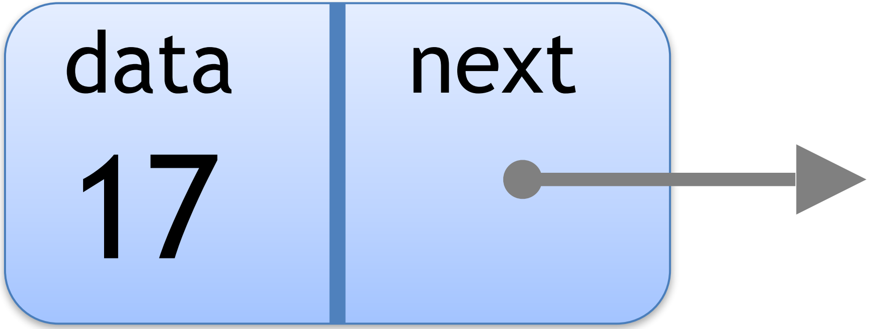
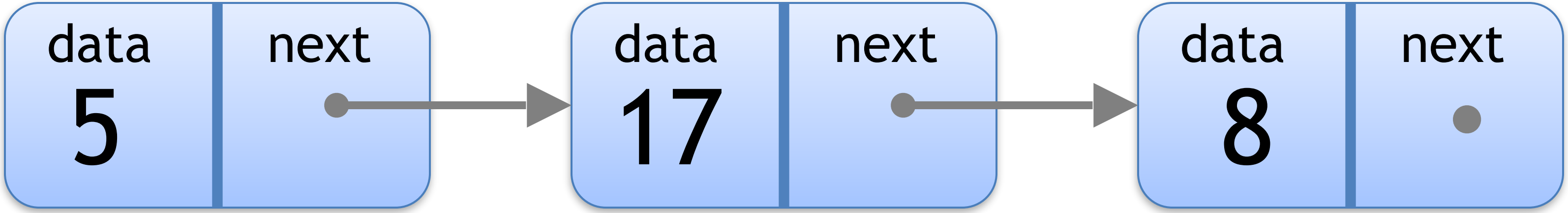
Python Linked Lists

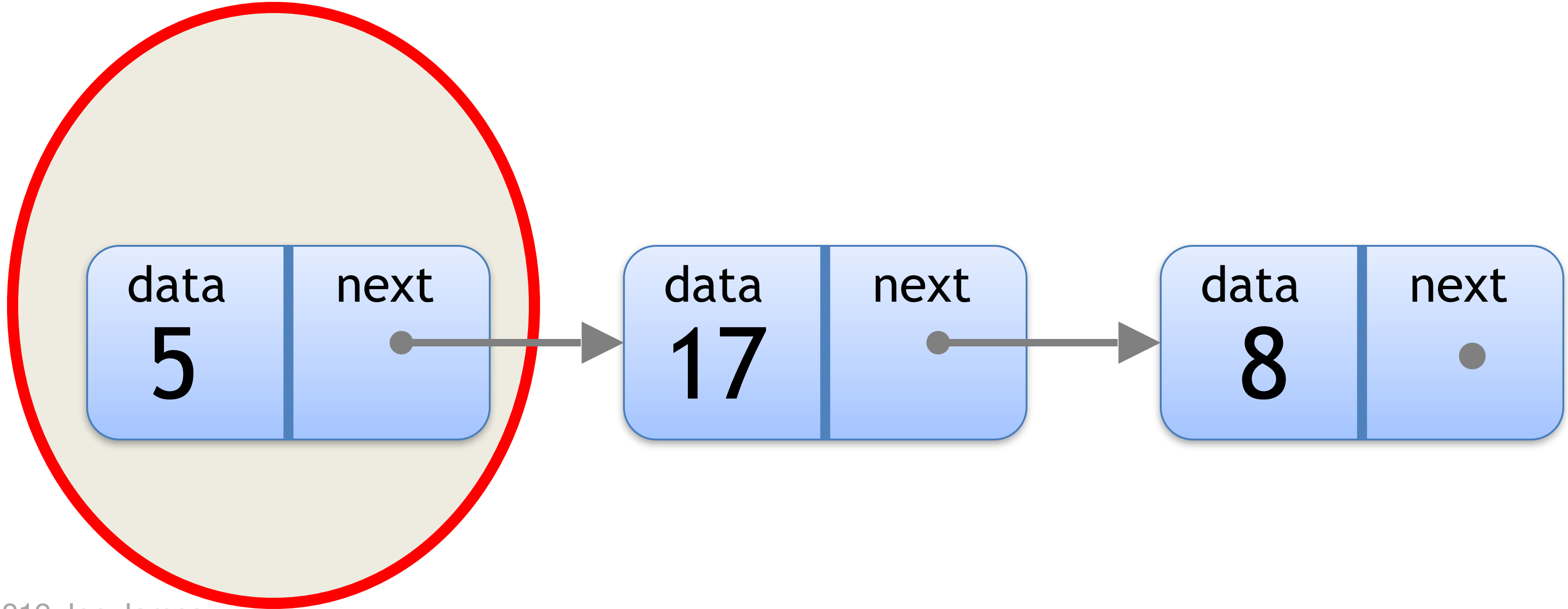
Every Node has 2 parts:

data and a pointer to the **next** Node




data
5
next
data
17
next
data
8
next

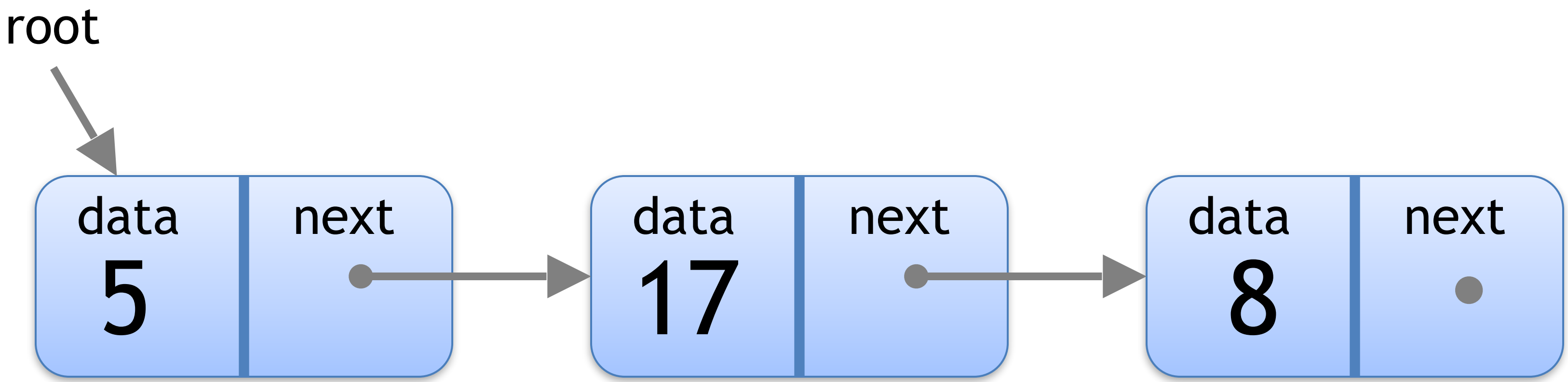
Root node



data 5 | next

data 17 | next

data 8 | next


root
data
5
next
data
17
next
data
8
next

Linked Lists

Attributes:

root - pointer to the beginning of the List

size - number of nodes in List

Operations:

