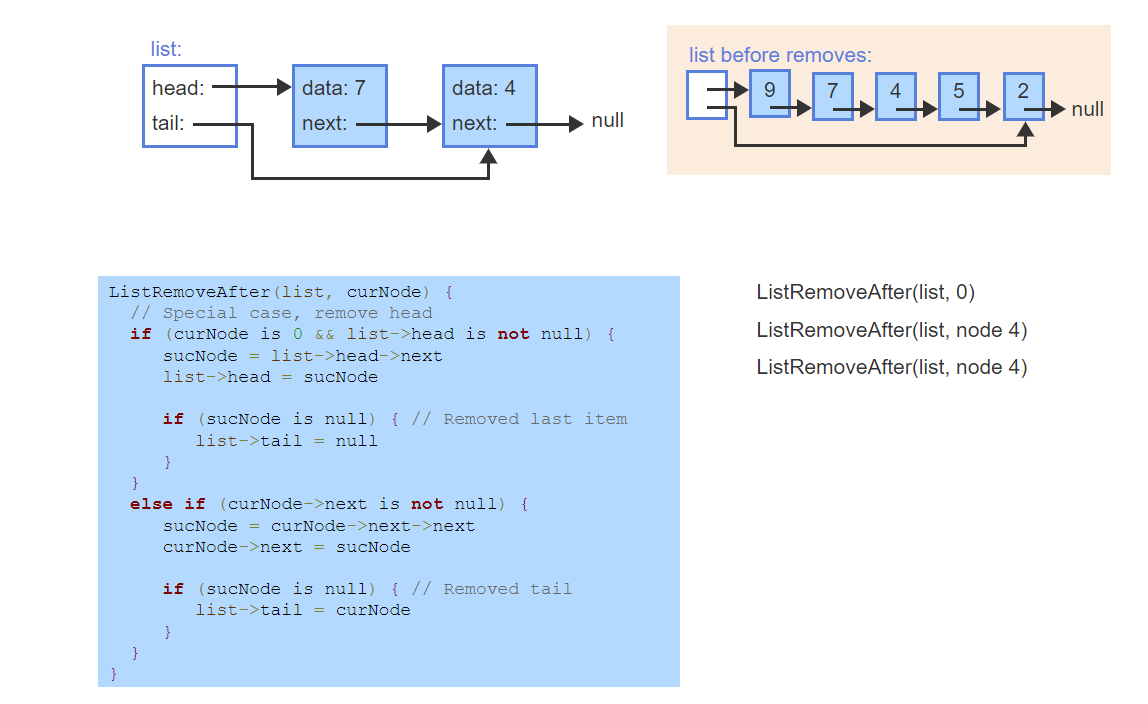


Prepending & Appending.

