LINKED LISTS

LECTURE 10-1

JIM FIX, REED COLLEGE CSCI 121

COURSE INFO

- **Today:**
  - we look at our first link-based data structure, *linked lists*
  - we will soon look at another, *search trees*

A NODE CLASS

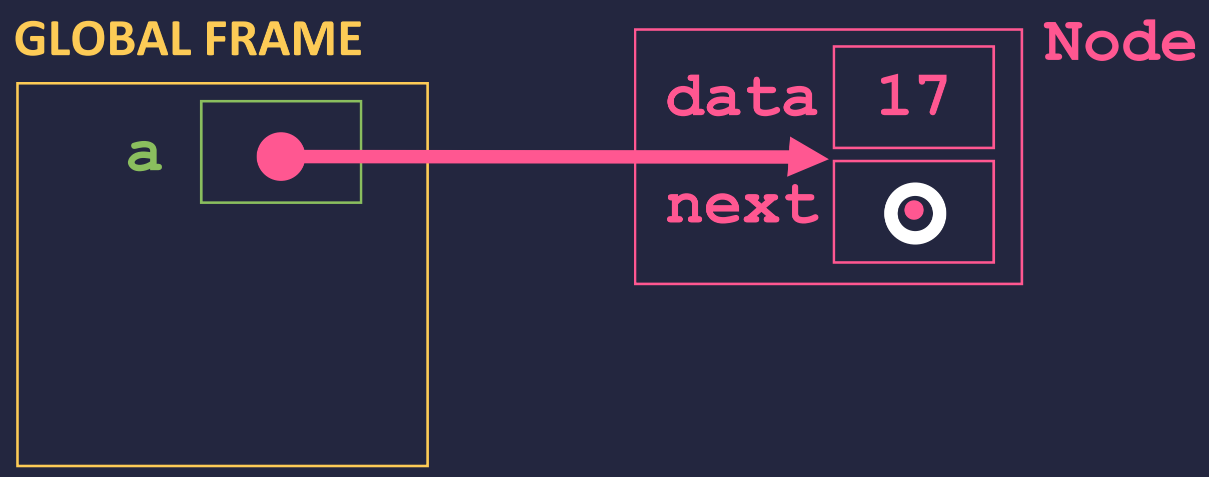
```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None
```

```
>>>
```

GLOBAL FRAME

A NODE CLASS

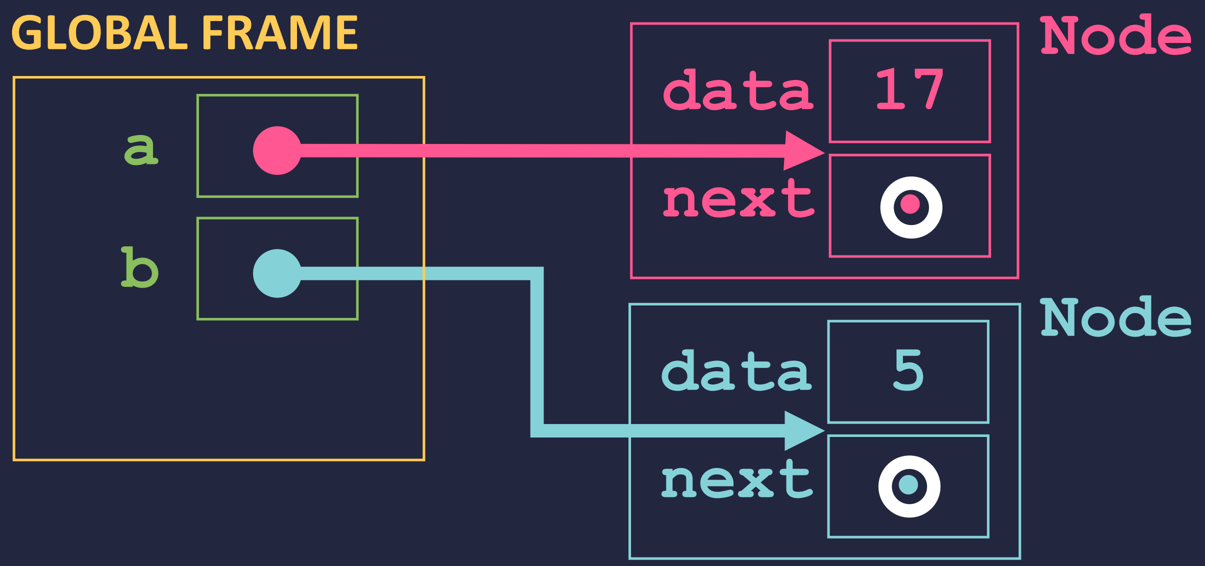
```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None

>>> a = Node(17)
>>>
```



A NODE CLASS

```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None
```

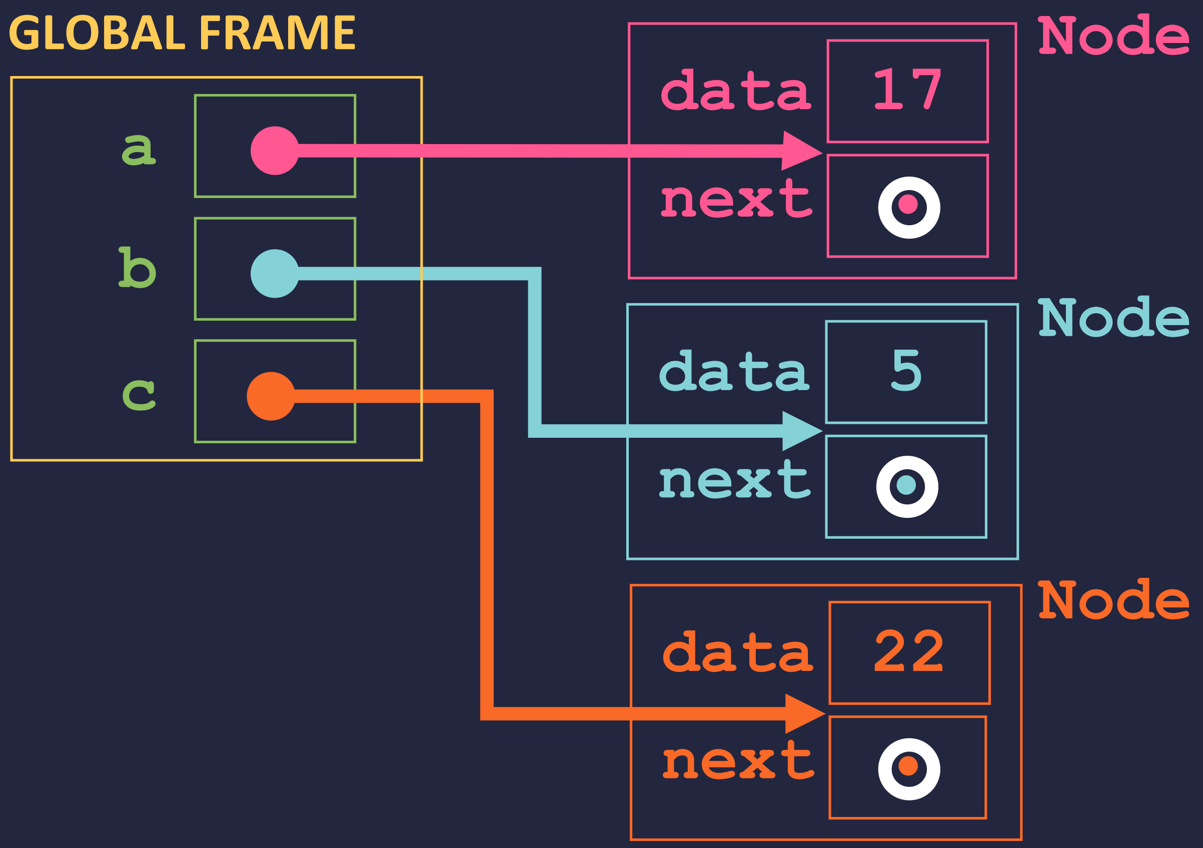
```
>>> a = Node(17)
>>> b = Node(5)
>>>
```



A NODE CLASS

```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None
```

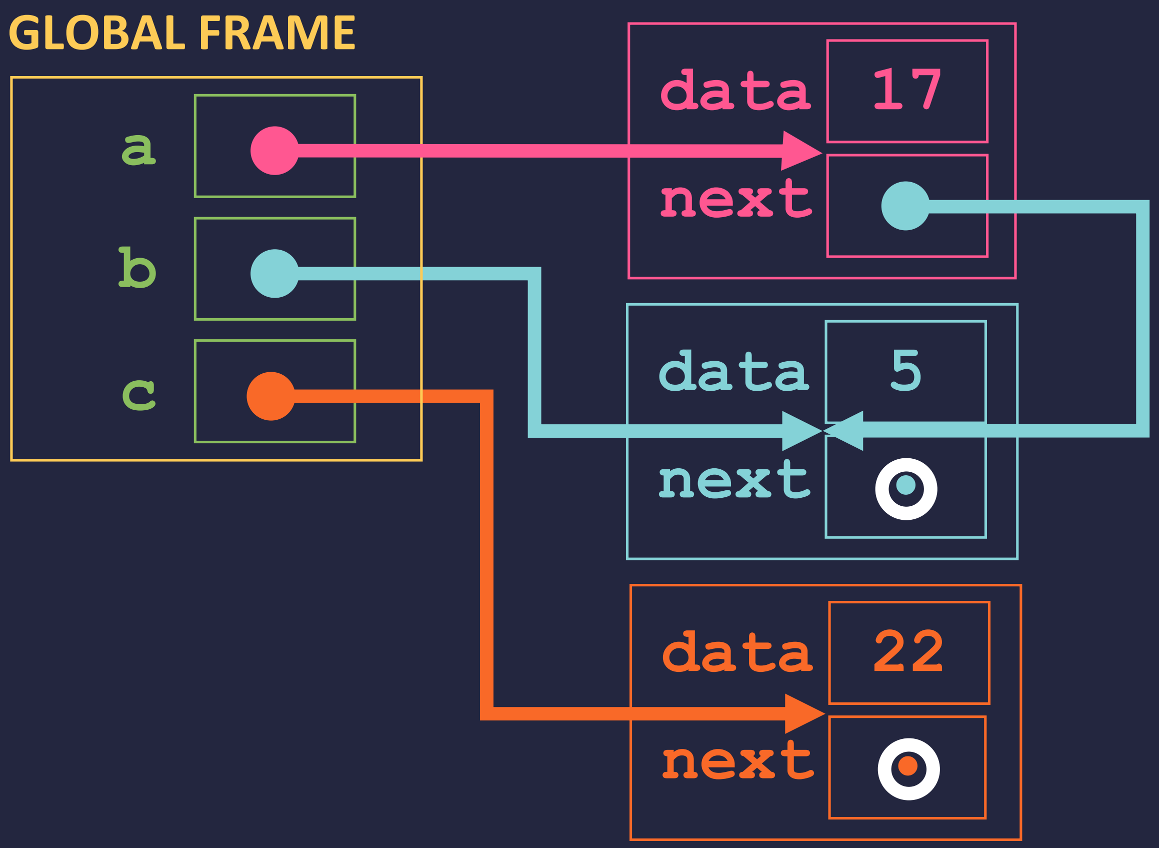
```
>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>>
```



LINKING NODES IN SERIES

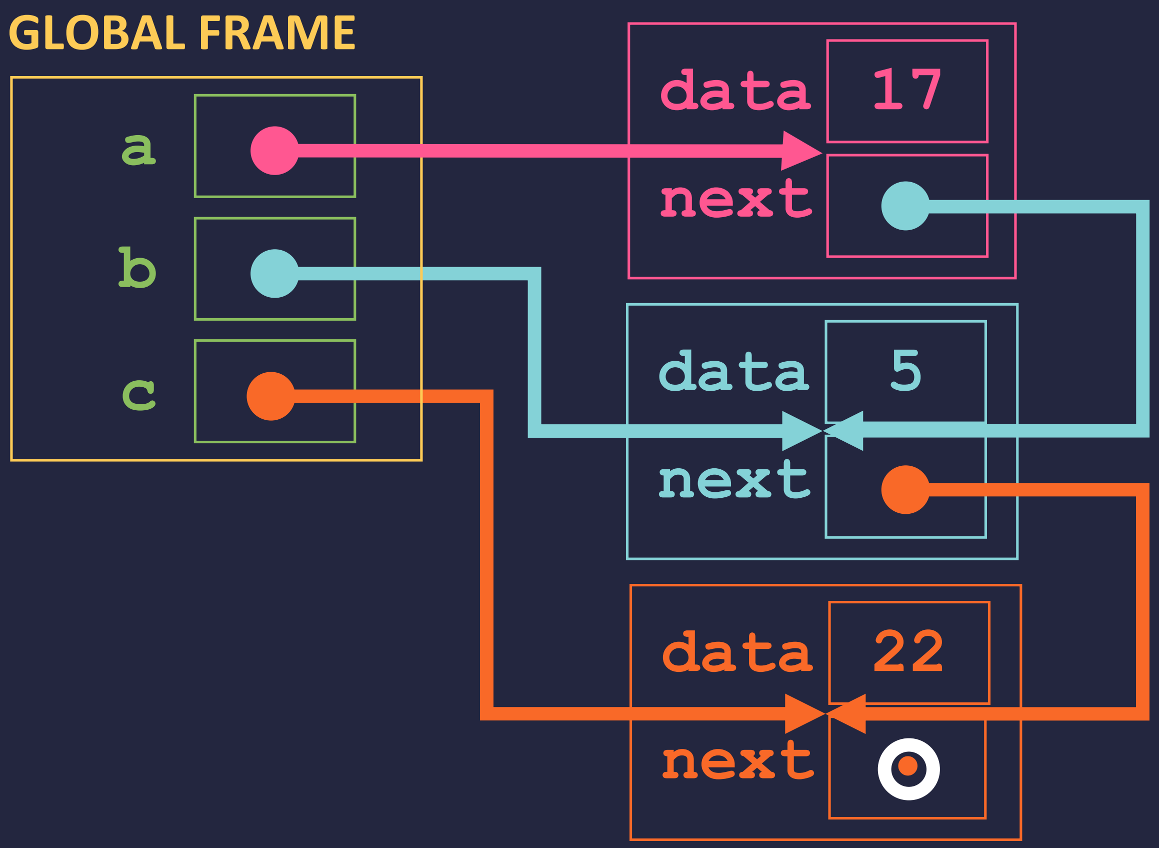
```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None
```

```
>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> a.next = b
>>>
```