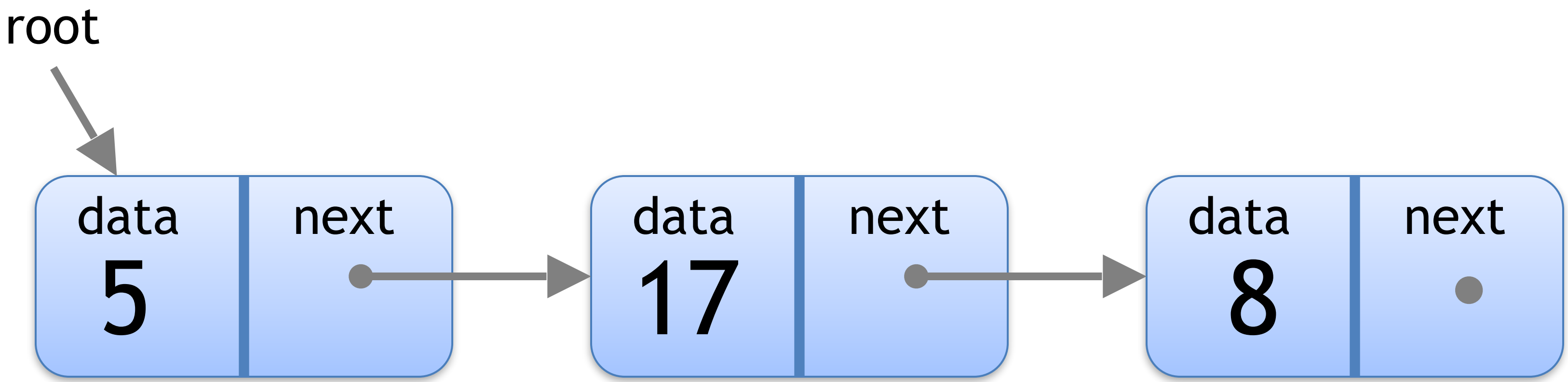
find(data)

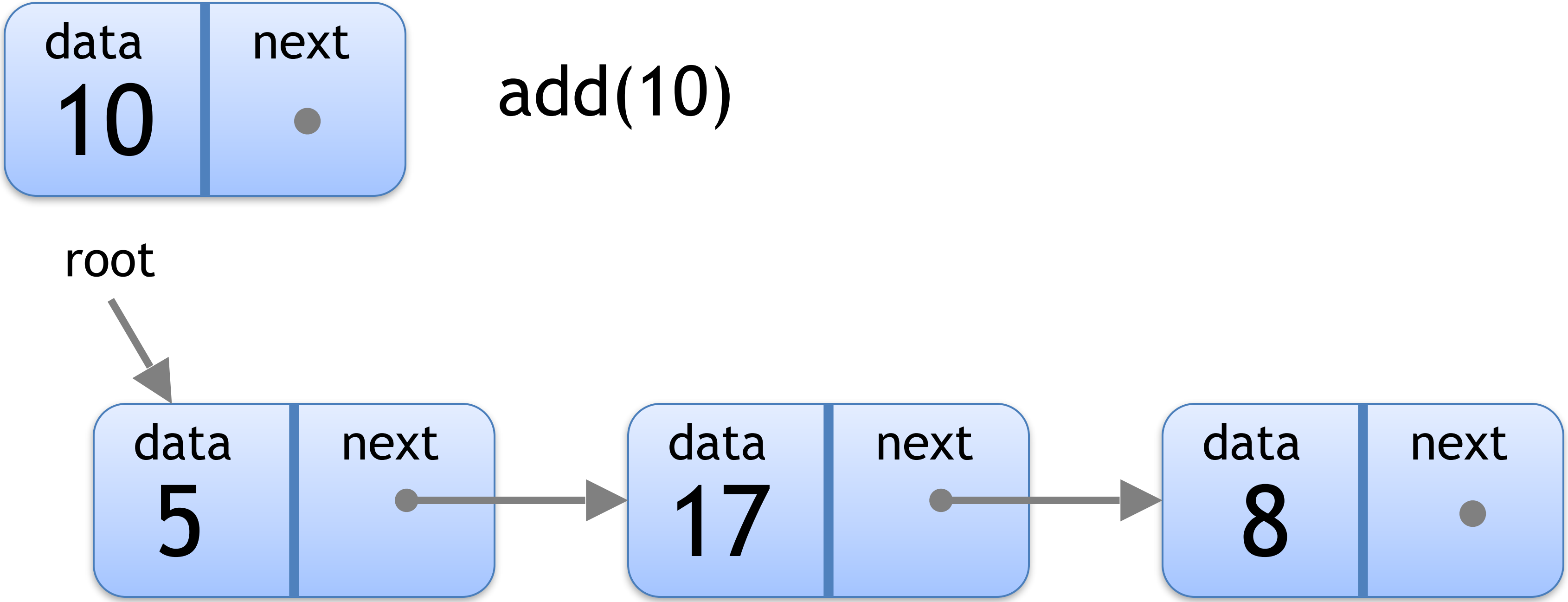
add(data)

remove(data)

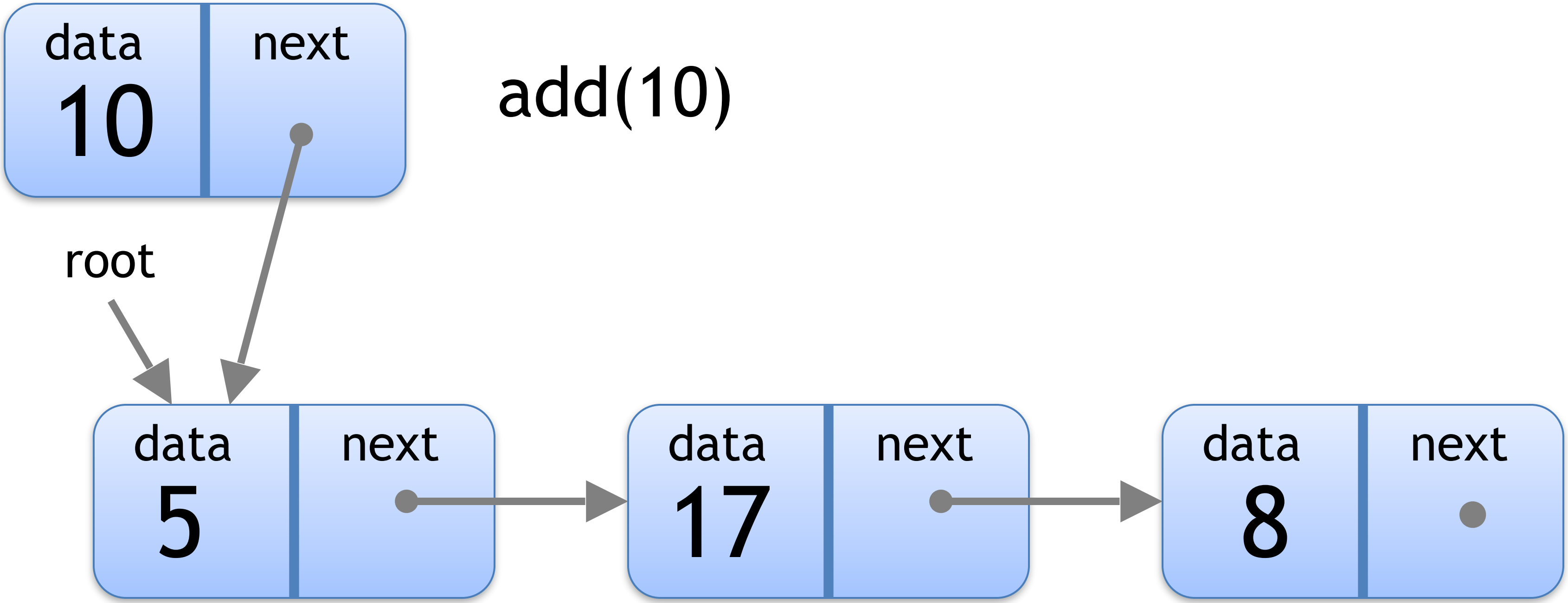
print_list()