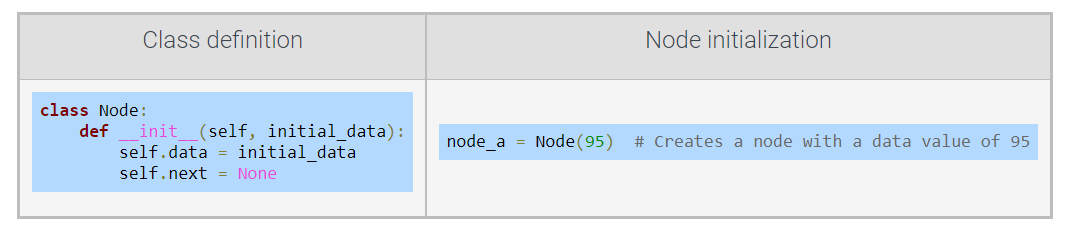
# 12.2 Singly-linked lists: Insert

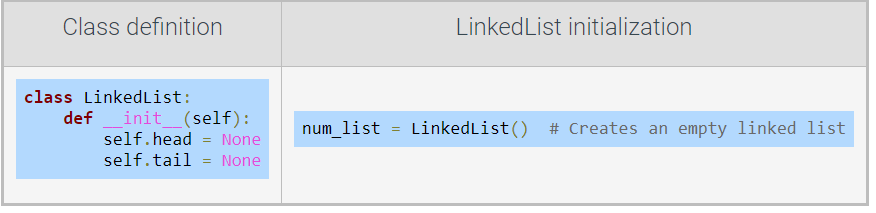


# 12.3 Singly-linked lists: Remove

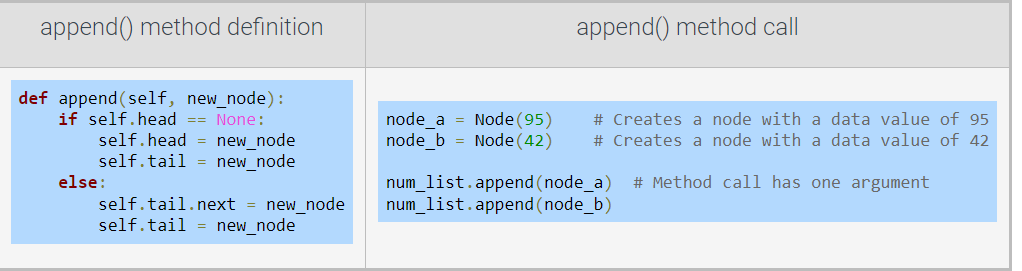


# 12.4 Python: Singly-linked lists

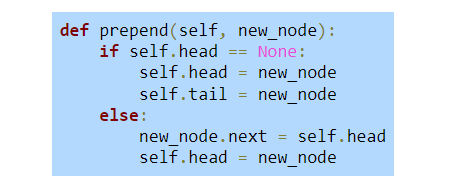


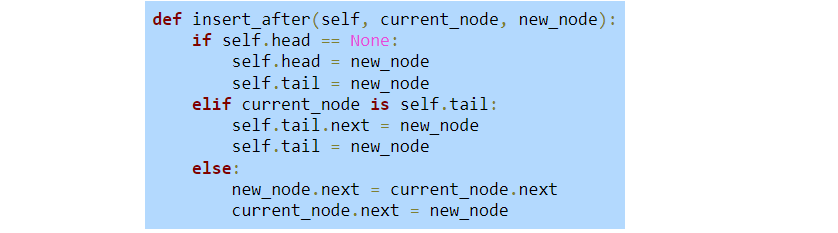


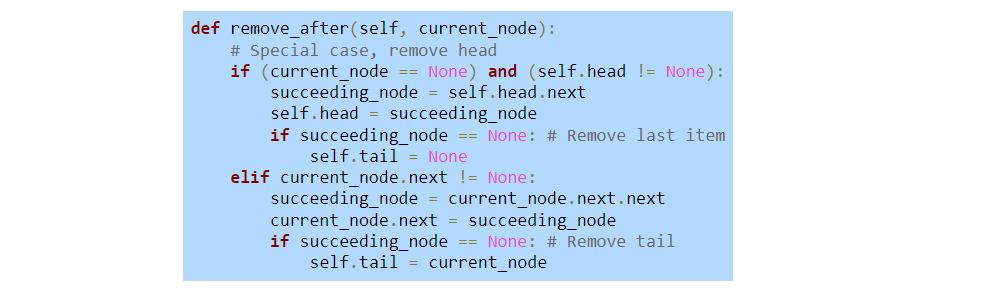
### Appending a node to a singly-linked list