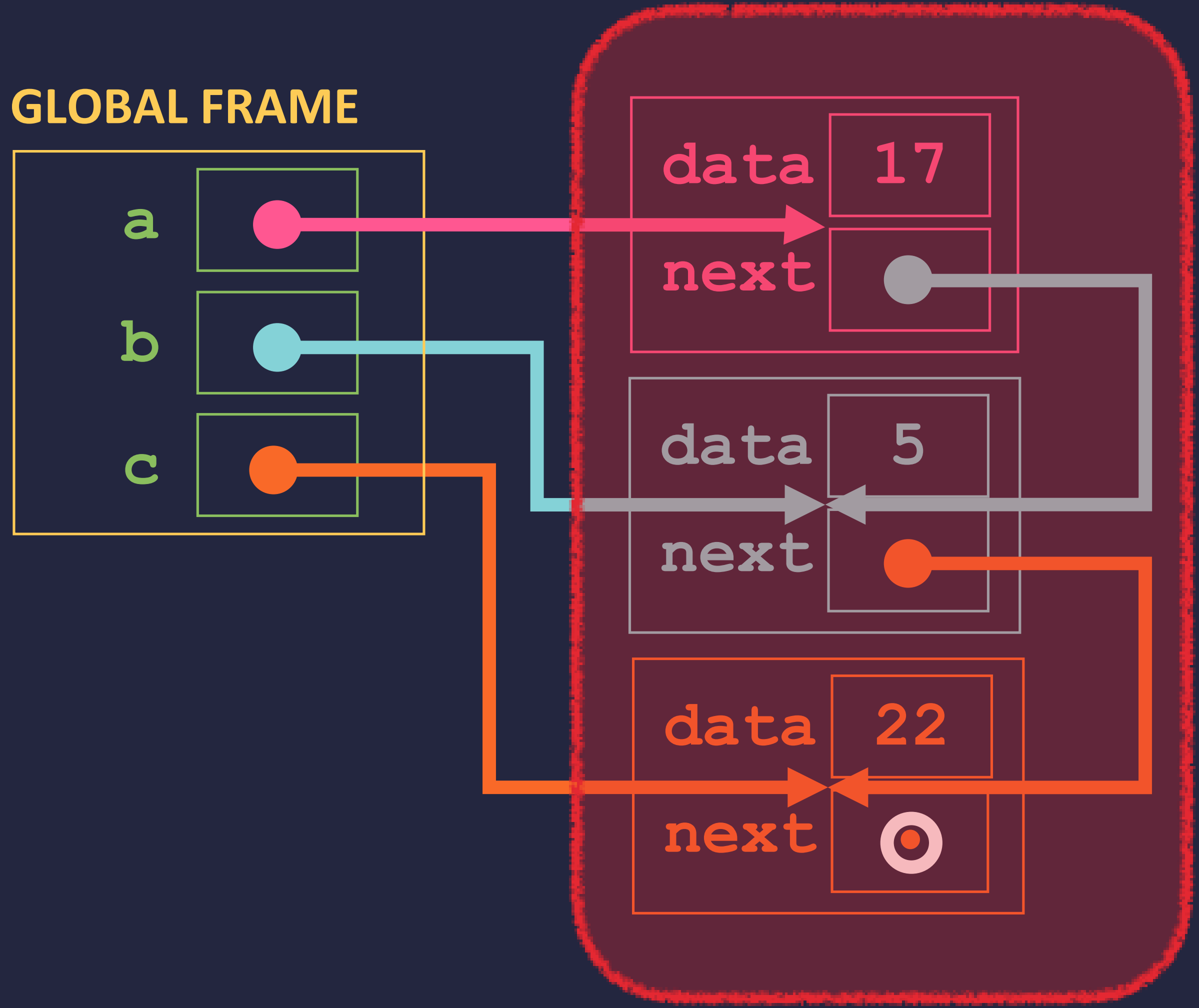
LINKING NODES IN SERIES

```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> a.next = b
>>> b.next = c
>>>
```



LINKING NODES IN SERIES

```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> a.next = b
>>> b.next = c
>>>
```

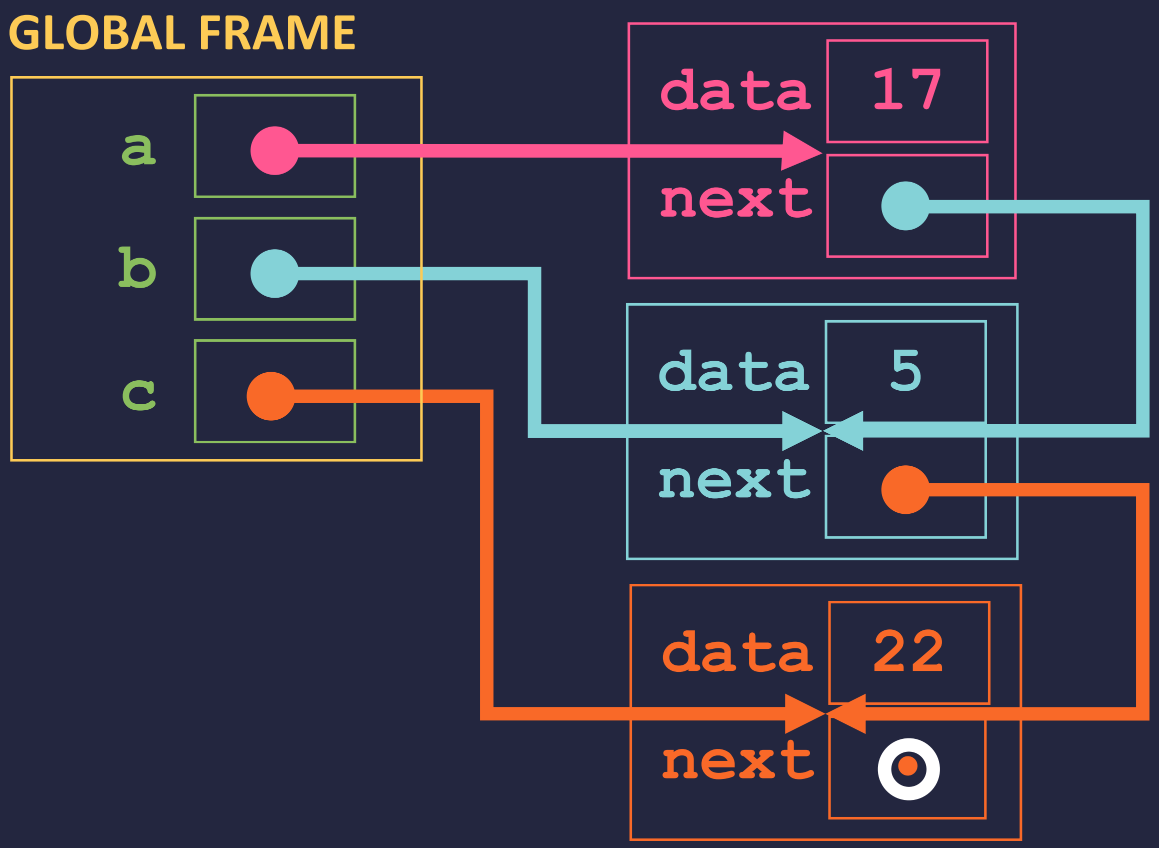


THIS STRUCTURE IS CALLED A LINKED LIST

FOLLOWING LINKS

```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None
```

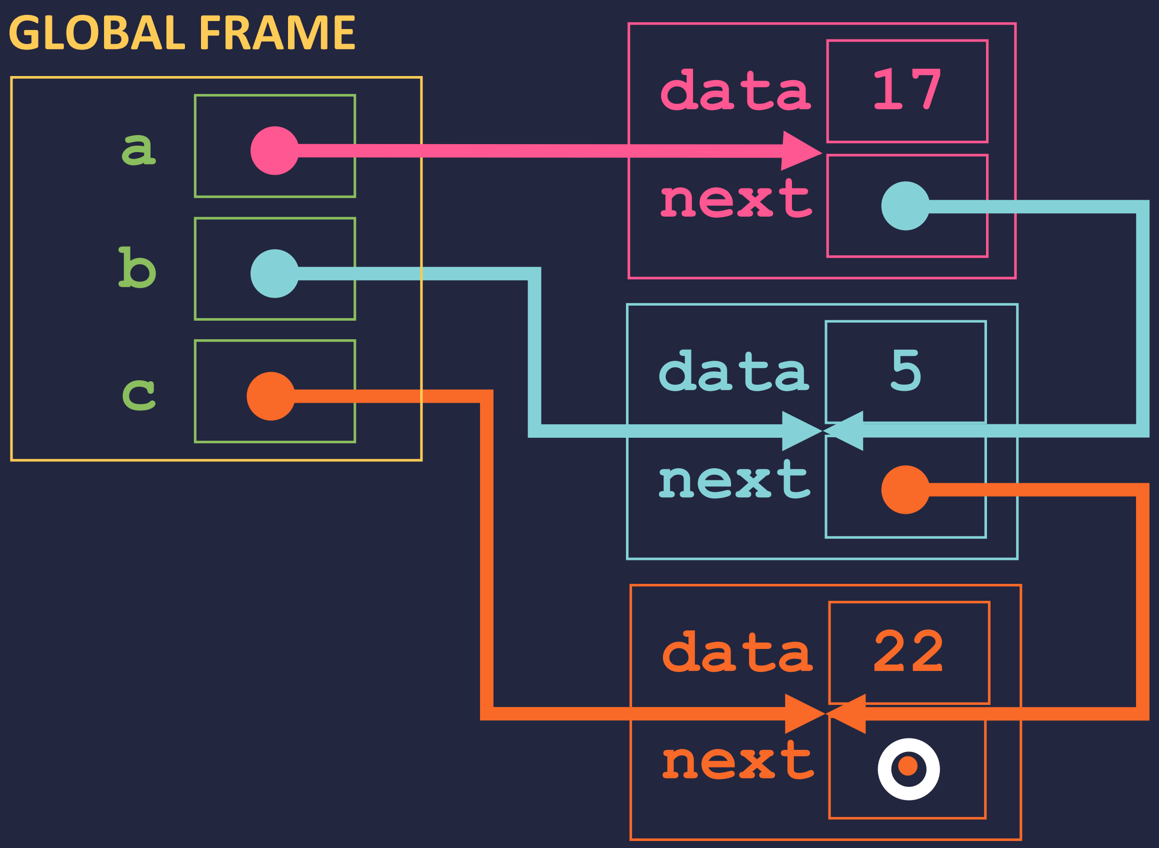
```
>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> a.next = b
>>> b.next = c
>>> a.value
17
>>> b.value
5
>>> c.value
22
>>> a.next.value
5
```



FOLLOWING LINKS

```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None
```

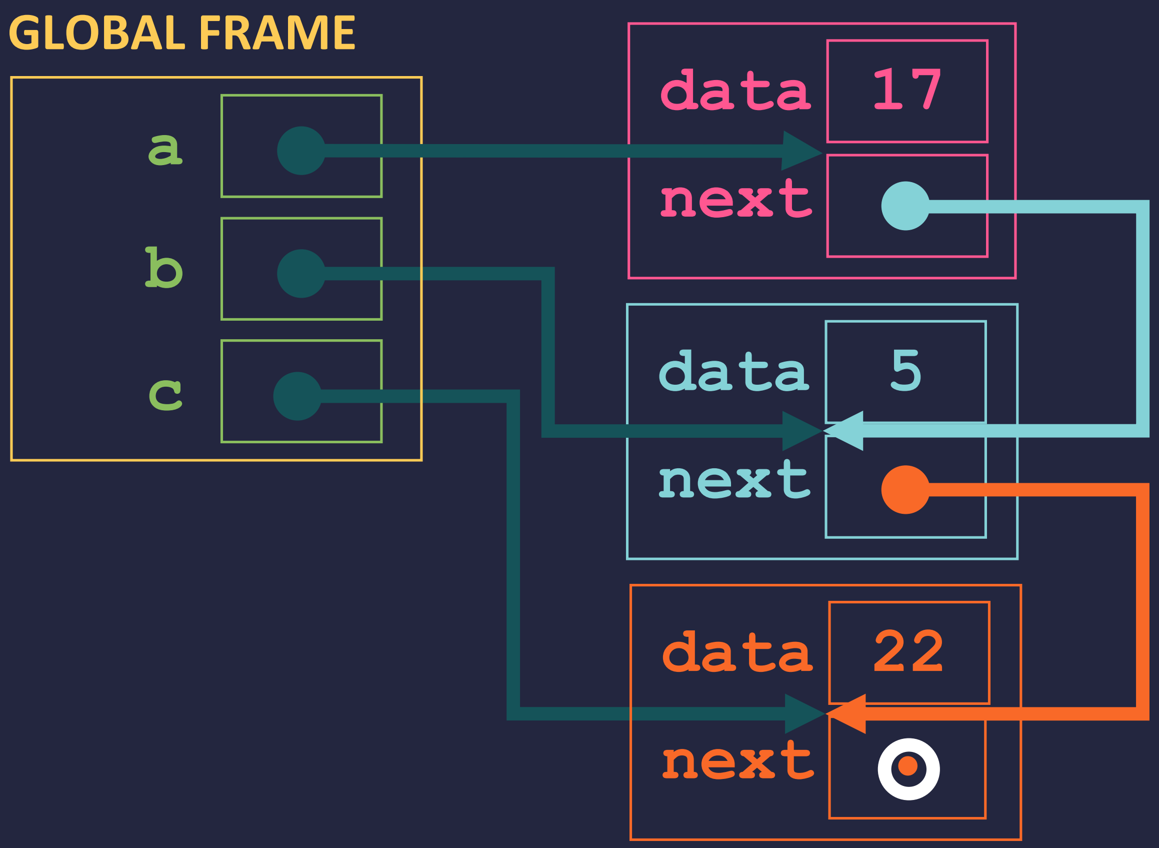
```
>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> a.next = b
>>> b.next = c
>>> a.value
17
>>> b.value
5
>>> c.value
22
>>> a.next.value
5
>>> a.next.next.value
22
```



LINKED LISTS

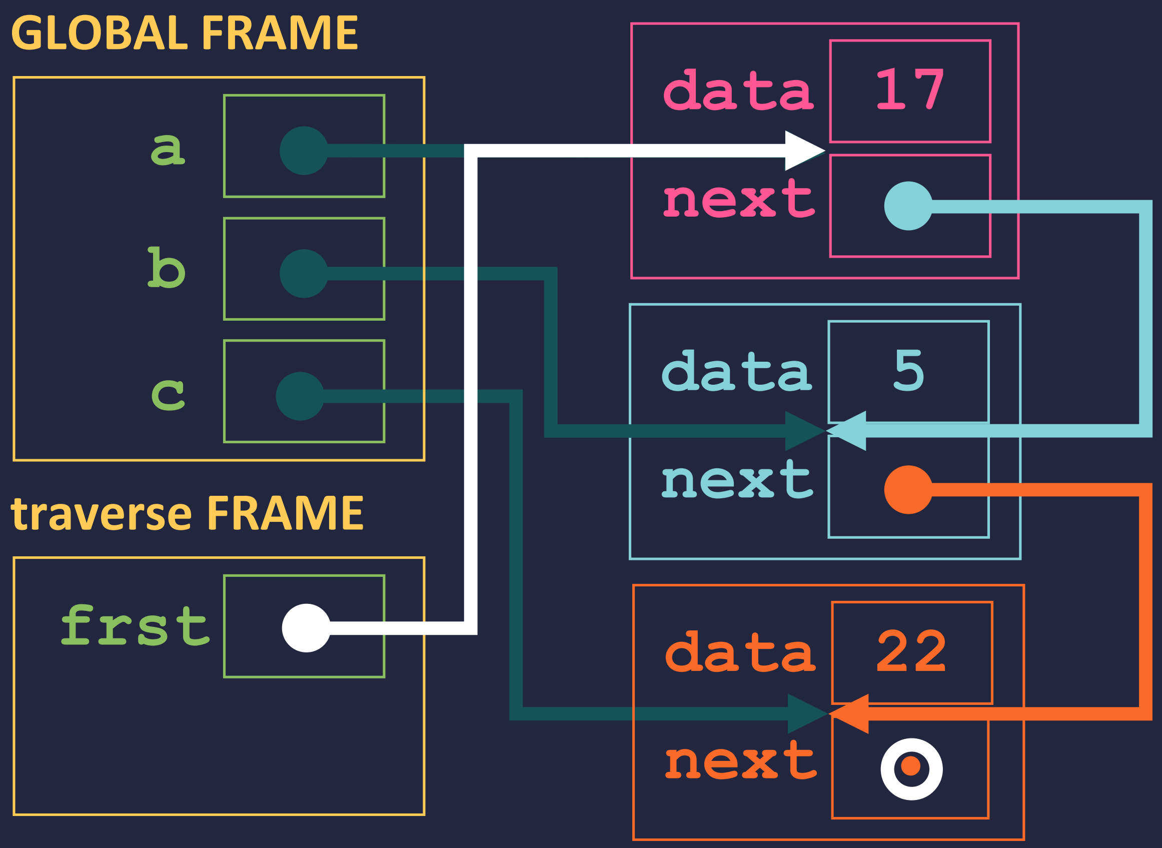
TRAVERSING A LINKED LIST

```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None

def traverse(frst):
    curr = frst
    while curr is not None:
        print(curr.value)
        curr = curr.next

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> traverse(a)
```