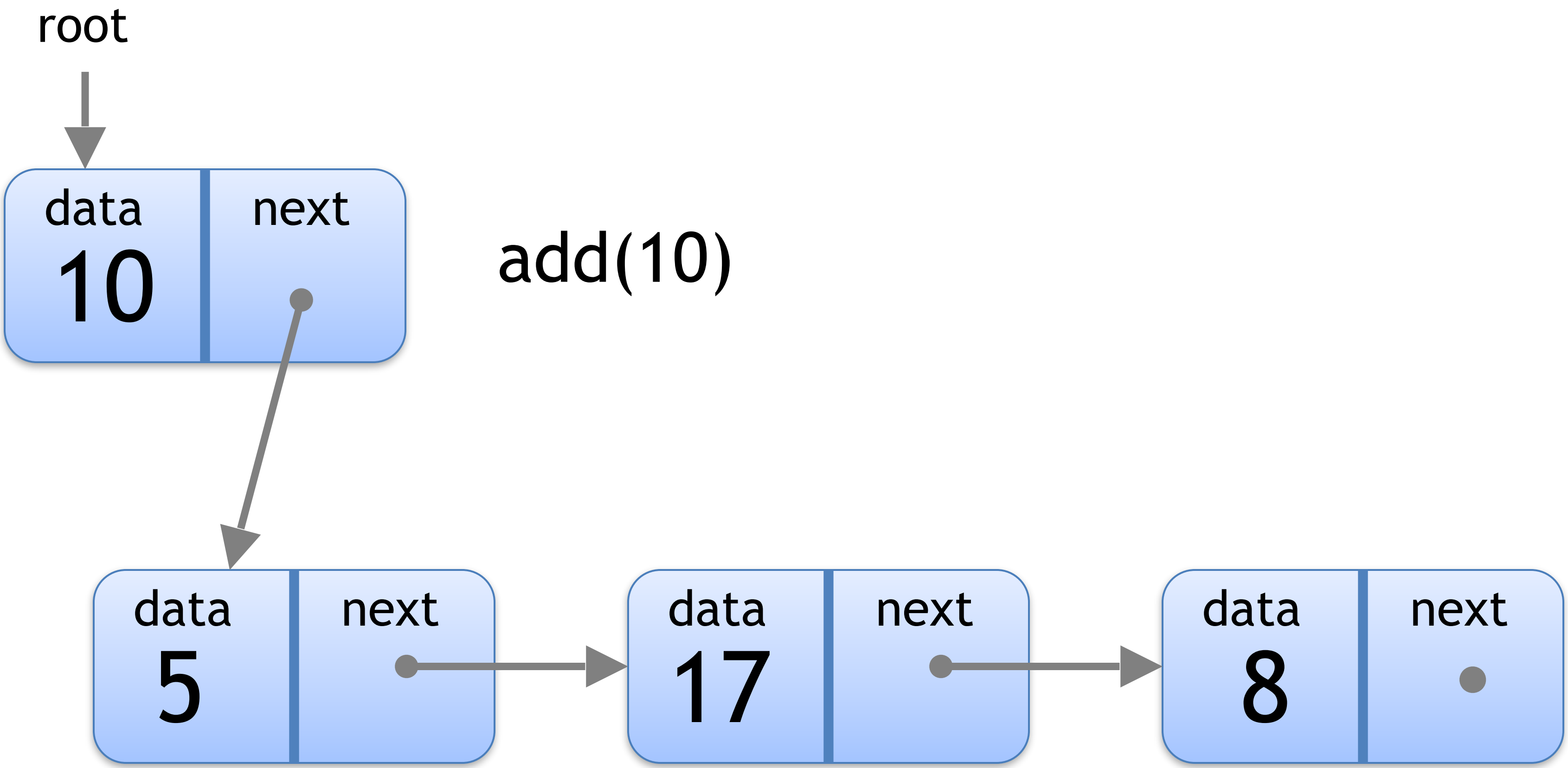
add(10)





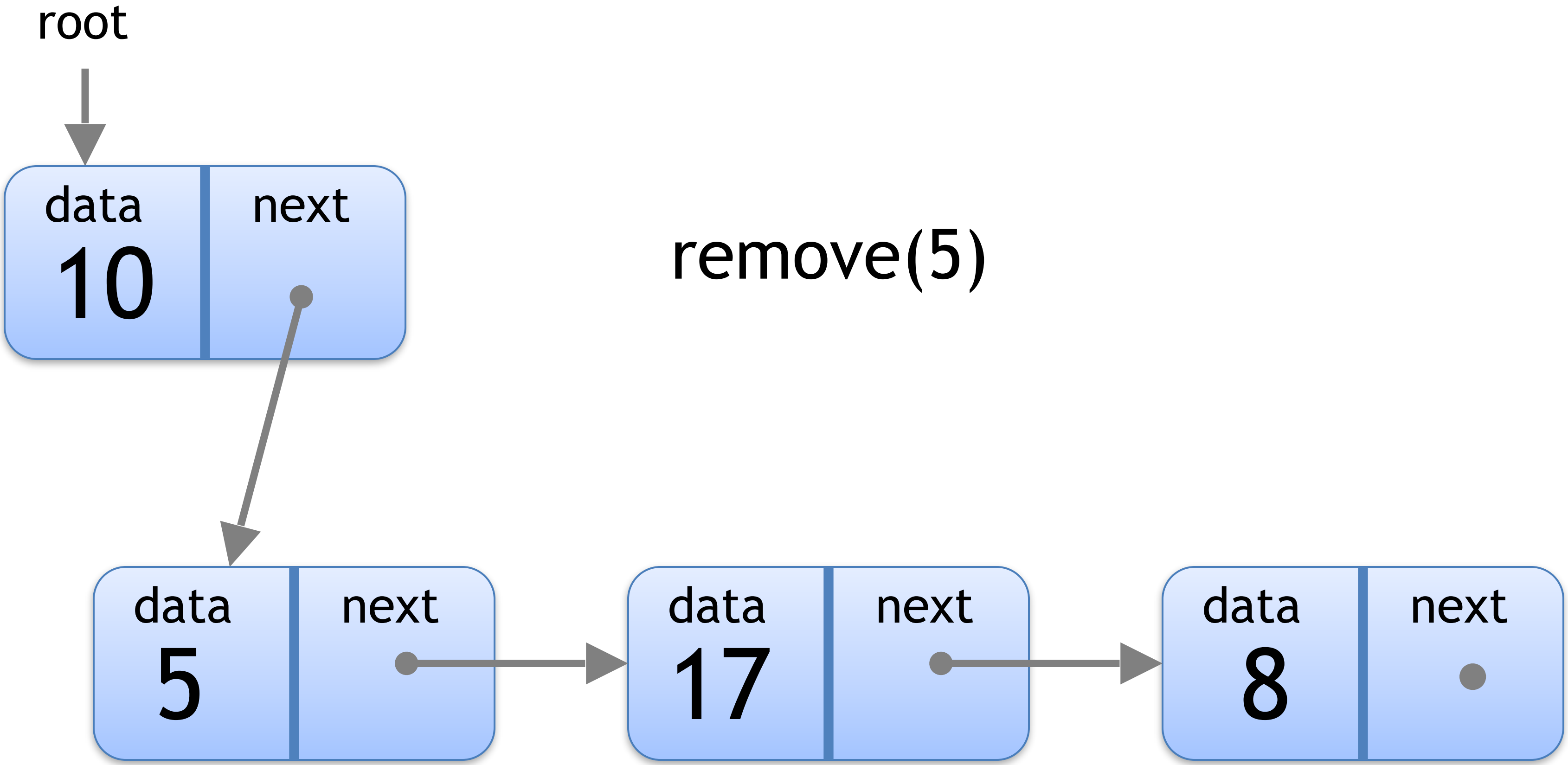
add(10)