### Additional singly-linked list methods

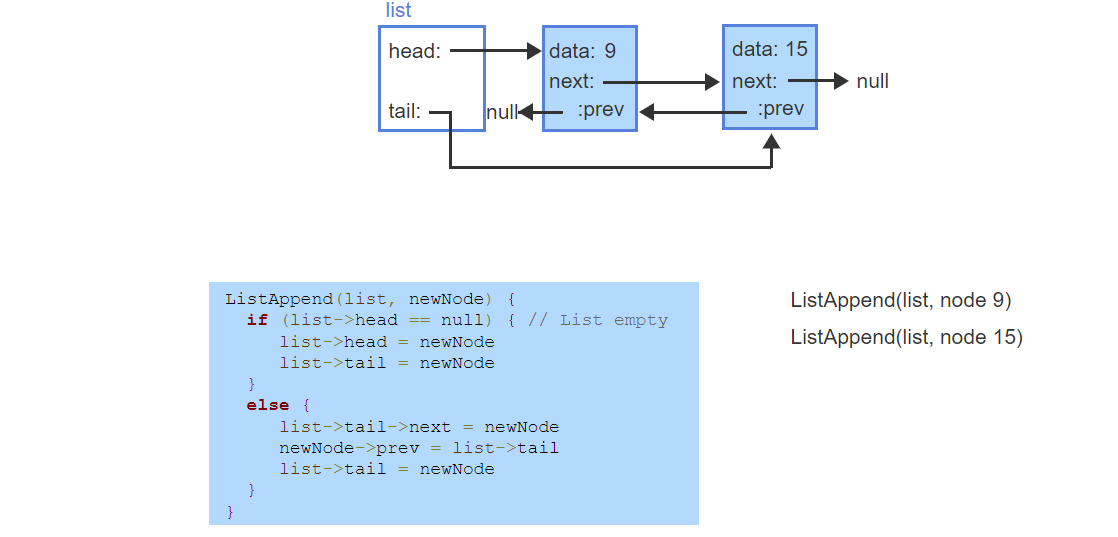


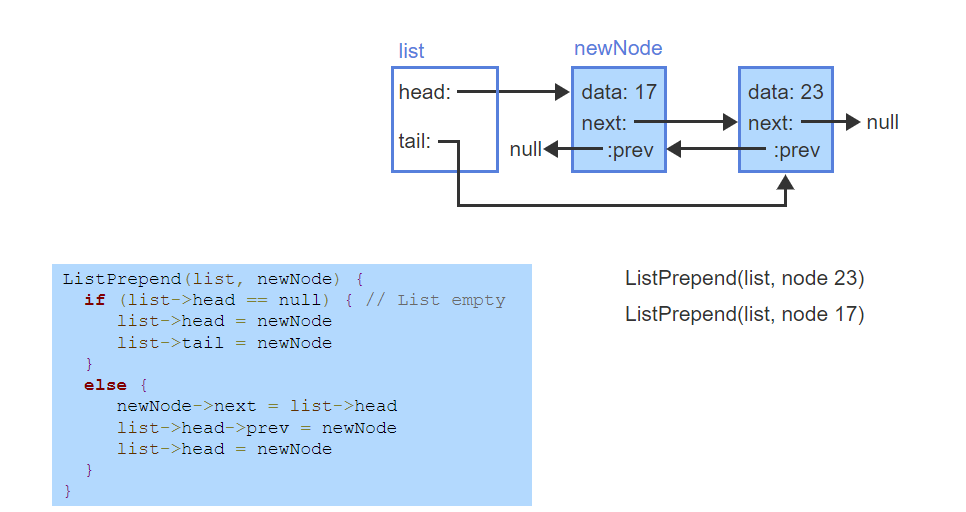




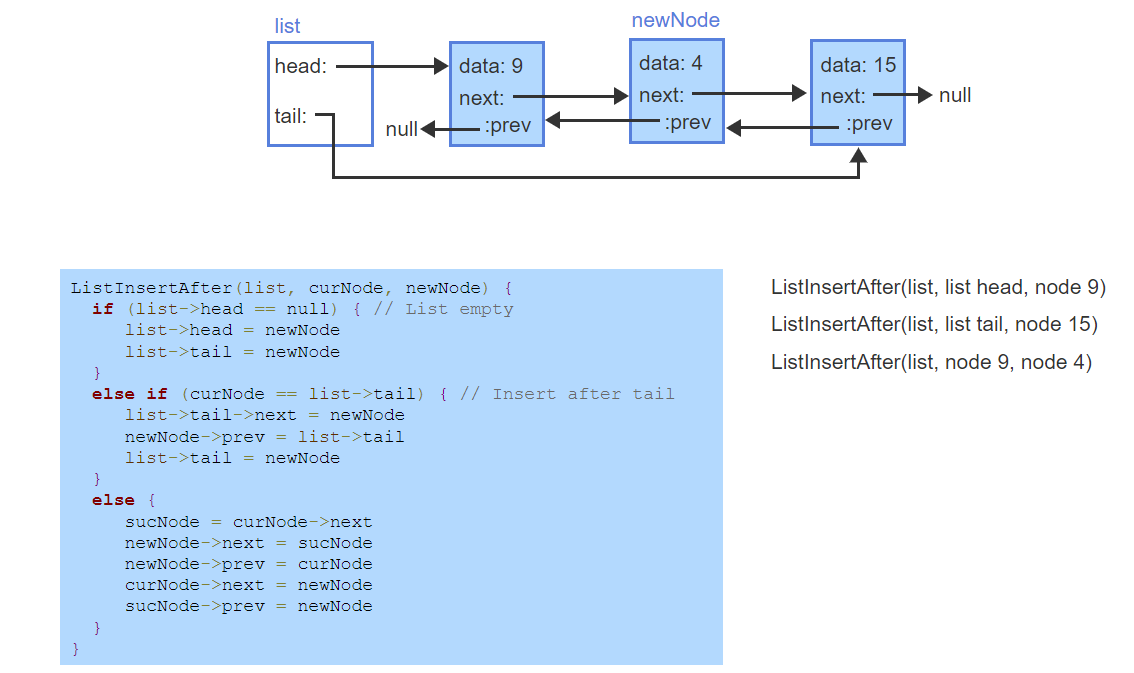
# 12.5 Doubly-linked lists

A ***doubly-linked list*** is a data structure for implementing a list ADT, where each node has data, a pointer to the next node, and a pointer to the previous node.