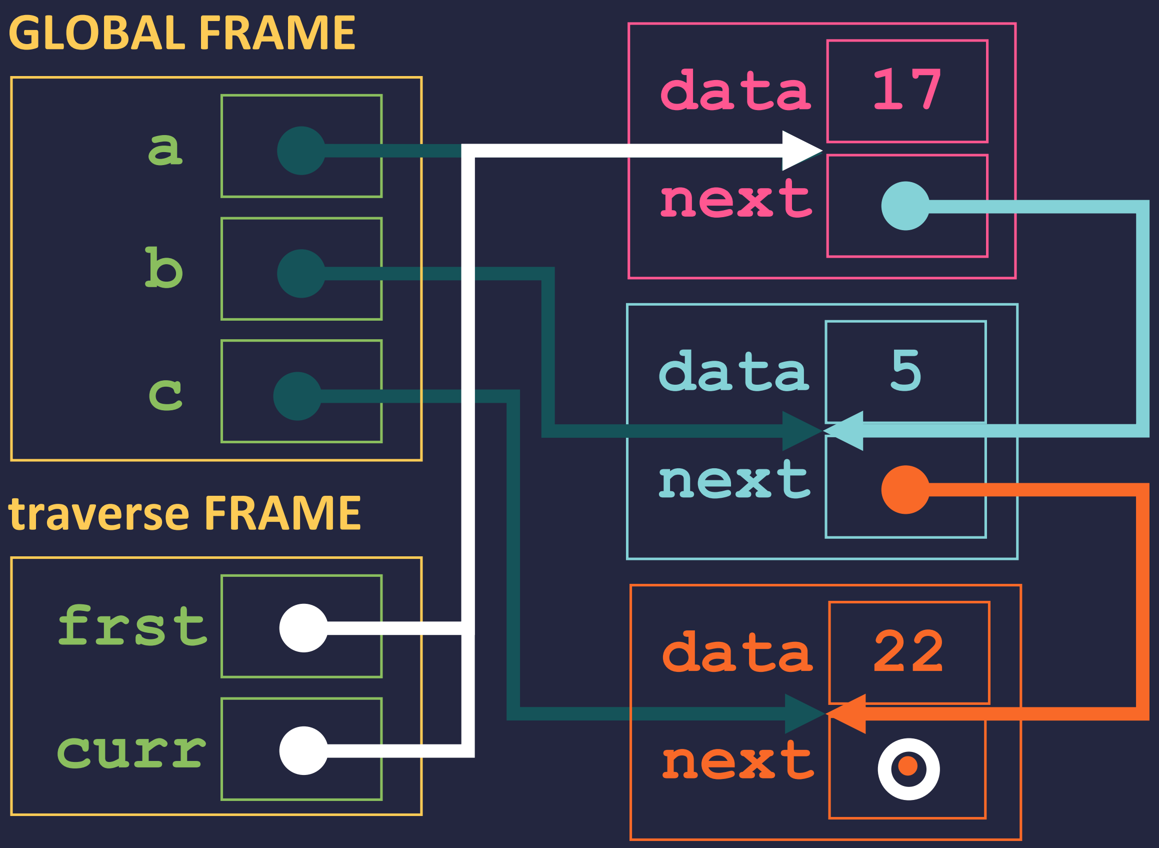
TRAVERSING A LINKED LIST

```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None

def traverse(frst):
    curr = frst
    while curr is not None:
        print(curr.value)
        curr = curr.next

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> traverse(a)
```



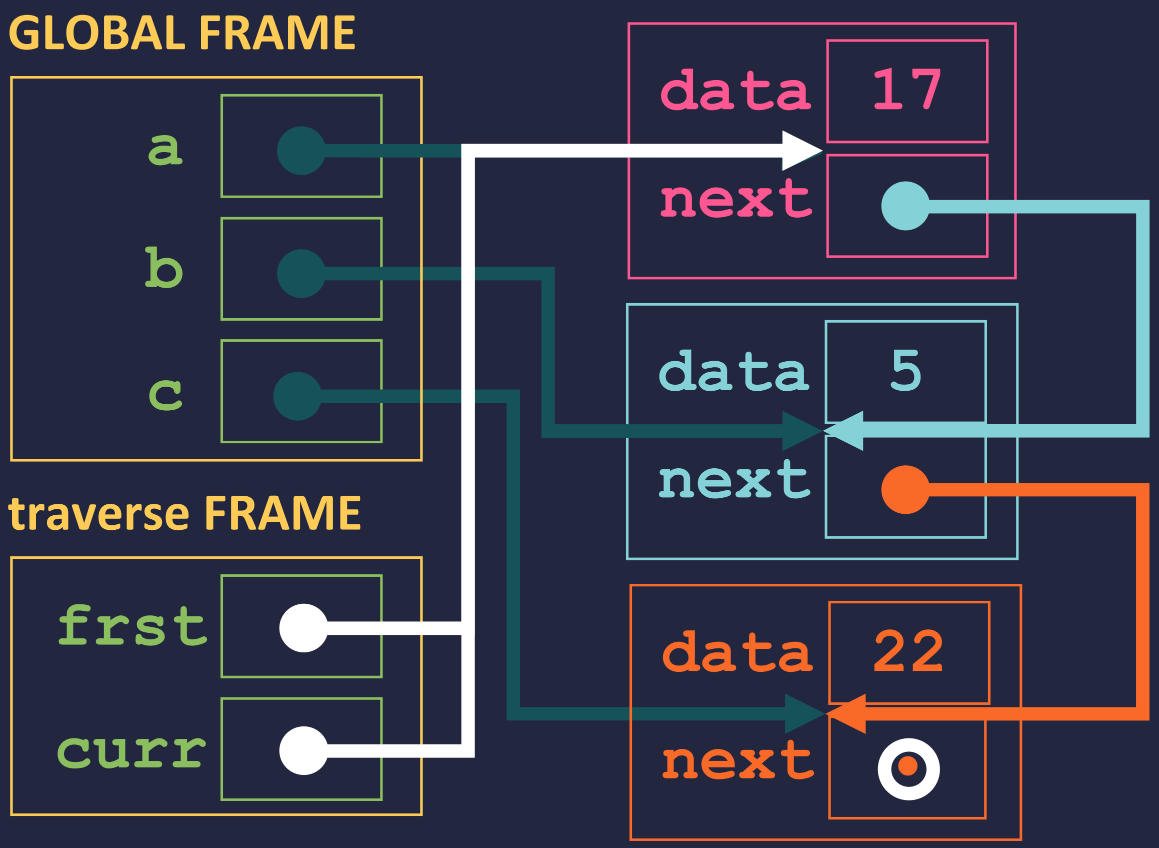
TRAVERSING A LINKED LIST

```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None

def traverse(frst):
    curr = frst
    while curr is not None:
        print(curr.value)
        curr = curr.next

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> traverse(a)
```



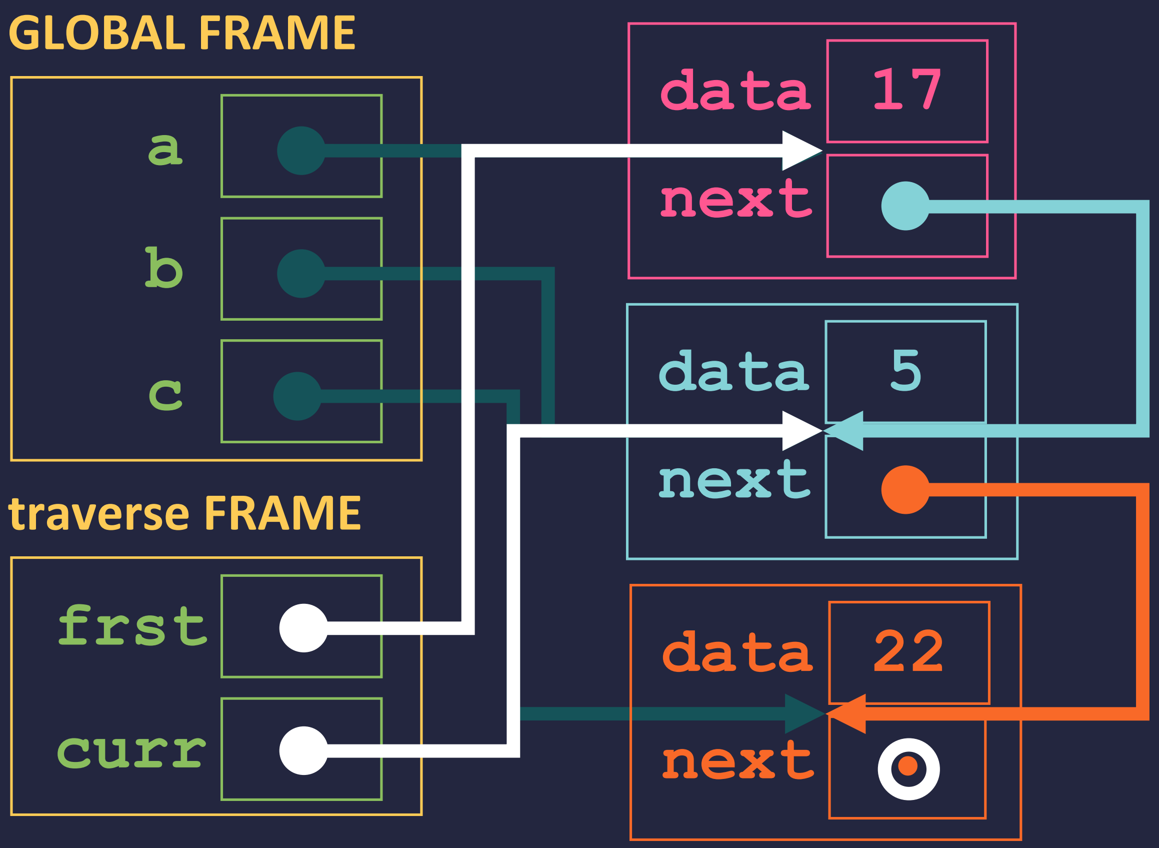
TRAVERSING A LINKED LIST

```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None

def traverse(frst):
    curr = frst
    while curr is not None:
        print(curr.value)
        curr = curr.next

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> traverse(a)
17
```



TRAVERSING A LINKED LIST

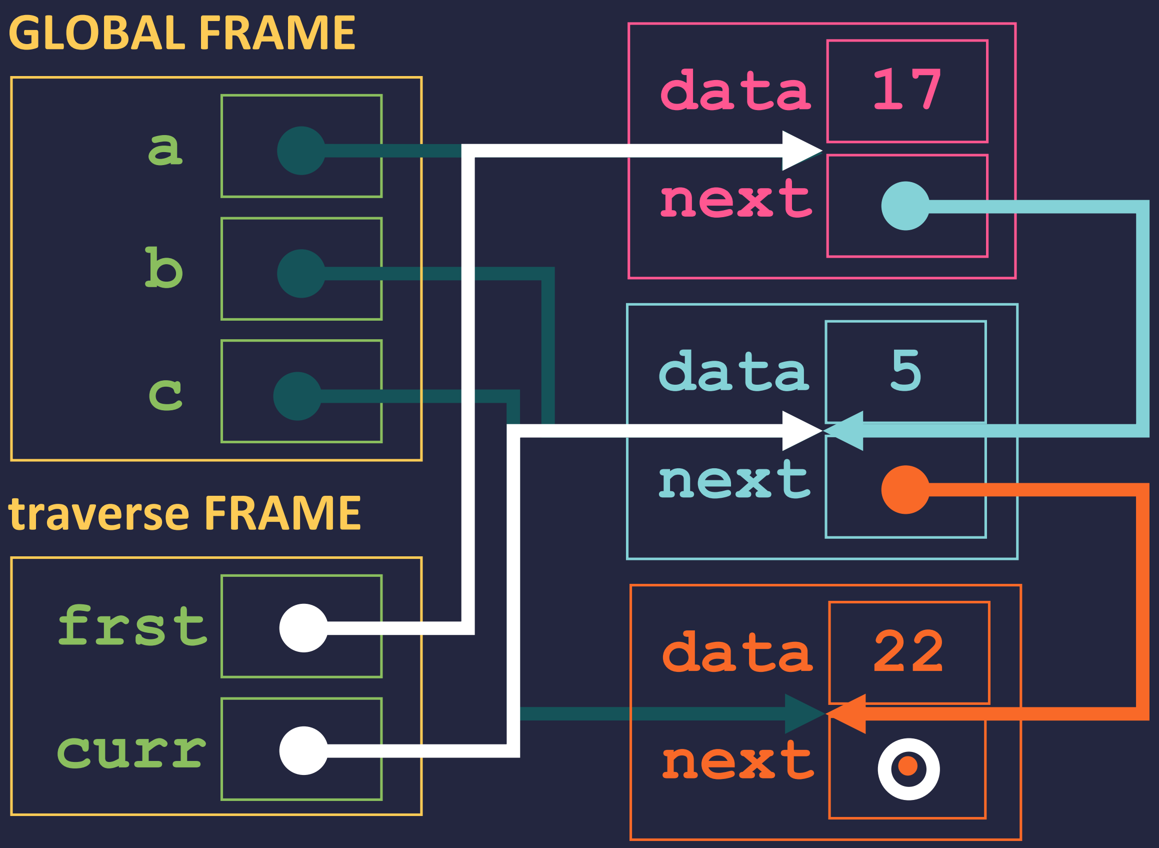
```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None

def traverse(frst):
    curr = frst
    while curr is not None:
        print(curr.value)
        curr = curr.next

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> traverse(a)
17
```