data
10
next
add(10)
root
data
5
next
data
17
next
data
8
next



add(10)

remove(5)



root

data
10

next

data
5

next

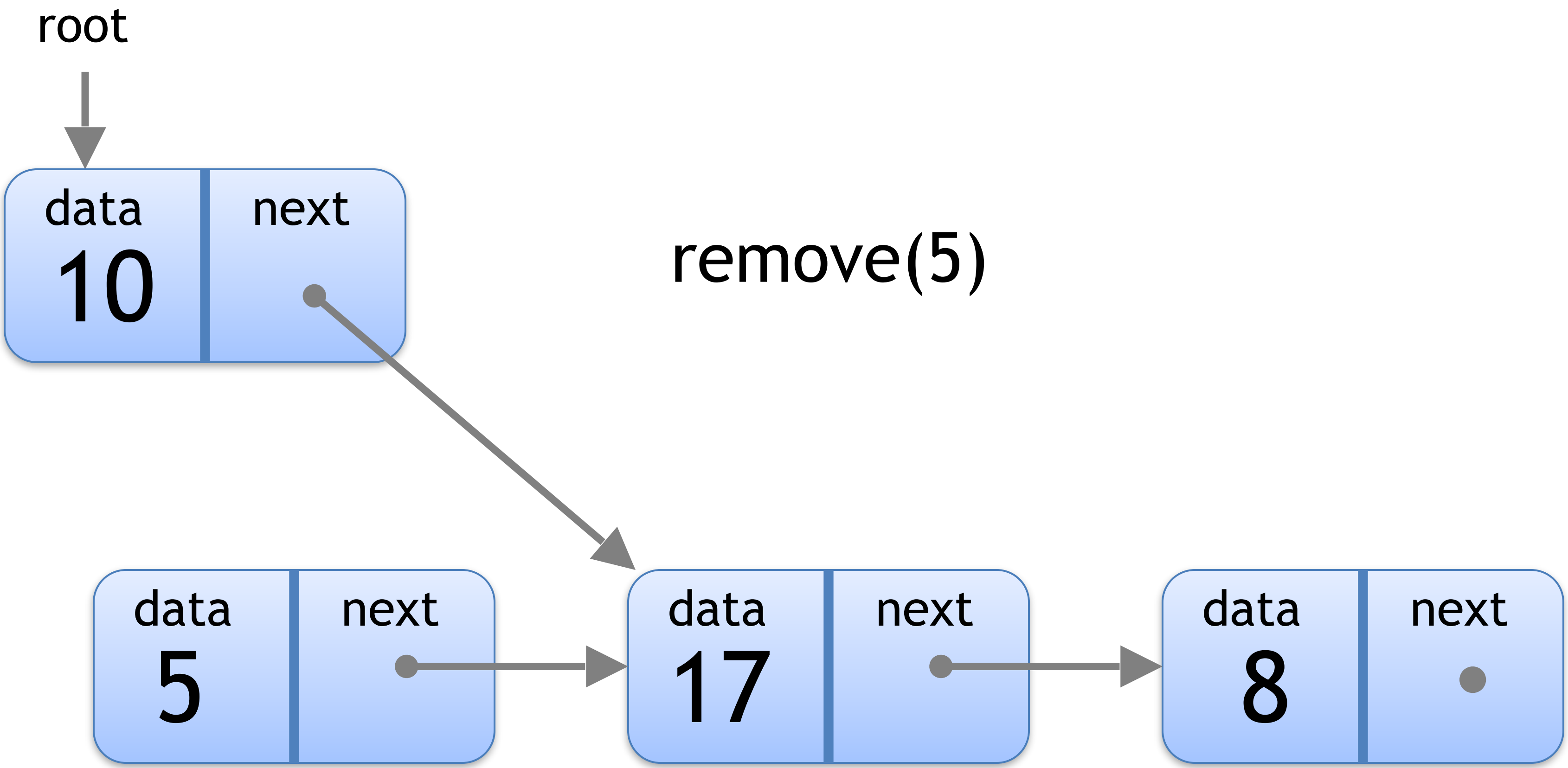
data
17

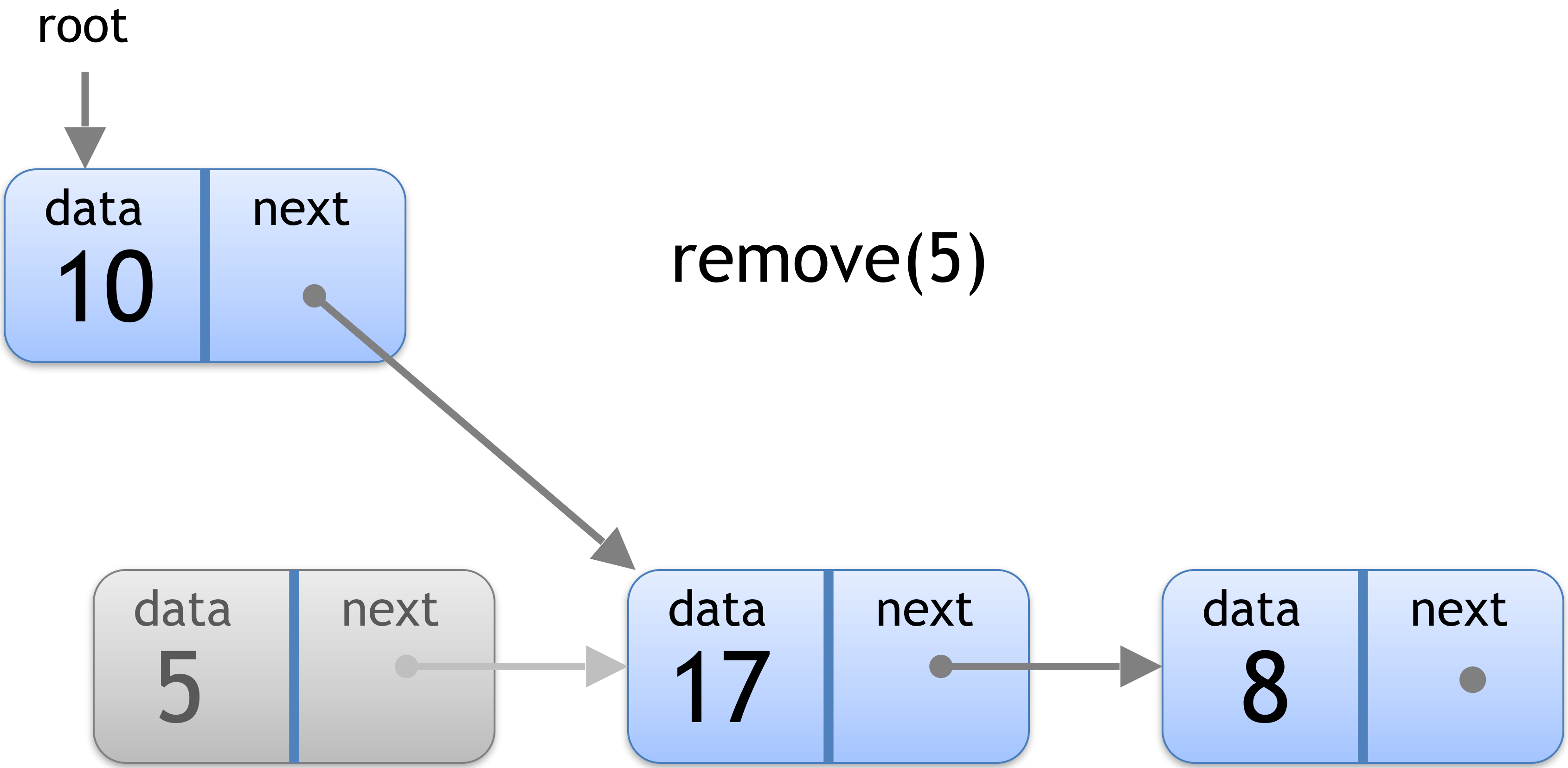
next

data
8

next

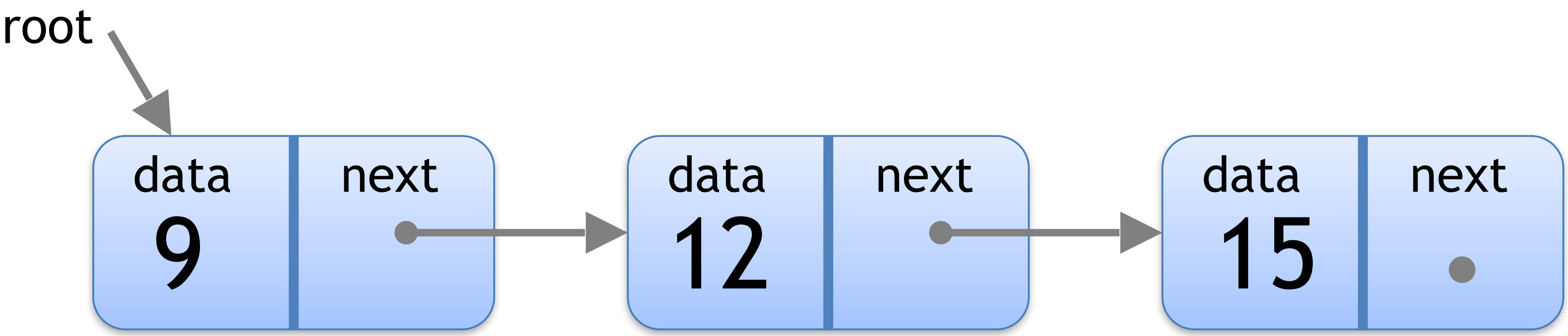
remove(5)