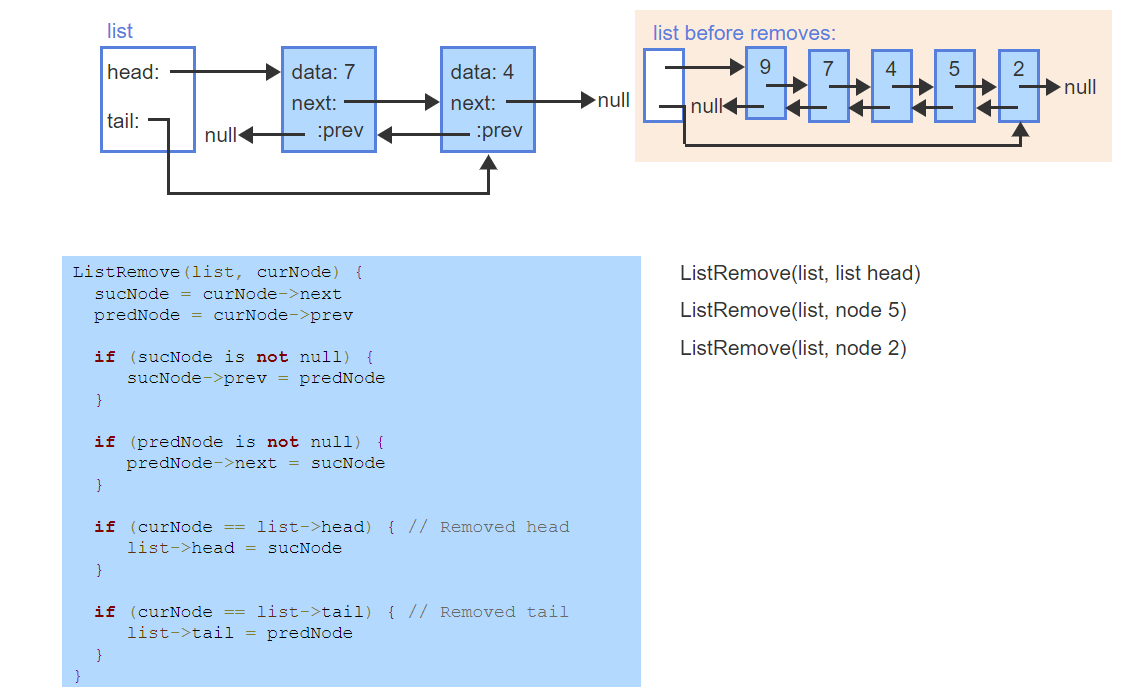
Append

prepend

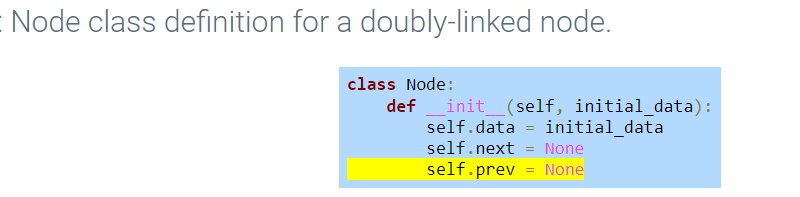
# 12.6 Doubly-linked lists: Insert

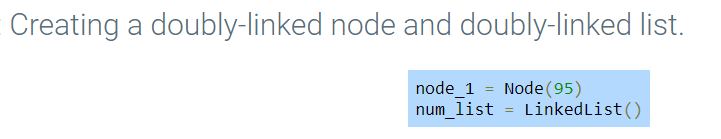


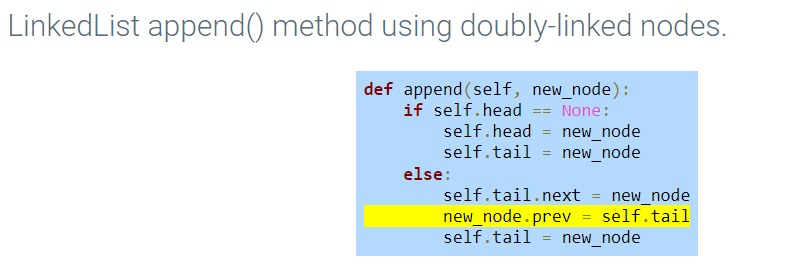
# 12.7 Doubly-linked lists: Remove



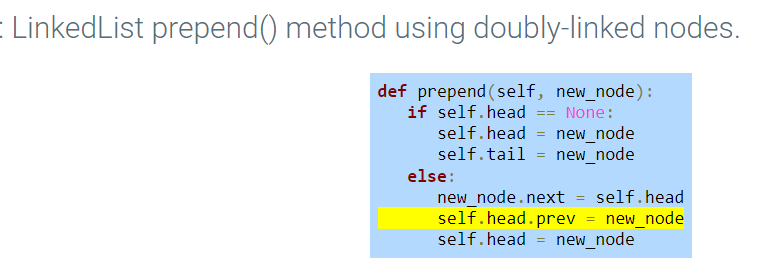
# 12.8 Python: Doubly-linked lists

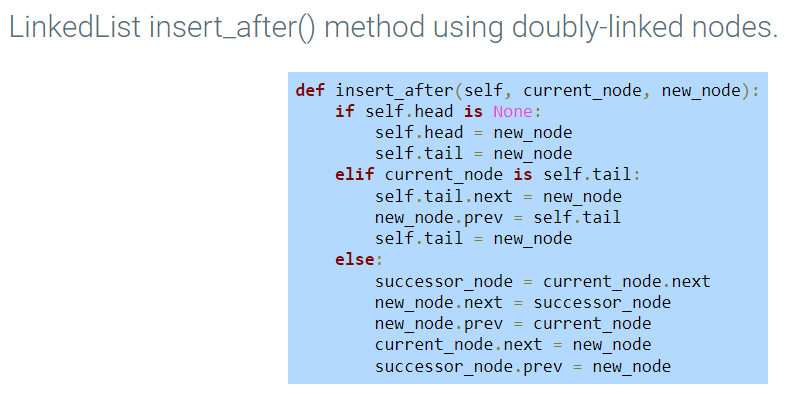