TRAVERSING A LINKED LIST

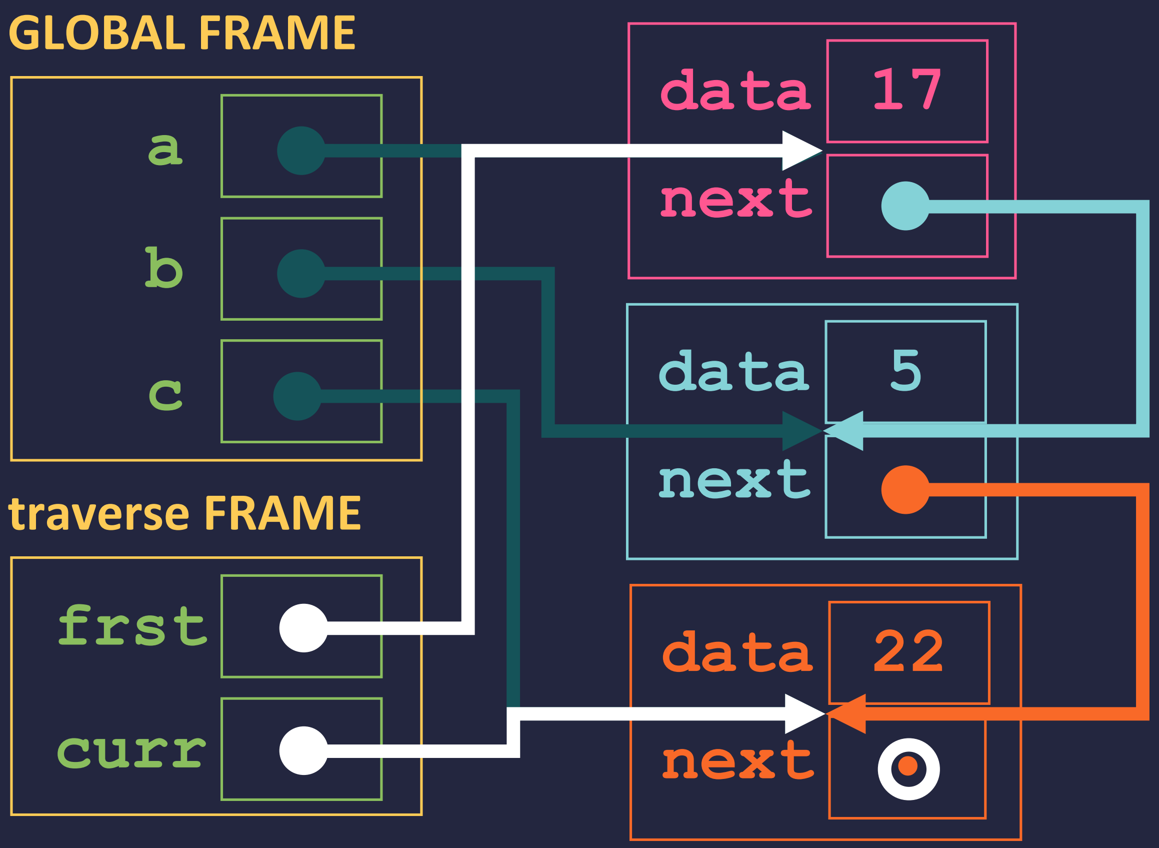
```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None

def traverse(frst):
    curr = frst
    while curr is not None:
        print(curr.value)
        curr = curr.next
```

```
>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> traverse(a)
17
5
```



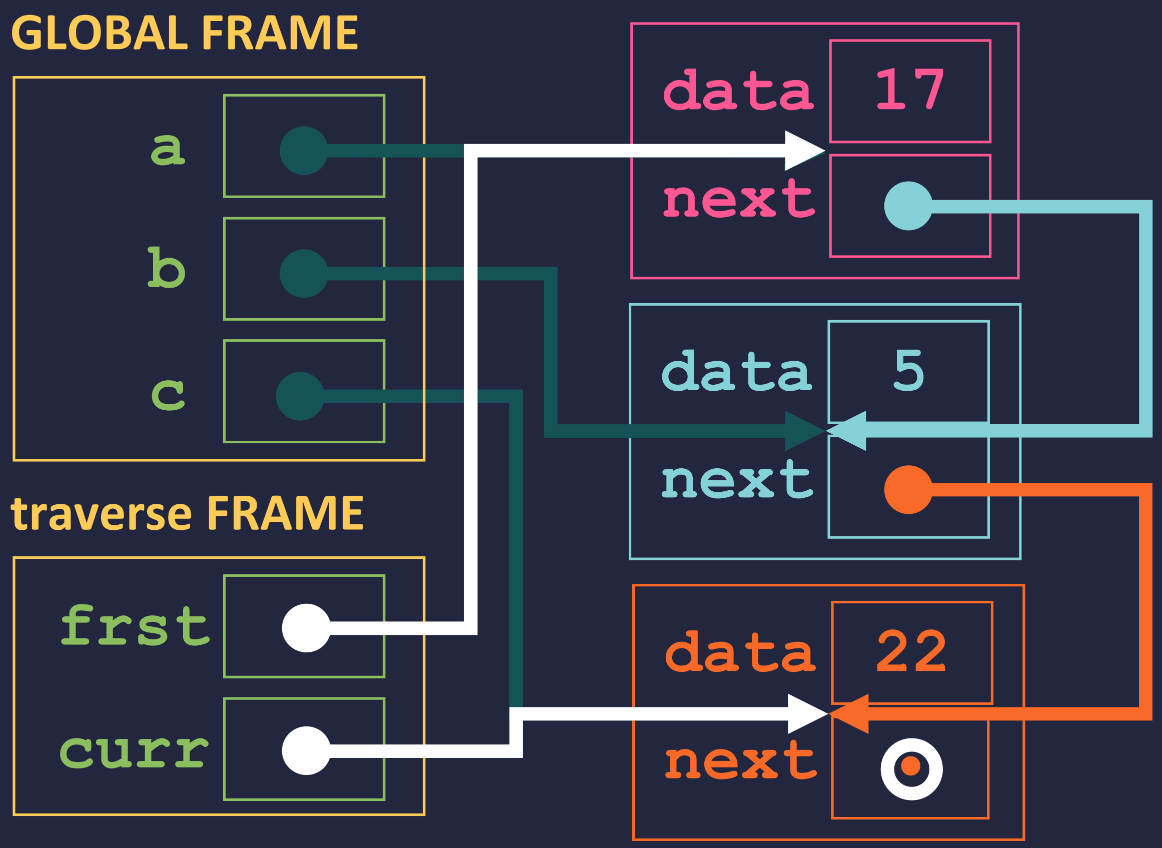
TRAVERSING A LINKED LIST

```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None

def traverse(frst):
    curr = frst
    while curr is not None:
        print(curr.value)
        curr = curr.next

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> traverse(a)
17
5
```



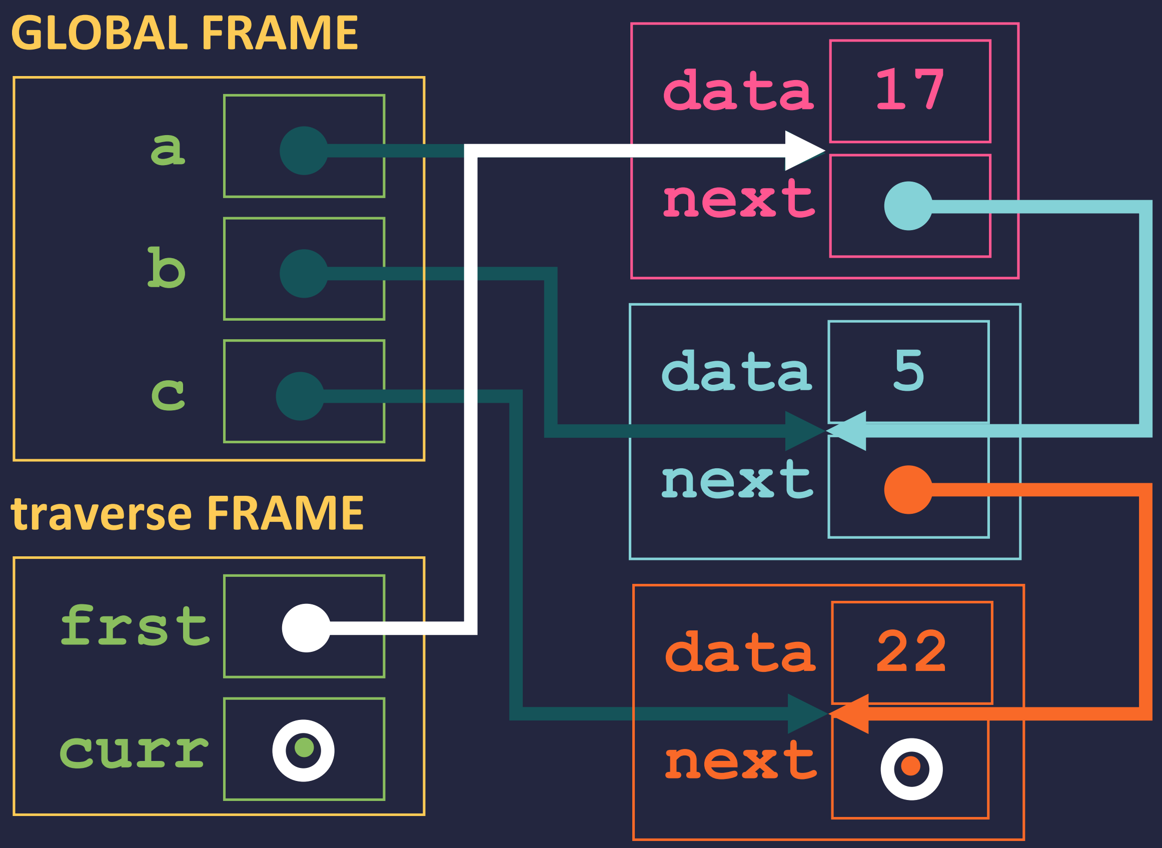
TRAVERSING A LINKED LIST

```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None

def traverse(frst):
    curr = frst
    while curr is not None:
        print(curr.value)
        curr = curr.next

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> traverse(a)
17
5
22
```



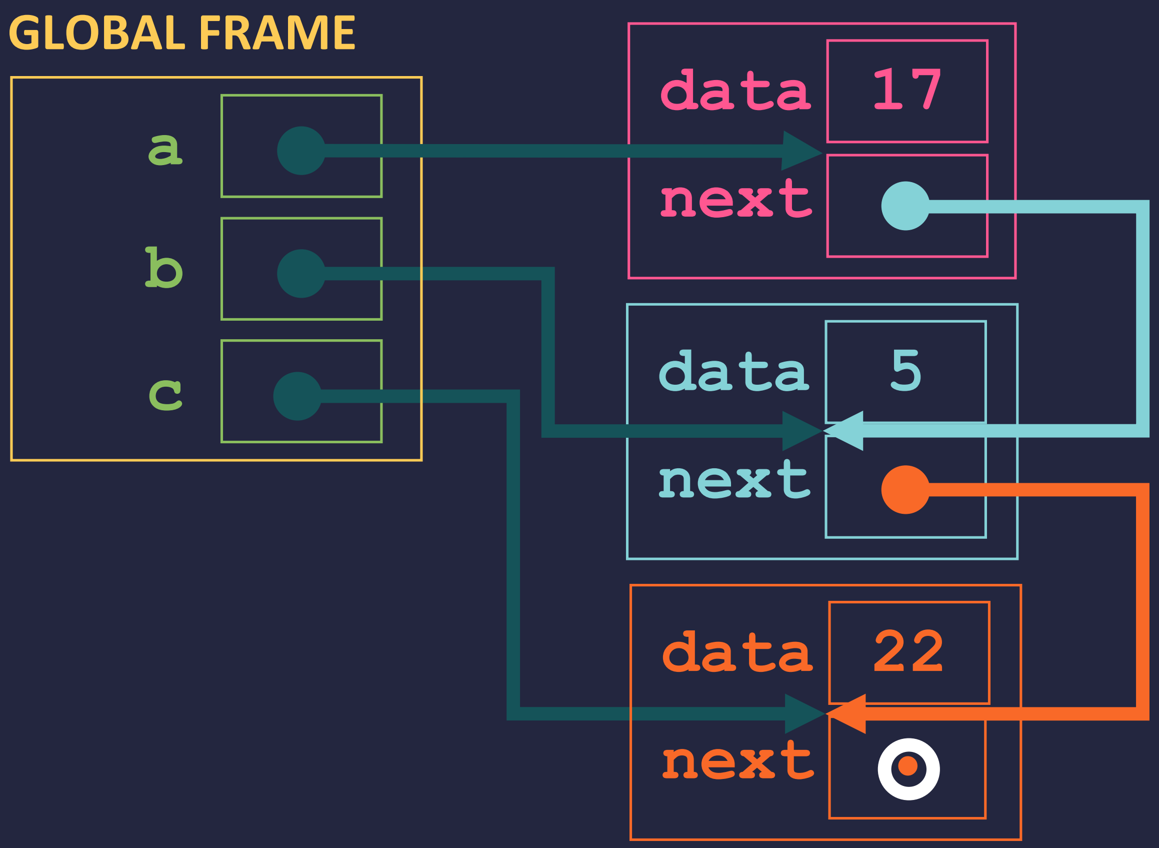
TRAVERSING A LINKED LIST

```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None

def traverse(frst):
    curr = frst
    while curr is not None:
        print(curr.value)
        curr = curr.next

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> traverse(a)
17
5
22
```



TRAVERSING A LINKED LIST

```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None

def traverse(frst):
    curr = frst
    while curr is not None:
        print(curr.value)
        curr = curr.next

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> traverse(a)
17
5
22
>>>
```



LINKED LISTS

LINKED LISTS

- Linked lists are a way of keeping a collection of items as a sequence.
- They are often the underlying structure for many organized data sets.

Generally:

Linked lists are an example of a ***link-based data structure***.

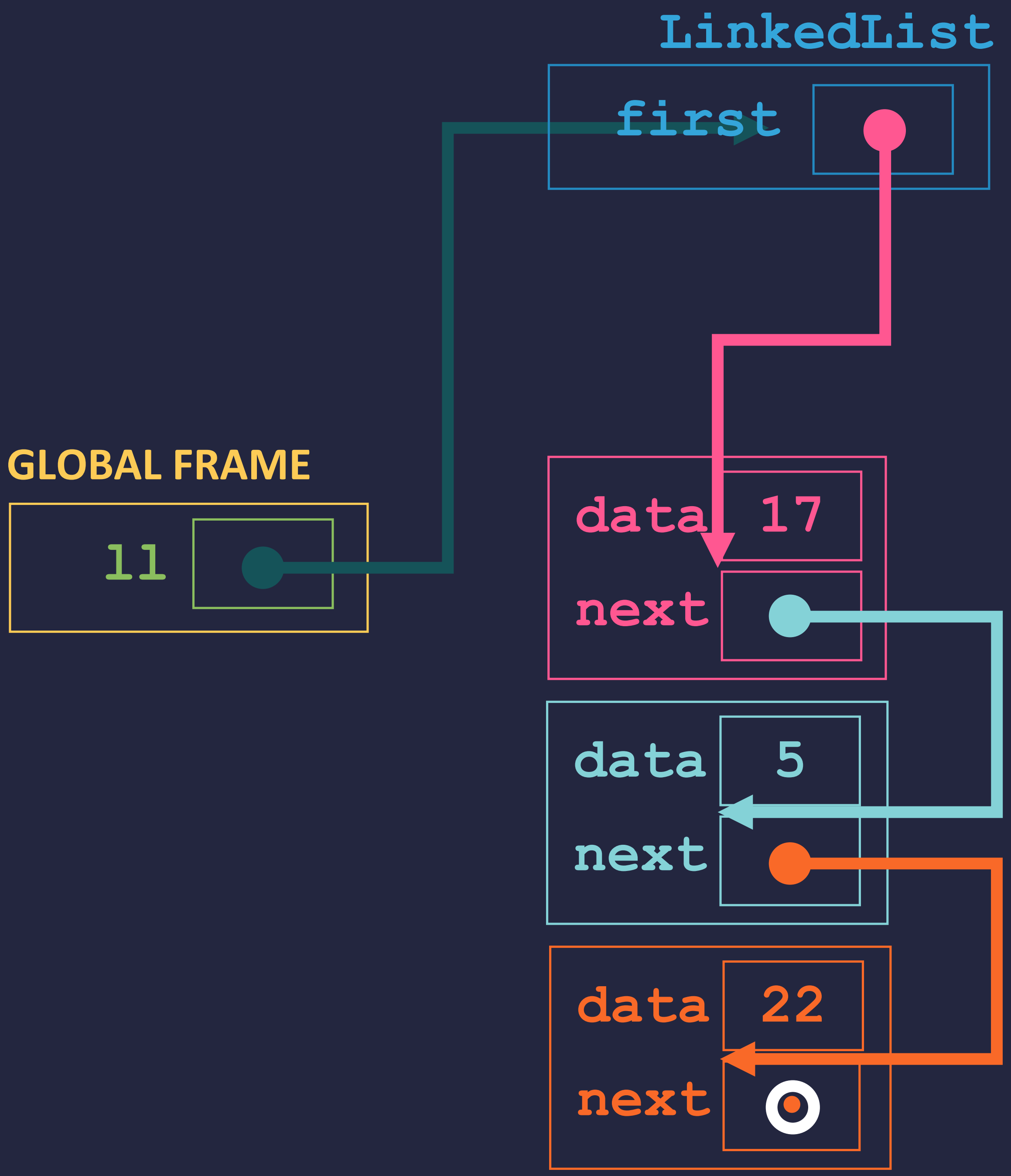
- Other examples are search trees, expression trees, graphs, ...
- The relationships amongst items can be edited by just relinking nodes.
- Items can be inserted anywhere with a few link changes.