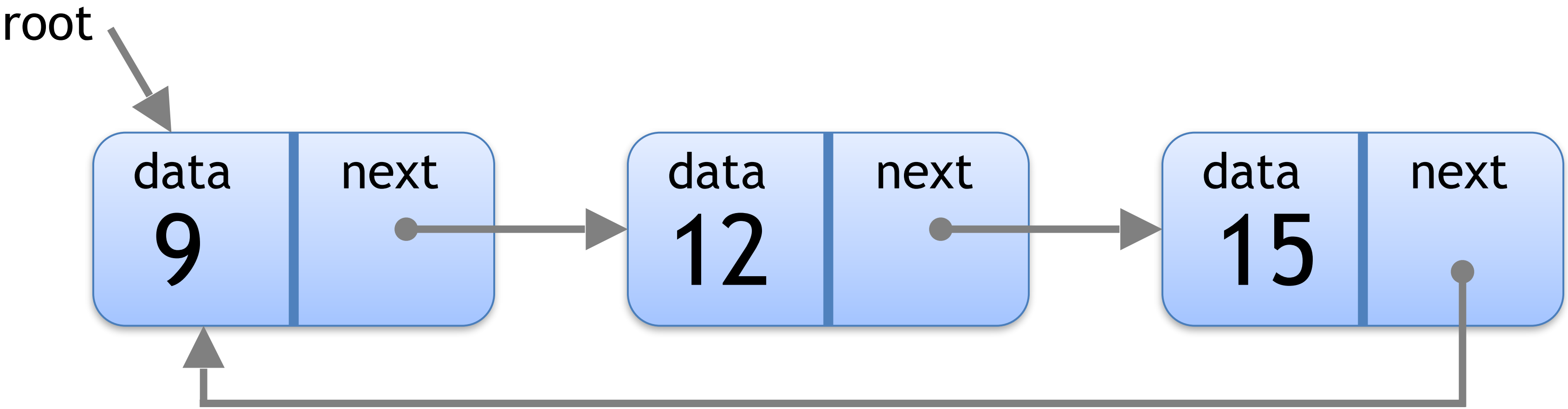

root
data
10
next
remove(5)
data
5
next
data
17
next
data
8
next

Python Circular Linked Lists

Regular Linked List

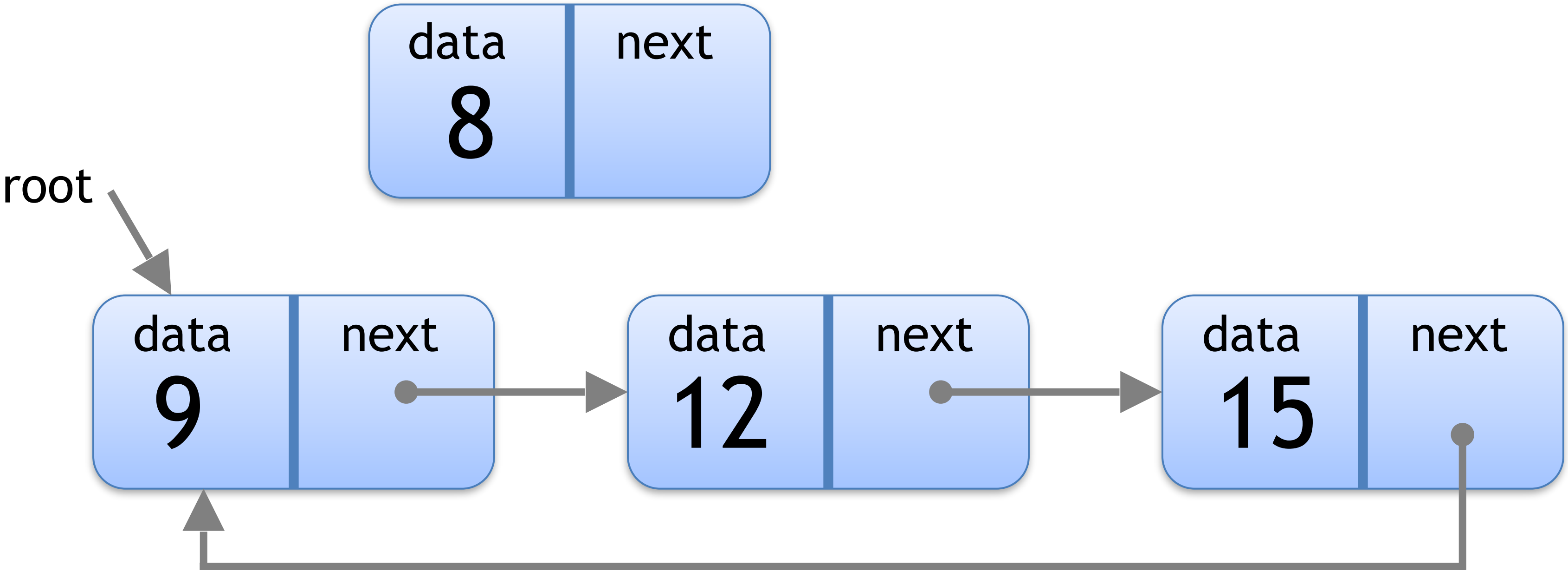


