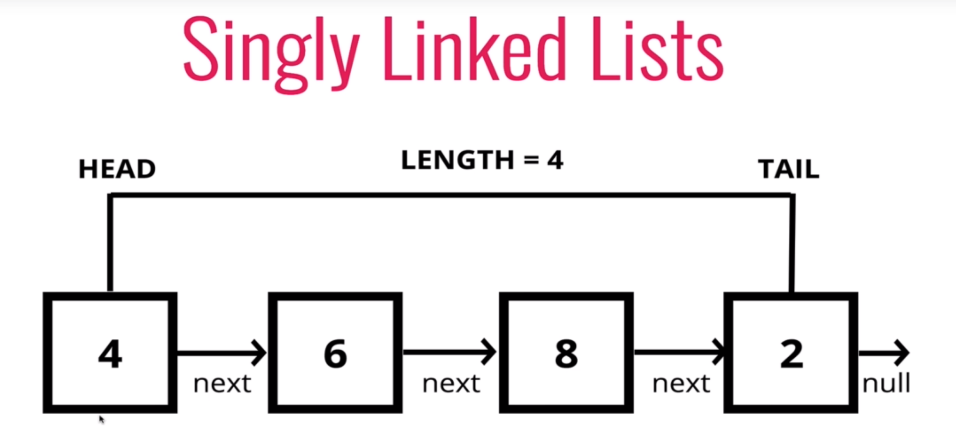
Singly-Linked List Implementation

1. **class** Node:
2. **def** \_\_init\_\_(self, data):
3. self.data = data
4. self.next = None
6. **class** LinkedList:
7. **def** \_\_init\_\_(self):
8. self.head = None
9. self.tail = None
10. self.length = 0
12. **def** append(self, data):
13. node = Node(data)
14. **if** **not** self.head:
15. self.head = node
16. self.tail = self.head
17. **else**:
18. self.tail.next = node
19. self.tail = node
20. self.length += 1
21. **return** self
23. **def** pop(self):
24. **if** **not** self.head:
25. **return** None
26. current = self.head
27. new\_tail = current
28. **while** current.next:
29. new\_tail = current
30. current = current.next
31. self.tail = new\_tail
32. self.tail.next = None
33. self.length -= 1
34. **if** self.length == 0:
35. self.head = None
36. self.tail = None
37. **return** current
39. **def** removeHead(self):
40. **if** **not** self.head:
41. **return** None
42. current = self.head
43. self.head = current.next
44. self.length -= 1
45. **if** self.length == 0:
46. self.tail = None
47. **return** current
49. **def** addHead(self, data):
50. node = Node(data)
51. **if** **not** self.head:
52. self.head = node
53. self.tail = self.head
54. node.next = self.head
55. self.head = node
56. self.length += 1
57. **return** self

60. **def** get(self, index):
61. **if** index < 0 **or** index >= self.length:
62. **return** None
63. counter = 0
64. current = self.head
65. **while** counter != index:
66. current = current.next
67. counter += 1
68. **return** current
70. **def** set(self, index, data):
71. found = self.get(index)
72. **if** found:
73. found.data = data
74. **return** True
75. **return** False
77. **def** insert(self, index, data):
78. **if** index < 0 **or** index > self.length:
79. **return** False
80. **if** index == self.length:
81. **return** self.append(data)
82. **if** index == 0:
83. **return** self.addHead(data)
84. node = Node(data)
85. prev = self.get(index - 1)
86. current = prev.next
87. prev.next = node
88. node.next = current
89. self.length += 1
90. **return** True
92. **def** remove(self, index):
93. **if** index < 0 **or** index > self.length:
94. **return** None
95. **if** index == 0:
96. **return** self.removeHead()
97. **if** index == self.length - 1:
98. **return** self.pop()
99. prev = self.get(index - 1)
100. current = prev.next
101. prev.next = current.next
102. self.length -= 1
103. **return** current
105. **def** reverse(self):
106. current = self.head
107. self.head = self.tail
108. self.tail = current
109. prev = None
110. **for** i **in** range(self.length):
111. next = current.next
112. current.next = prev
113. prev = current
114. current = next
115. **return** self

# Linked List Cheat Sheet



* Consists of a **head**, a **tail**, and has a **length** property
* Consists of **nodes**, and each node has a **data** **value** and a **pointer** to another node or None
* Analogy: skyscraper with only stairs to get from floor to floor (list has an elevator with indexes)

Linked Lists:

Python Lists:

* Do not have indexes
* Connected via nodes with a next pointer
* Random access is not allowed
* Indexed in order
* Insertion and deletion is expensive
* Can quickly be accessed at specific index

### Singly Linked Big O

* Insertion = O(1)
* Deletion = O(1)
* Searching = O(n)
* Access = O(n)