A LINKED LIST CLASS

- On the remaining slides, we develop a linked list class.
- **Operations:**
  - Adding an item to the front.
  - Adding an item to the end.
  - Checking for inclusion of an item.
  - Displaying all the items.
  - Removing an item.
- Most of the operations rely on *list traversal* of some sort.

A LINKED LIST CLASS

```
class LLNode:
    def __init__(self, value):
        self.value = value
        self.next  = None

class LinkedList:
    def __init__(self):
        self.first = None

    def prepend(self, value):
        newNode = LLNode(value)
        newNode.next = self.first
        self.first = newNode
```

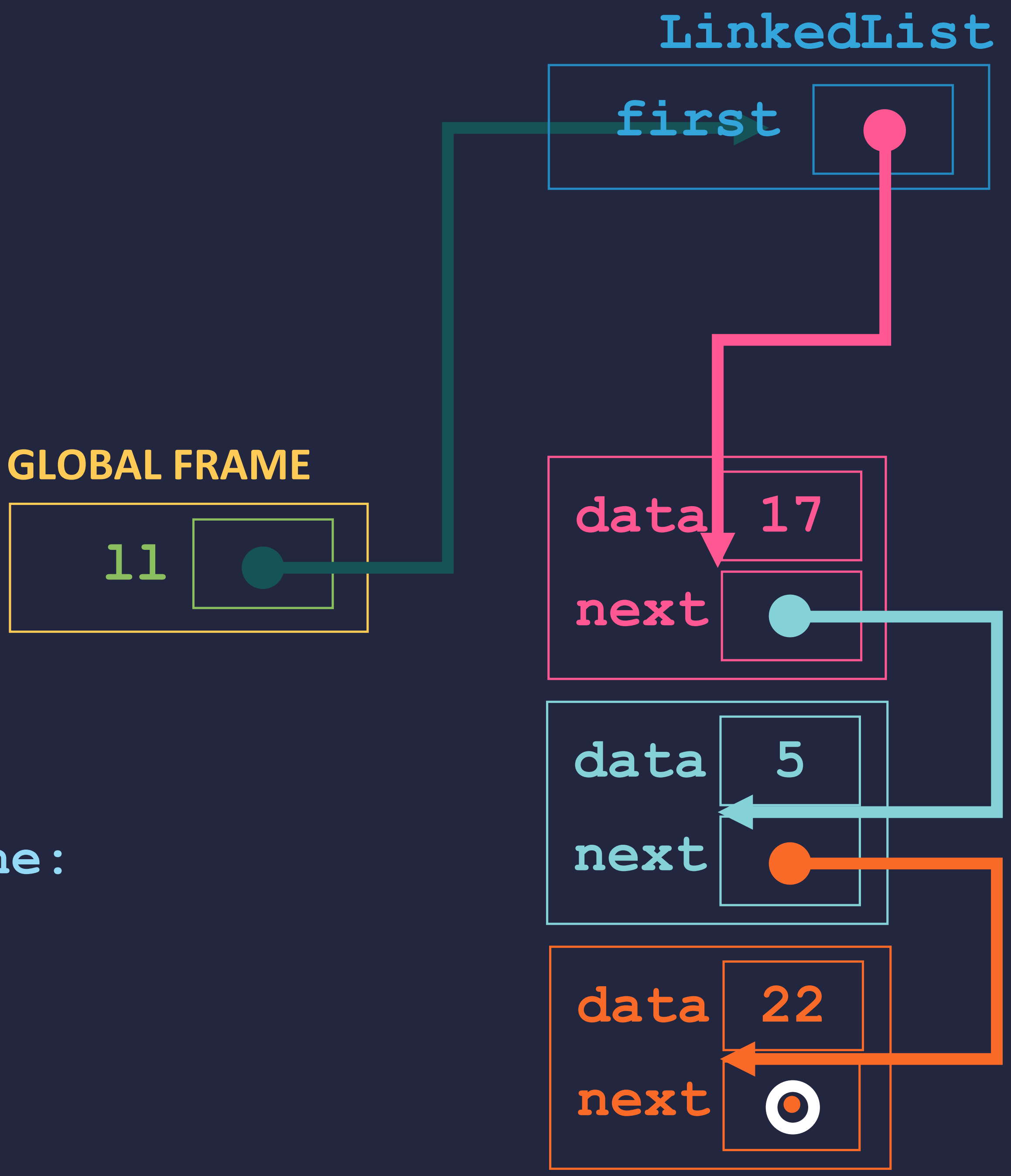


```
>>> ll = new LinkedList()
>>> ll.prepend(22)
>>> ll.prepend(5)
>>> ll.prepend(17)
```

LINKED LIST APPEND

A LINKED LIST CLASS

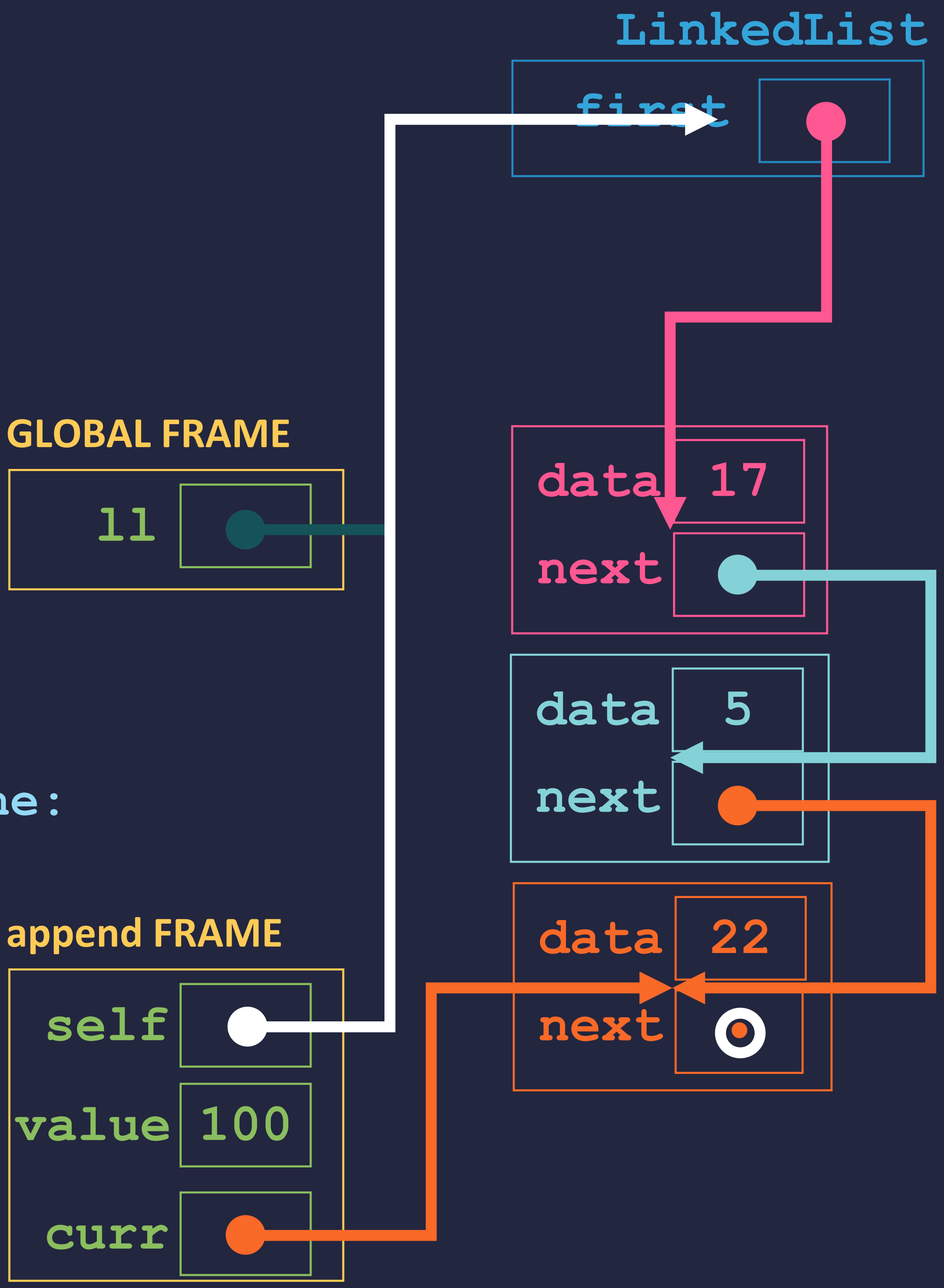
```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None

class LinkedList:
    ...
    def append(self, value):
        if self.first is None:
            self.first = LLNode(value)
        else:
            curr = self.first
            while curr.next is not None:
                curr = curr.next
            curr.next = LLNode(value)

>>> ll = new LinkedList()
>>> ll.append(17)
>>> ll.append(5)
>>> ll.append(22)
>>>
```



A LINKED LIST CLASS

```
class LLNode:
    def __init__(self, value):
        self.value = value
        self.next  = None

class LinkedList:
    ...
    def append(self, value):
        if self.first is None:
            self.first = LLNode(value)
        else:
            curr = self.first
            while curr.next is not None:
                curr = curr.next
            curr.next = LLNode(value)
```

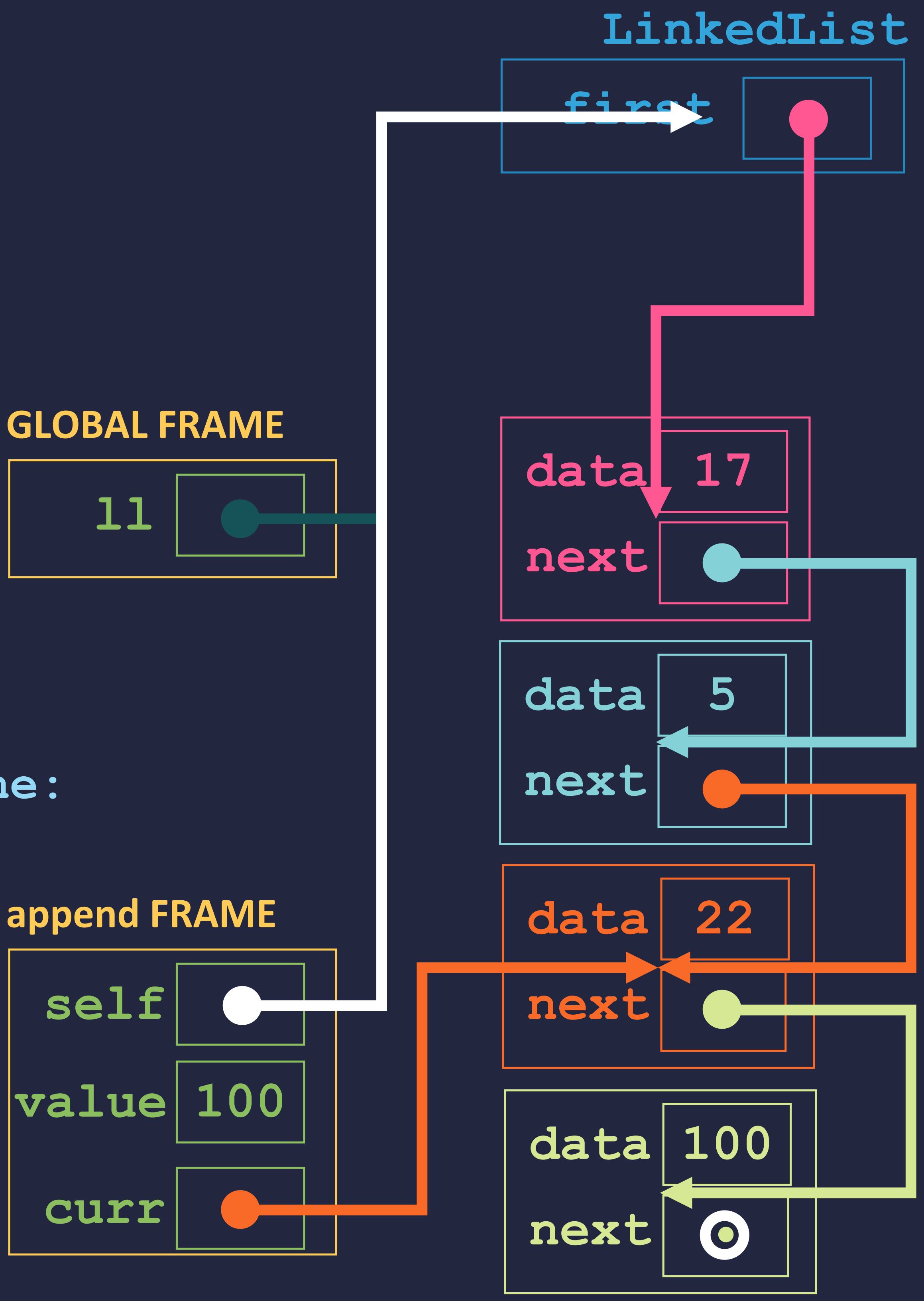
```
>>> ll = new LinkedList()
>>> ll.append(17)
>>> ll.append(5)
>>> ll.append(22)
>>> ll.append(100)
```