Circular Linked List



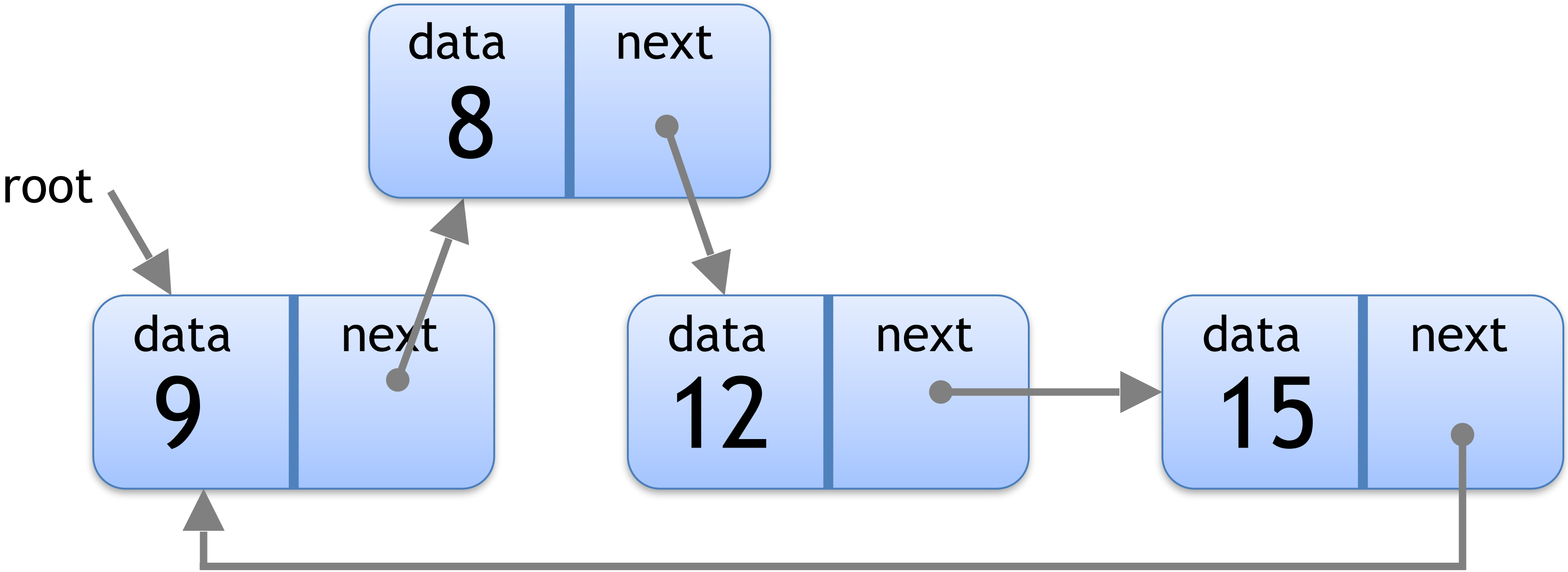
Circular Linked List

add(8)



Circular Linked List

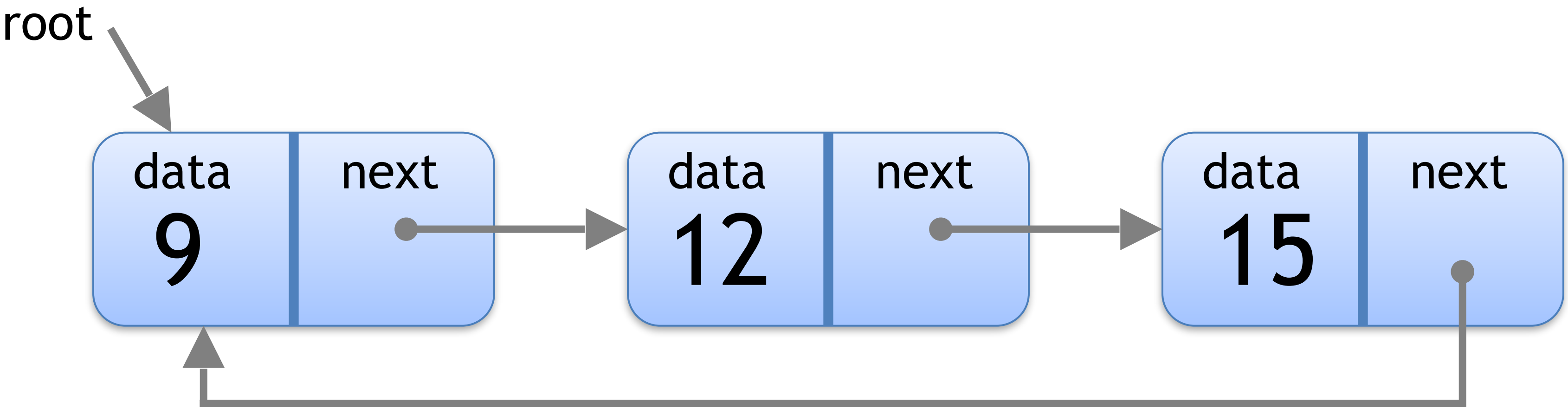
add(8)