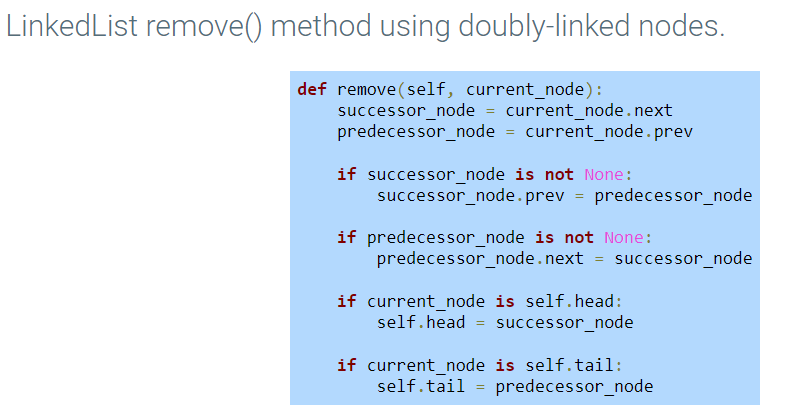




### Additional doubly-linked list methods



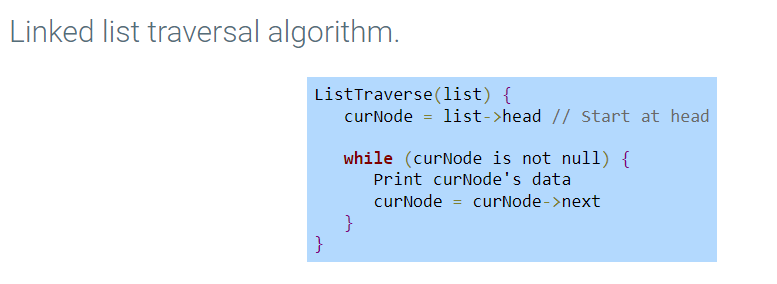




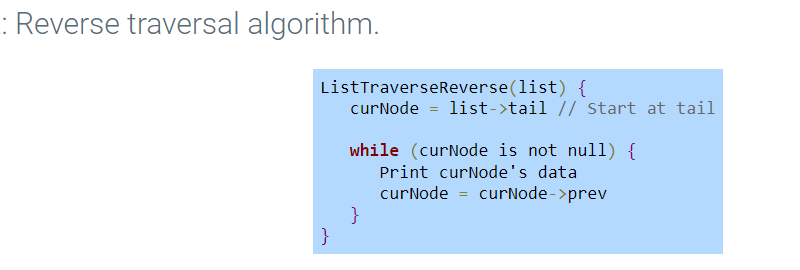
# 12.10 Linked list traversal

A ***list traversal*** algorithm visits all nodes in the list once and performs an operation on each node.