A LINKED LIST CLASS

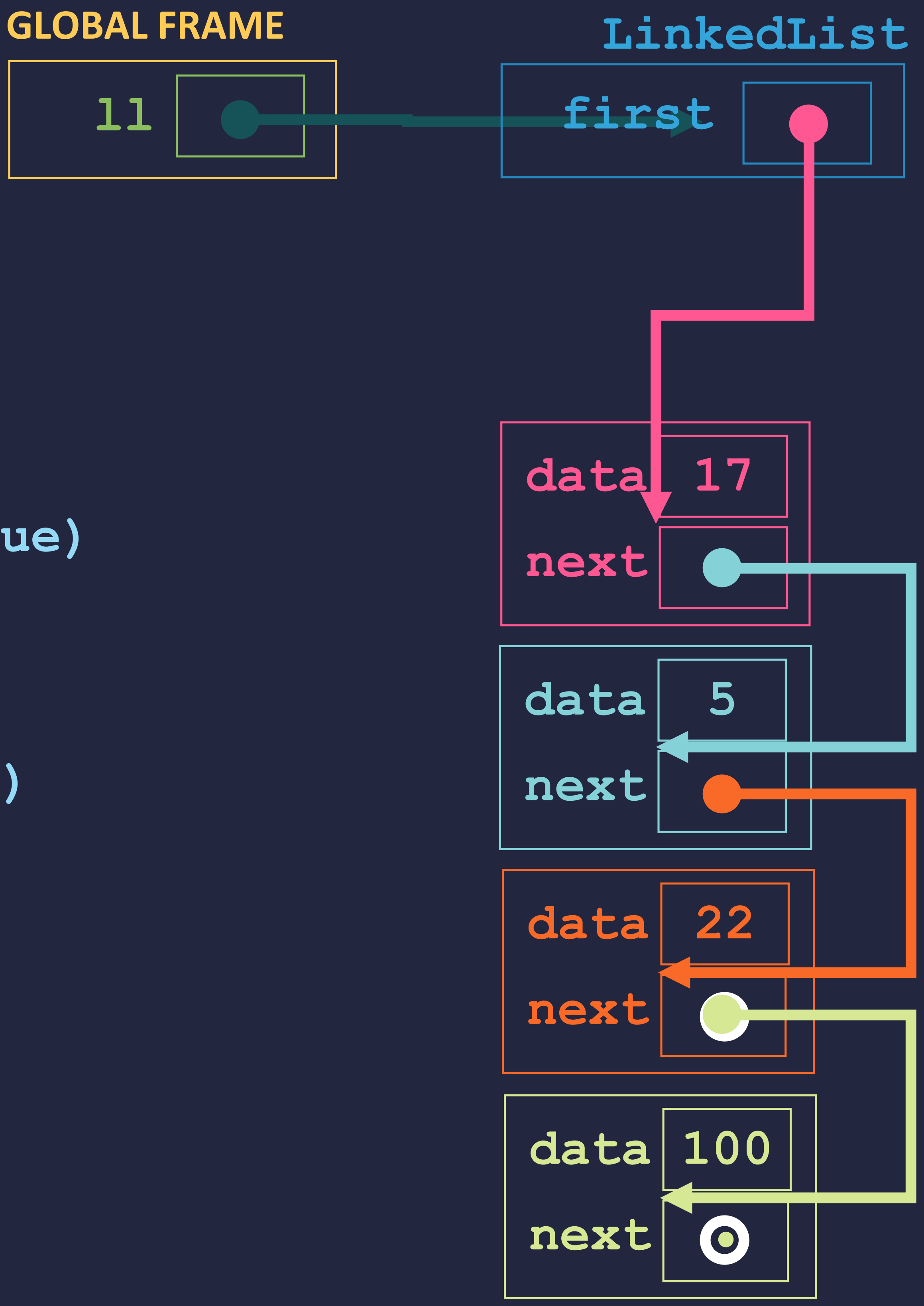
```
class LLNode:
    def __init__(self, value):
        self.value = value
        self.next  = None

class LinkedList:
    ...
    def append(self, value):
        if self.first is None:
            self.first = LLNode(value)
        else:
            curr = self.first
            while curr.next is not None:
                curr = curr.next
            curr.next = LLNode(value)
```

```
>>> ll = new LinkedList()
>>> ll.append(17)
>>> ll.append(5)
>>> ll.append(22)
>>> ll.append(100)
```



A LINKED LIST CLASS

```
class LinkedList:
    ...
    def asString(self):
        if self.first is None:
            return "<>"
        else:
            s = "<"
            s += str(self.first.value)
            curr = self.first.next
            while curr is not None:
                s += ", "
                s += str(curr.value)
                curr = curr.next
            s += ">"
            return s
```

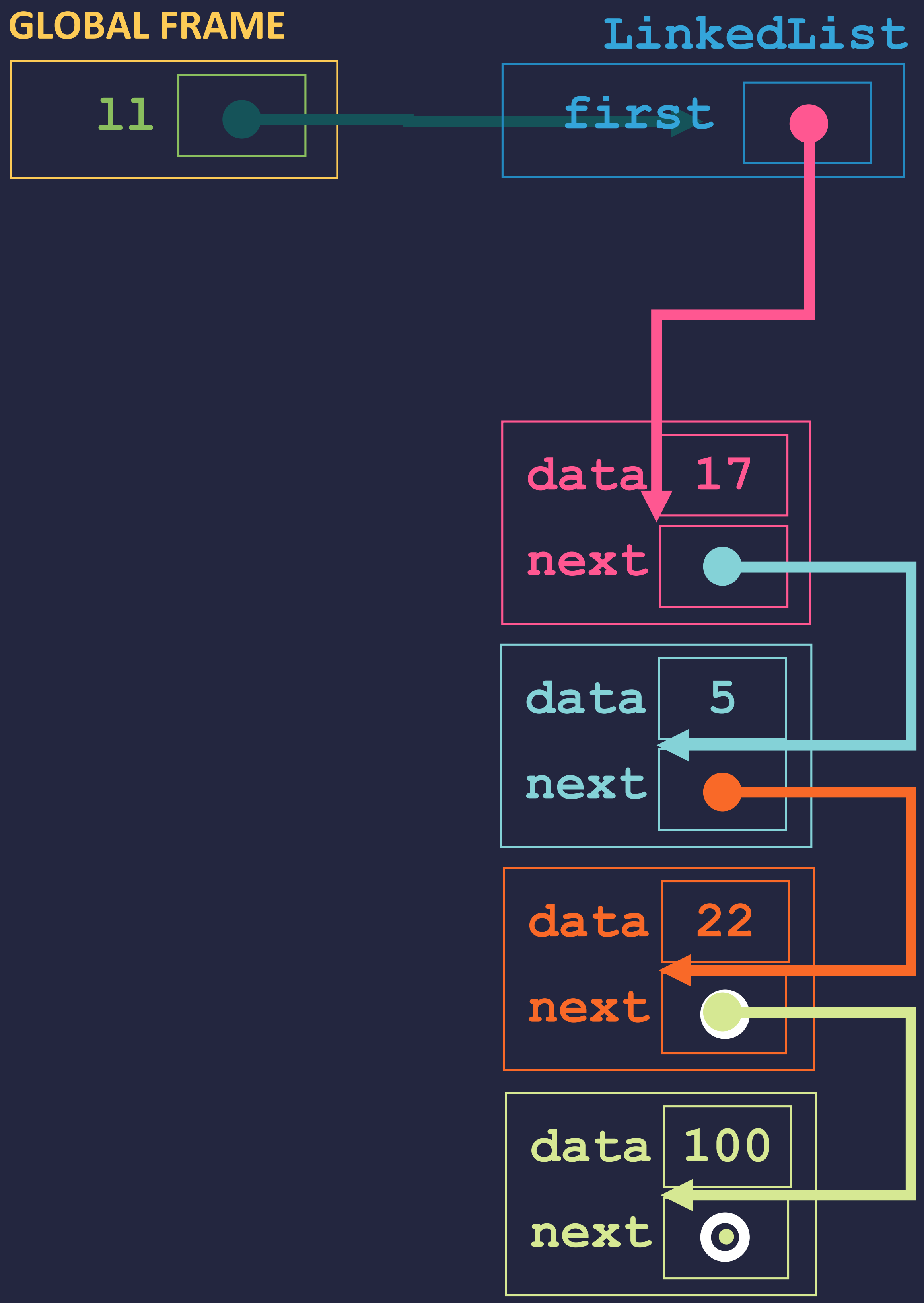


```
>>> ll.asString()
'<17, 5, 22, 100>'
>>>
```

LINKED LIST DELETION

A LINKED LIST CLASS

```
class LinkedList:
    ...
    def delete(self, value):
        prev = None
        curr = self.first
        if curr is None:
            return None
        while curr.value != value:
            prev = curr
            curr = curr.next
        if prev is None:
            self.first = curr.next
        else:
            prev.next = curr.next


>>> ll.delete(22)
```

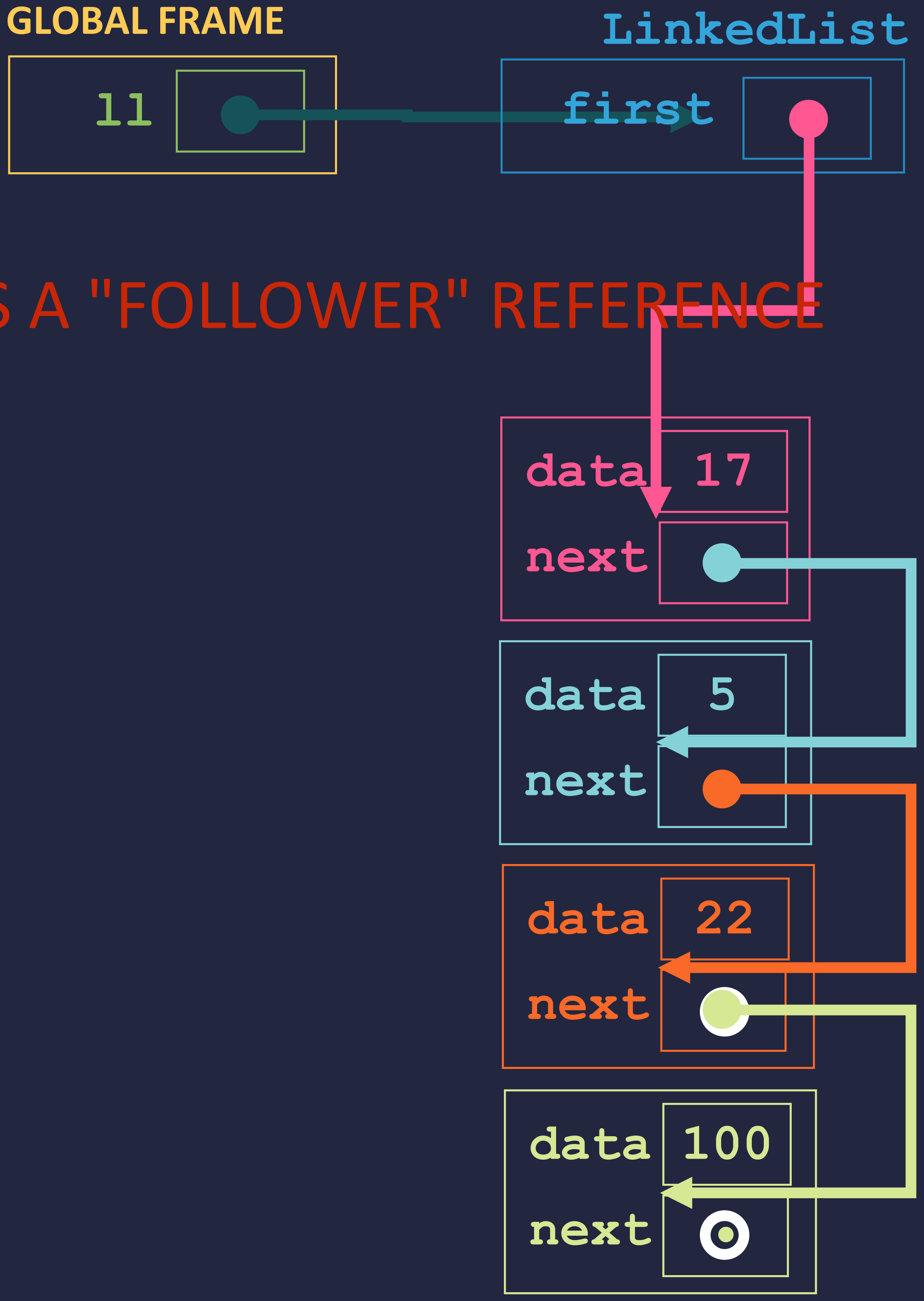


A LINKED LIST CLASS

```
class LinkedList:
    ...
    def delete(self, value):
        prev = None
        curr = self.first
        if curr is None:
            return None
        while curr.value != value:
            prev = curr
            curr = curr.next
        if prev is None:
            self.first = curr.next
        else:
            prev.next = curr.next
```

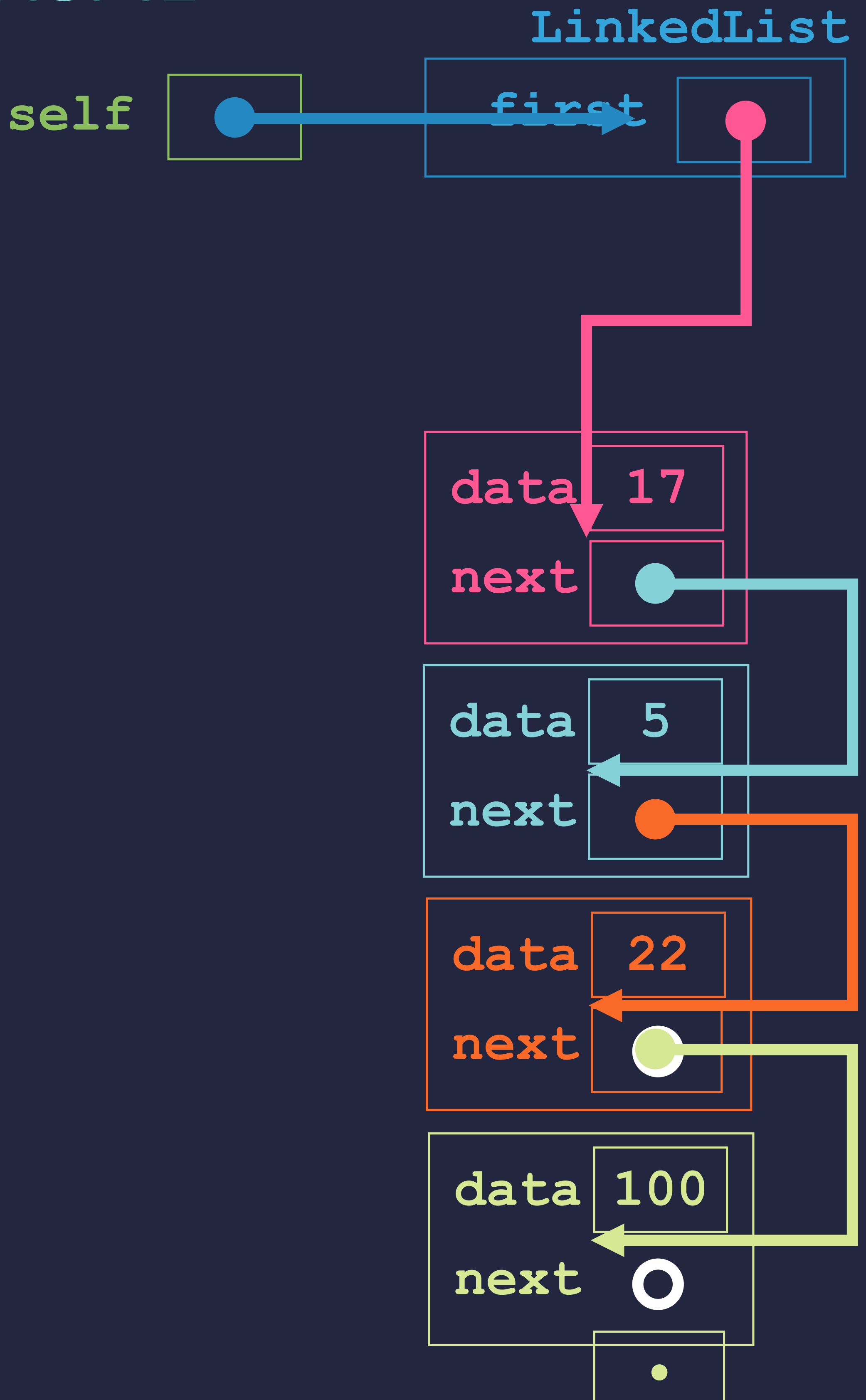
THIS USES A "FOLLOWER" REFERENCE

```
>>> ll.delete(22)
```