Circular Linked List

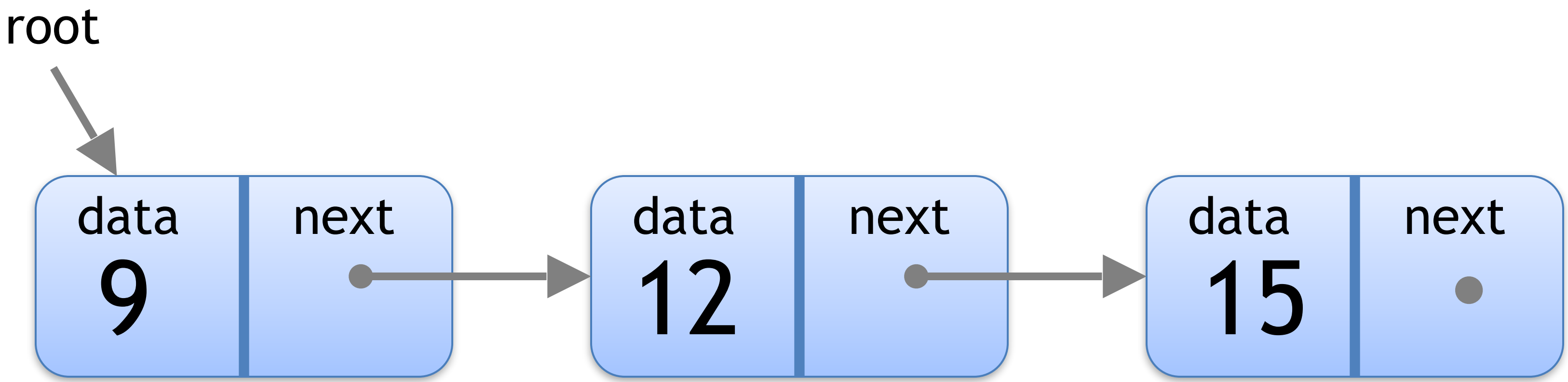
Advantage over regular (singly) linked lists:

- Ideal for modeling continuous looping objects, such as a Monopoly board or a race track.

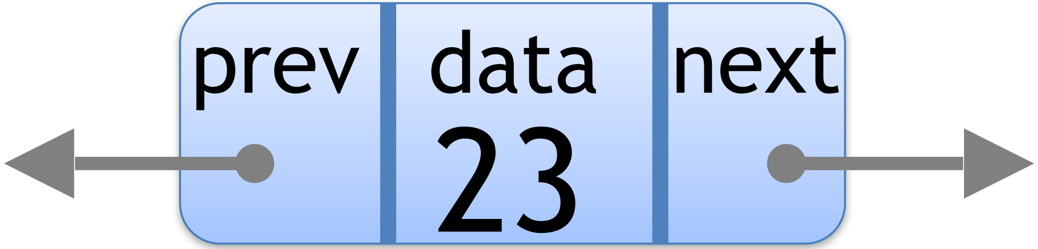


Python Doubly Linked Lists

Regular Linked List

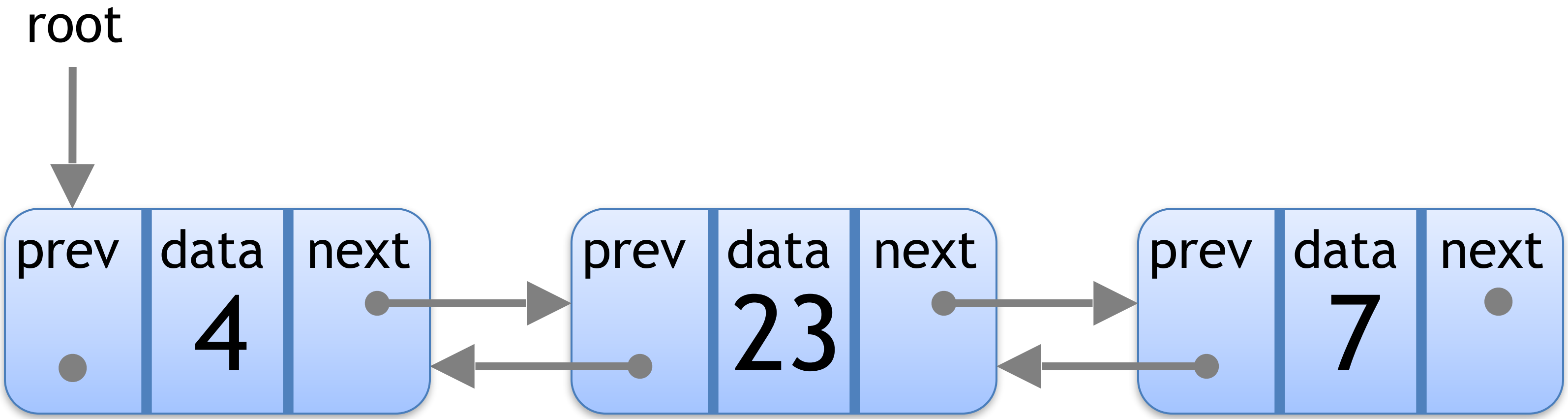


Doubly Linked List



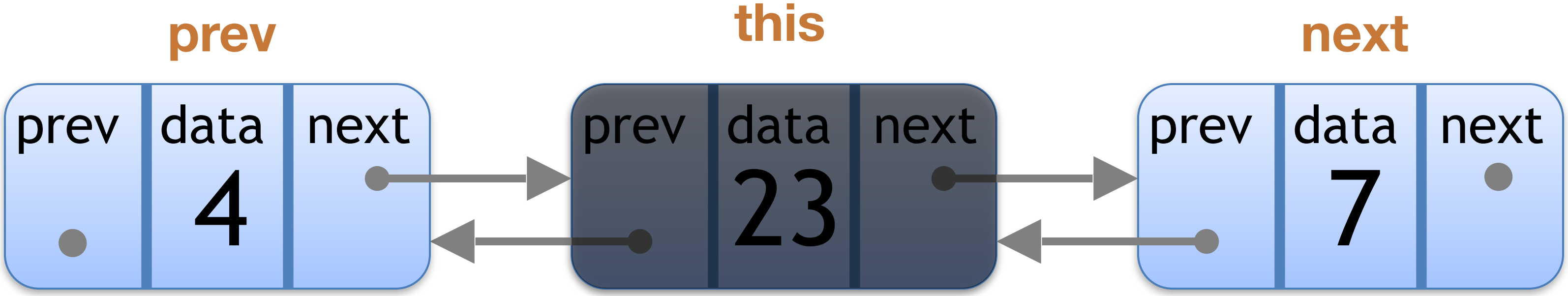
Every Node has 3 parts:
data and pointers to
previous and **next** Nodes