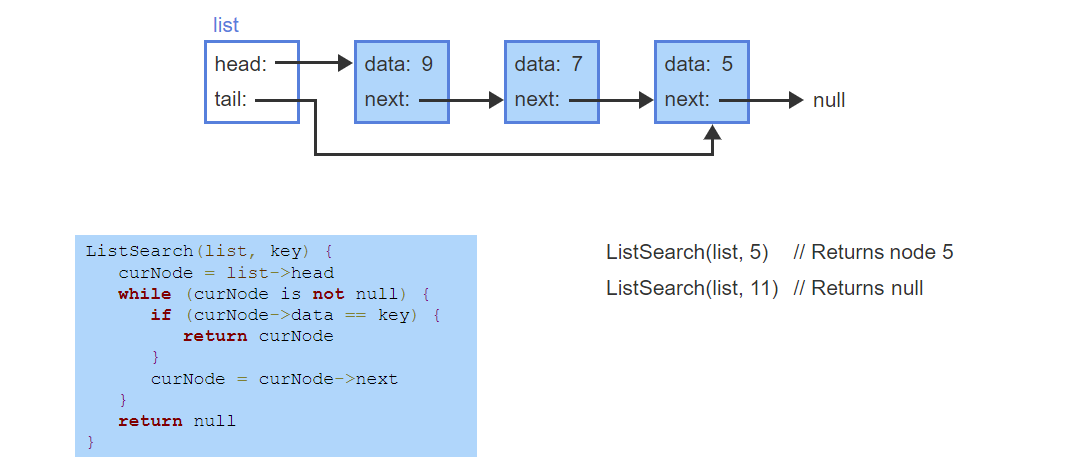
A common traversal operation prints all list nodes.



### Doubly-linked list reverse traversal

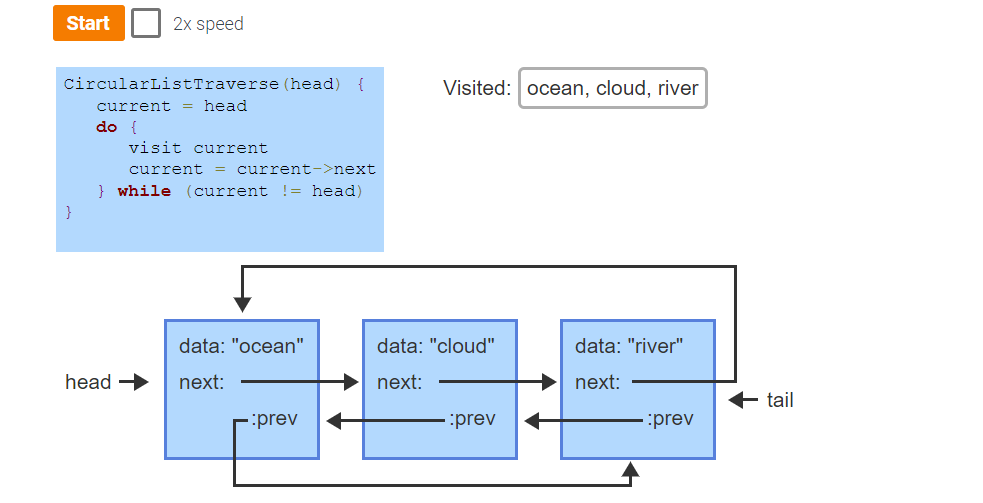


# 12.11 Linked list search



# 12.12 Circular lists

A ***circular linked list*** is a linked list where the tail node's next pointer points to the head of the list, instead of null.



# 12.13 Stack abstract data type (ADT)

A ***stack*** is an ADT in which items are only inserted on or removed from the top of a stack.

The stack ***push*** operation inserts an item on the top of the stack.

The stack ***pop*** operation removes and returns the item at the top of the stack.

A stack is referred to as a ***last-in first-out*** ADT. A stack can be implemented using a linked list, an array, or a vector.