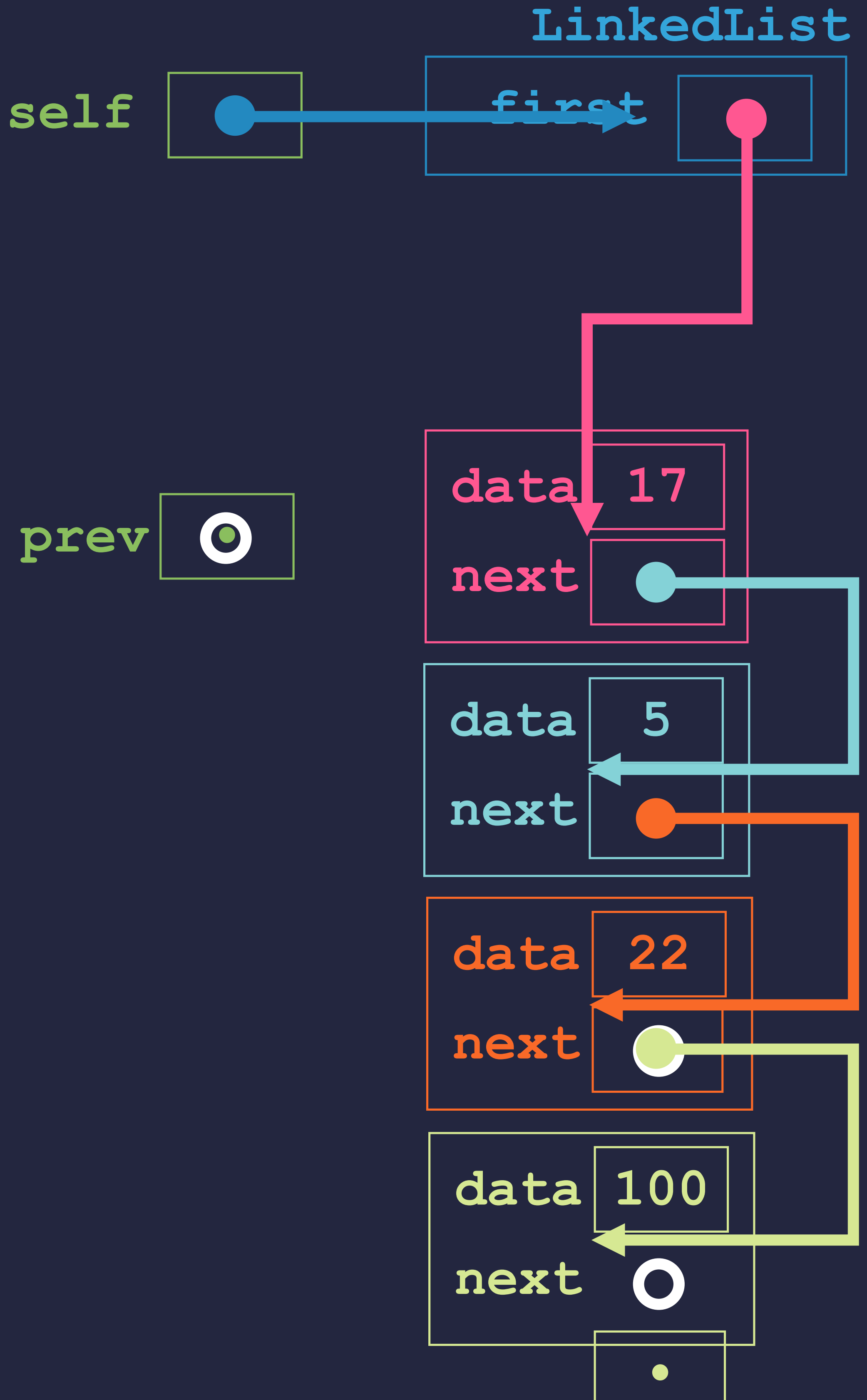
FOLLOWER POINTER TRAVERSAL

```
prev = None
curr = self.first
if curr is None:
    return None
while curr.value != value:
    prev = curr
    curr = curr.next
```



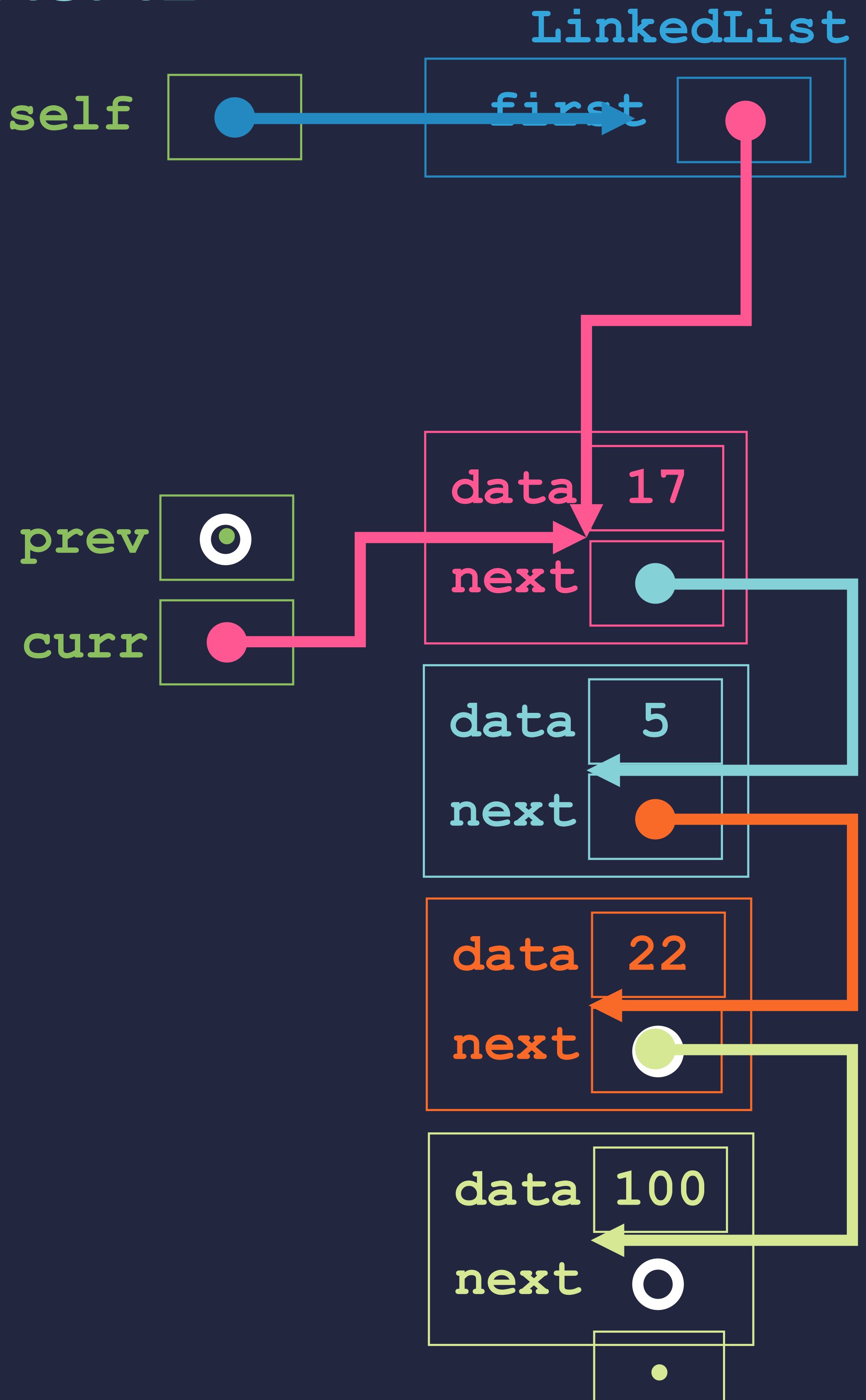
FOLLOWER POINTER TRAVERSAL

```
prev = None
curr = self.first
if curr is None:
    return None
while curr.value != value:
    prev = curr
    curr = curr.next
```



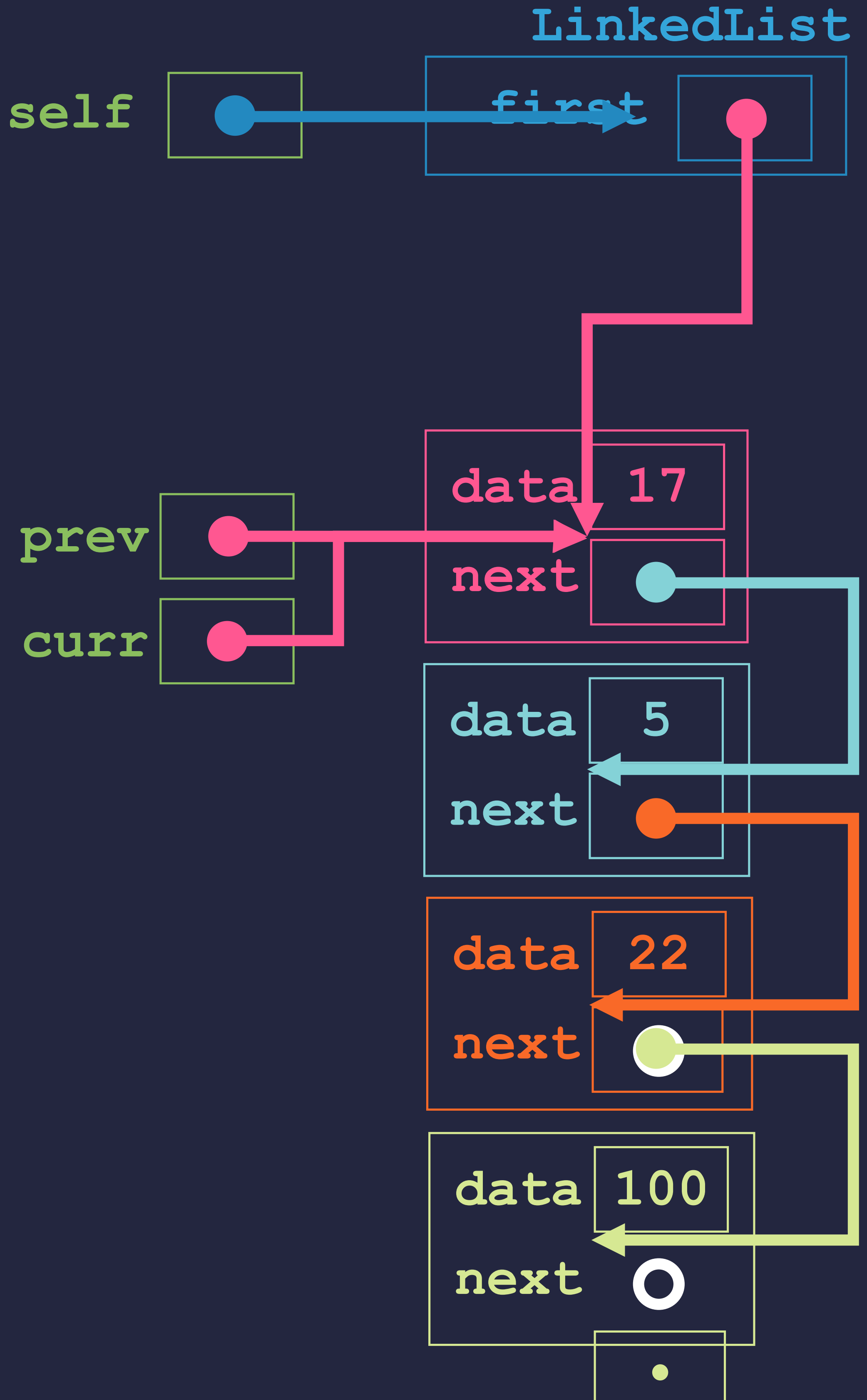
FOLLOWER POINTER TRAVERSAL

```
prev = None
curr = self.first
if curr is None:
    return None
while curr.value != value:
    prev = curr
    curr = curr.next
```



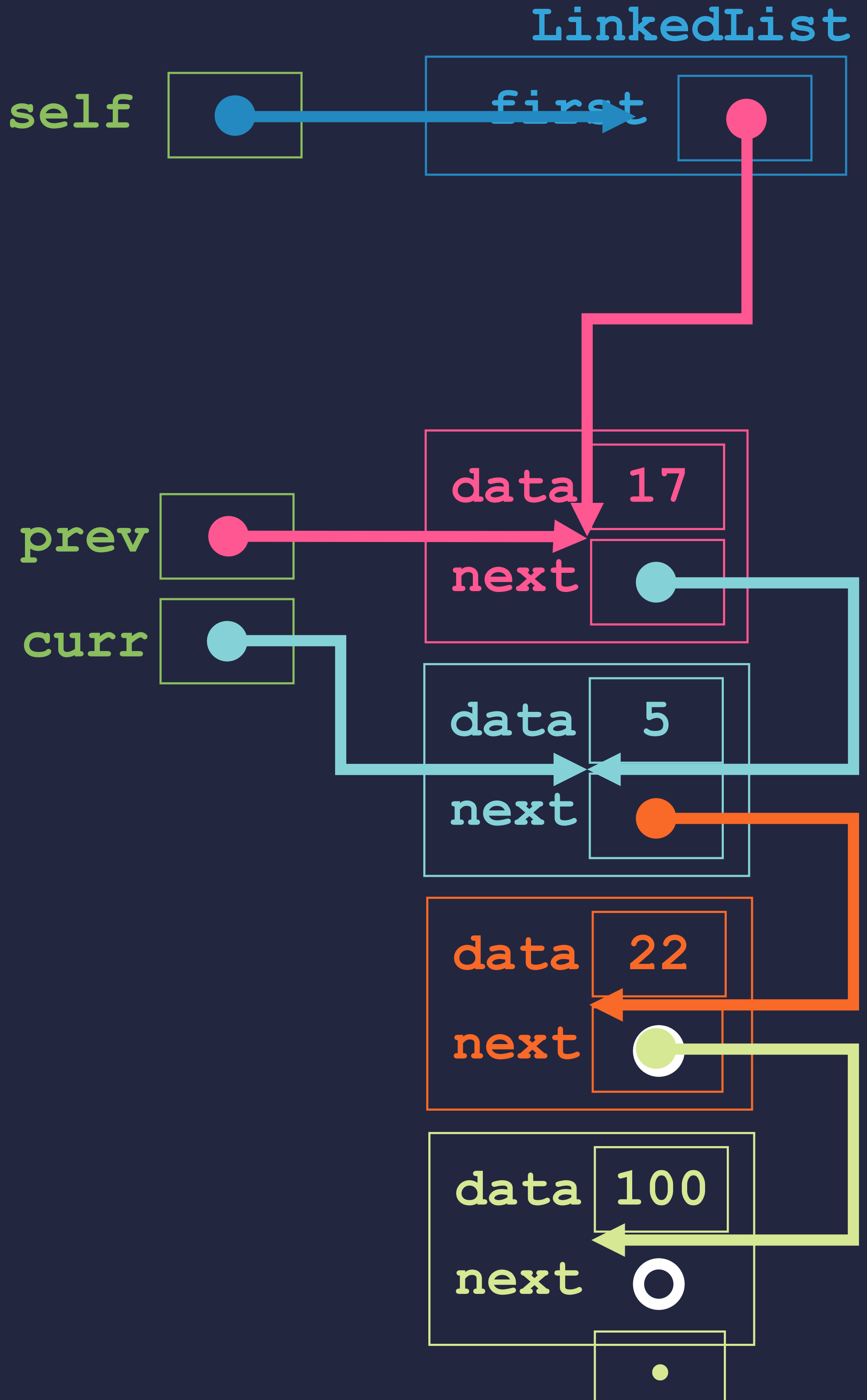
FOLLOWER POINTER TRAVERSAL

```
prev = None
curr = self.first
if curr is None:
    return None
while curr.value != value:
    prev = curr
    curr = curr.next
```