Doubly Linked List



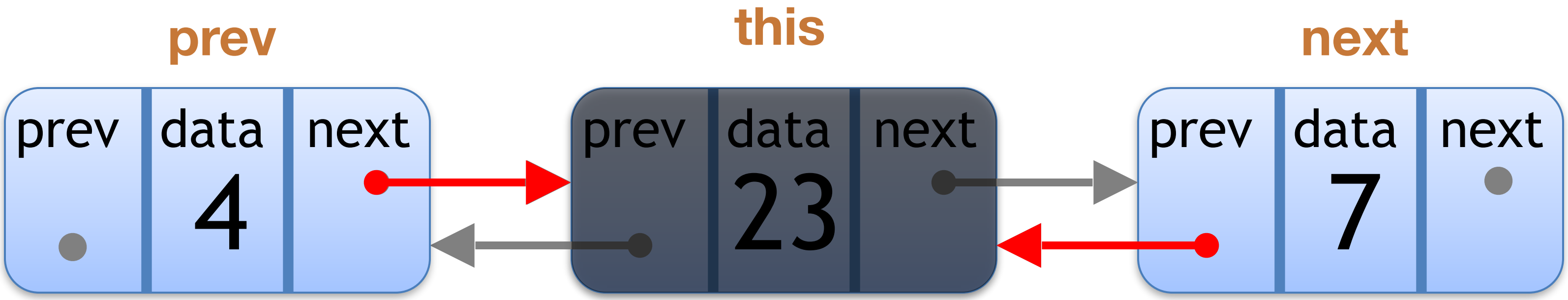
Doubly Linked List

Delete Node



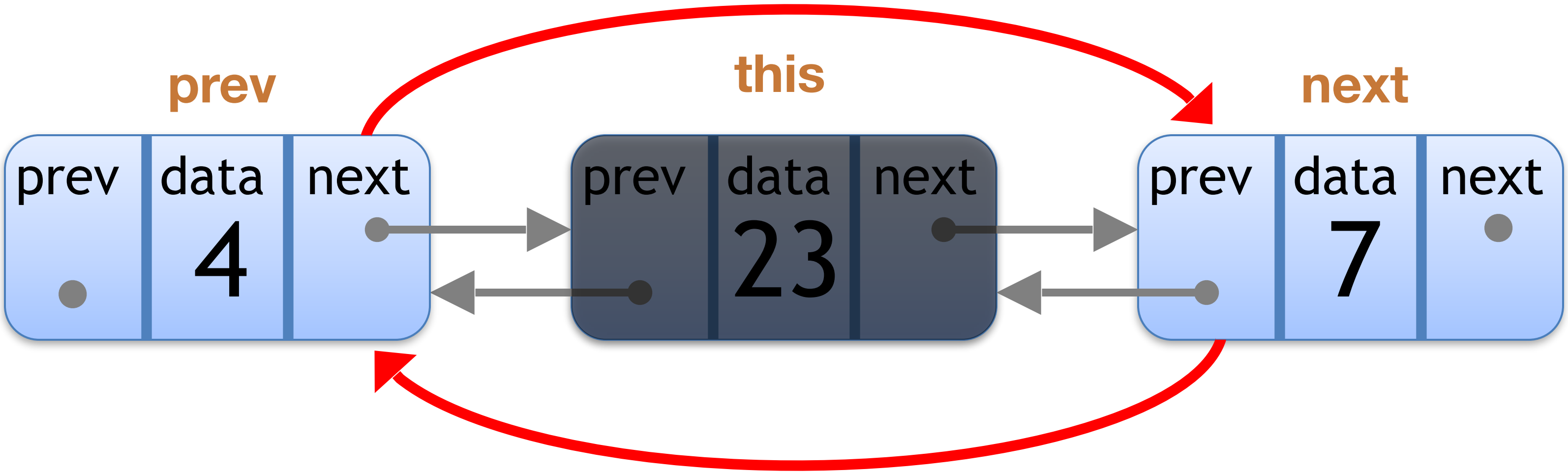
Doubly Linked List

Delete Node



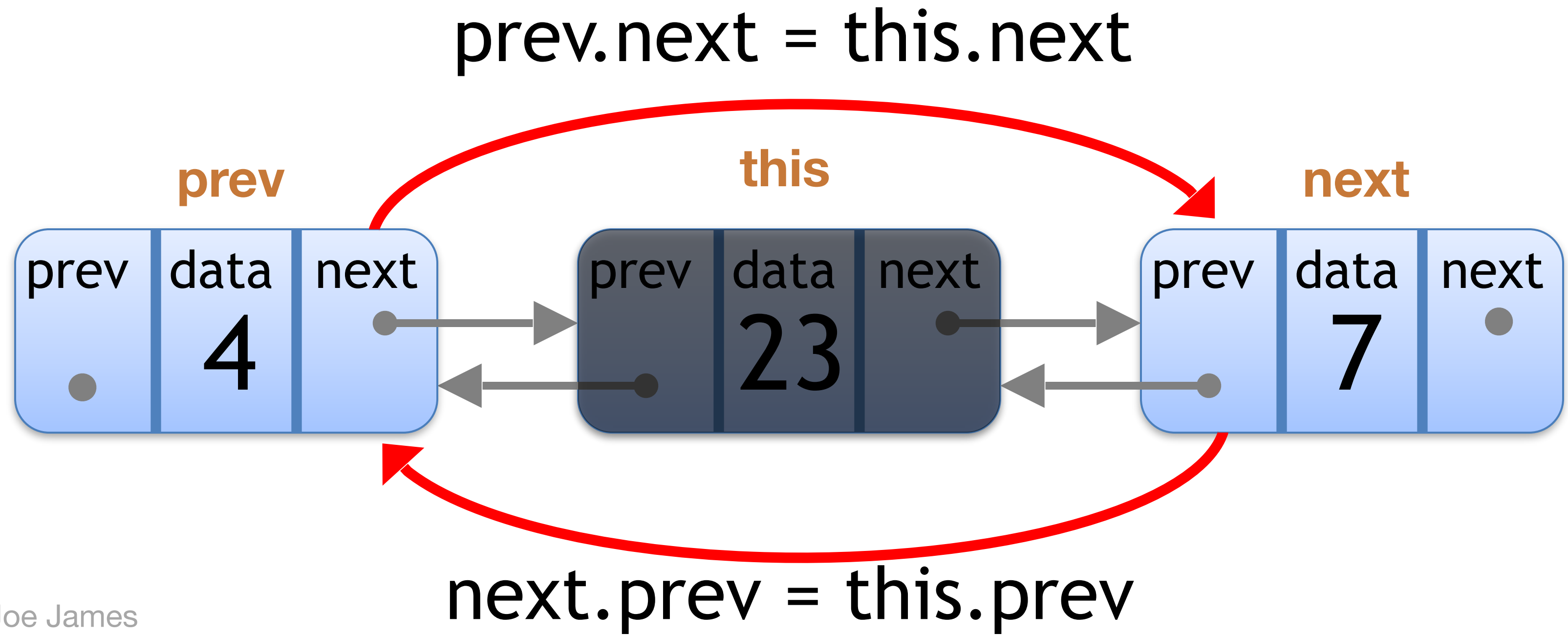
Doubly Linked List

Delete Node



Doubly Linked List

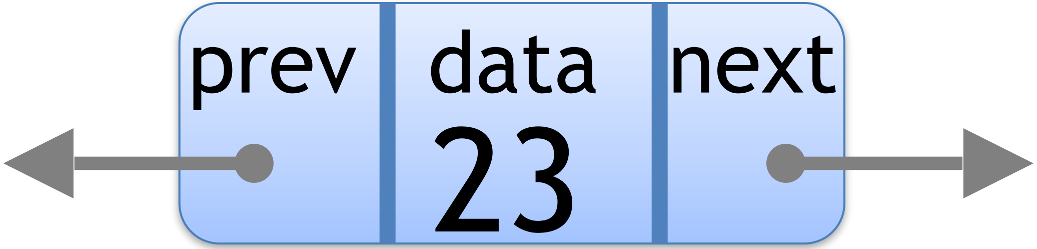
Delete Node



Doubly Linked List

Advantages over regular (singly) linked lists:

- Can iterate the list in either direction
- Can delete a node without iterating through the list (if given a pointer to the node)


prev
data
23
next