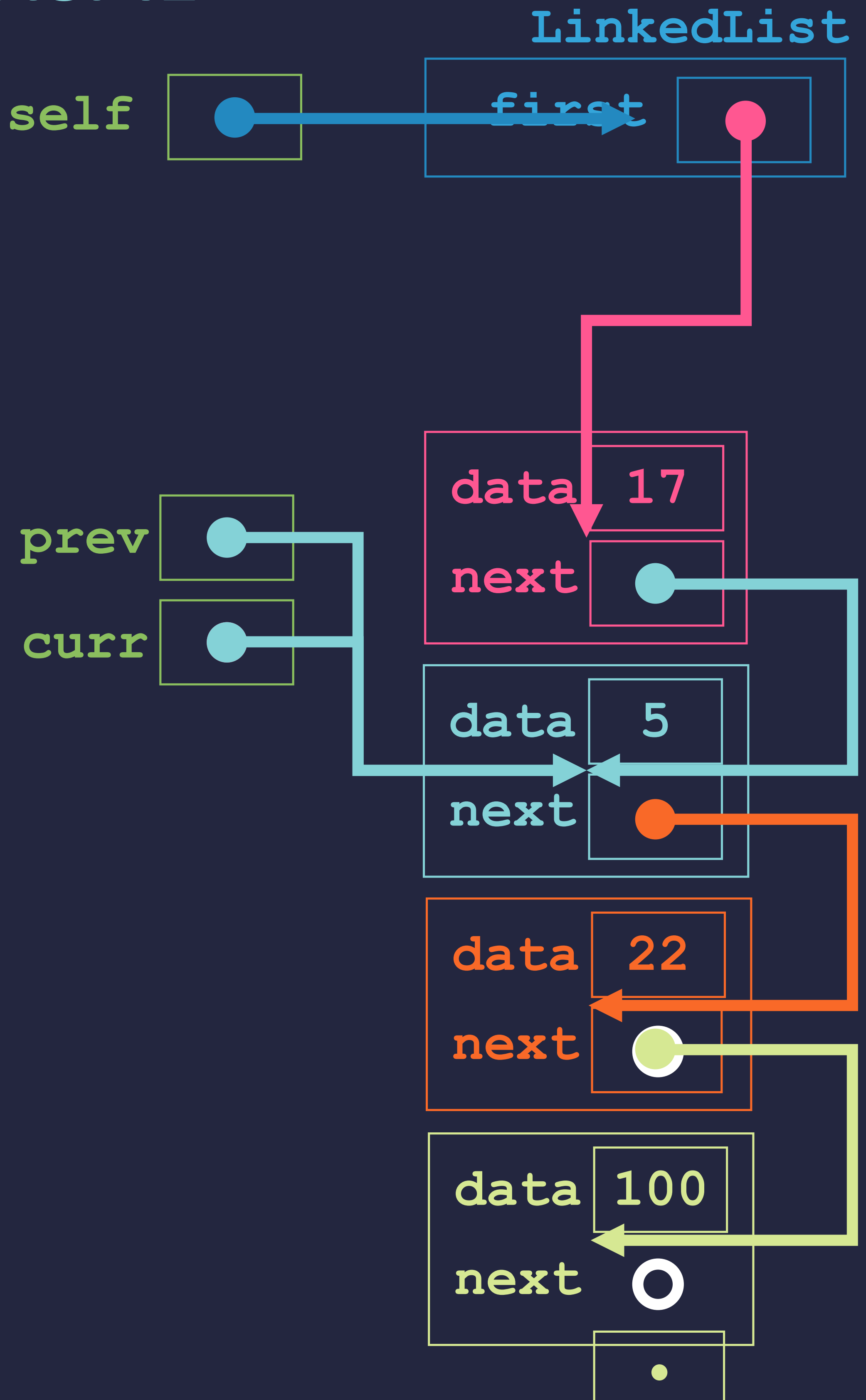
FOLLOWER POINTER TRAVERSAL

```
prev = None
curr = self.first
if curr is None:
    return None
while curr.value != value:
    prev = curr
    curr = curr.next
```



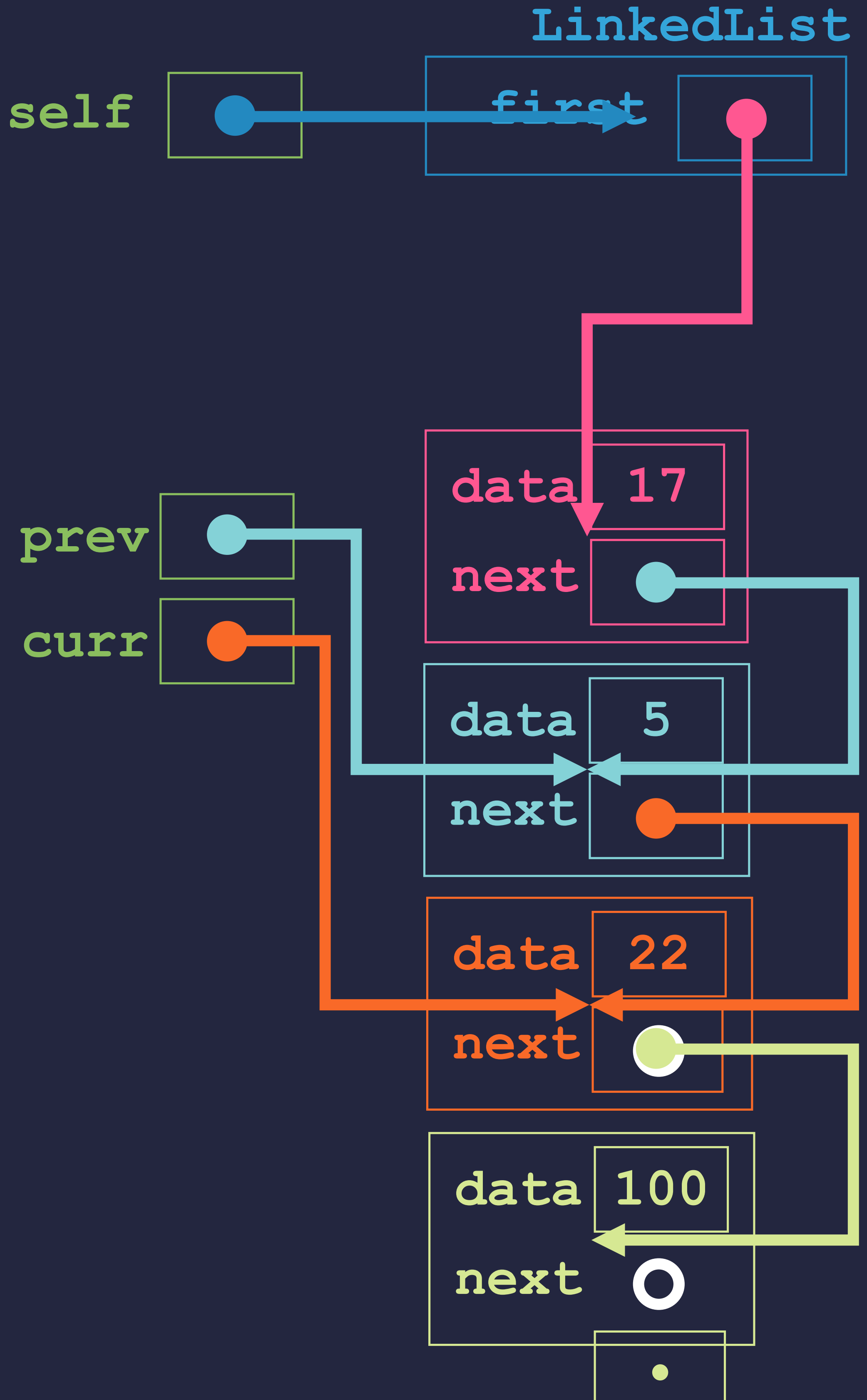
FOLLOWER POINTER TRAVERSAL

```
prev = None
curr = self.first
if curr is None:
    return None
while curr.value != value:
    prev = curr
    curr = curr.next
```



FOLLOWER POINTER TRAVERSAL

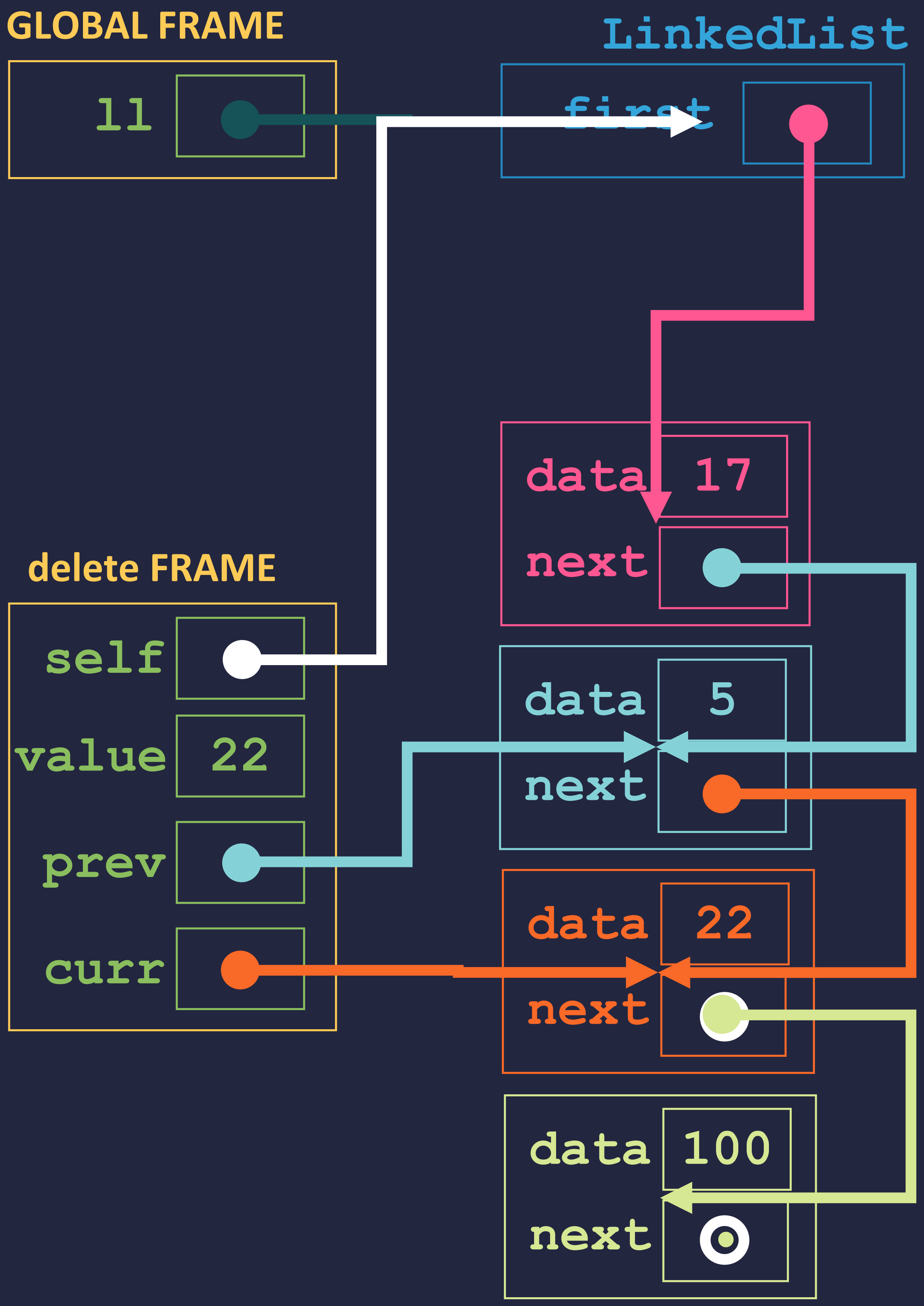
```
prev = None
curr = self.first
if curr is None:
    return None
while curr.value != value:
    prev = curr
    curr = curr.next
```



A LINKED LIST CLASS

```
...
def delete(self, value):
    prev = None
    curr = self.first
    if curr is None:
        return None
    while curr.value != value:
        prev = curr
        curr = curr.next
    if prev is None:
        self.first = curr.next
    else:
        prev.next = curr.next
```

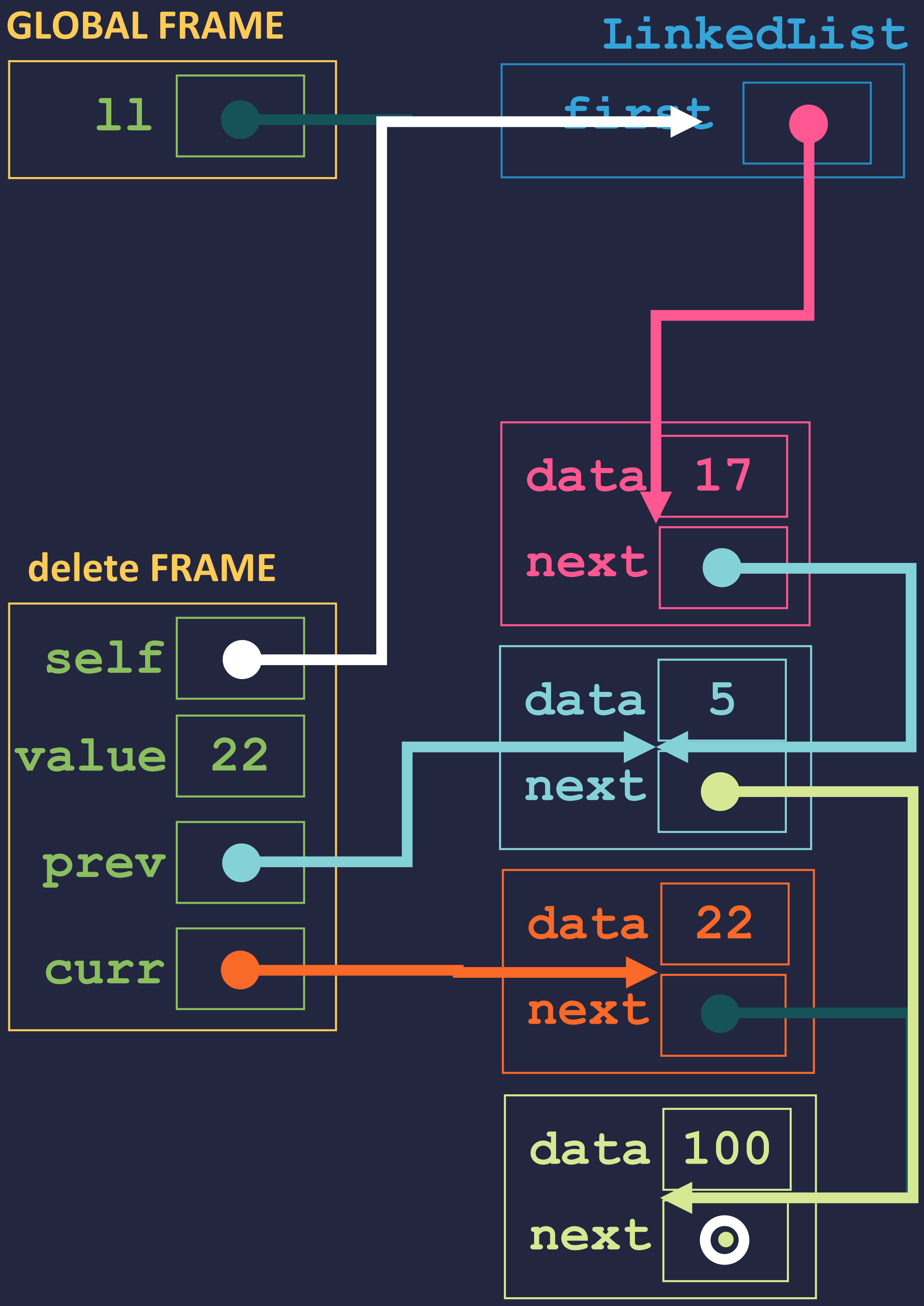
```
>>> ll.delete(22)
```



A LINKED LIST CLASS

```
...
def delete(self, value):
    prev = None
    curr = self.first
    if curr is None:
        return None
    while curr.value != value:
        prev = curr
        curr = curr.next
    if prev is None:
        self.first = curr.next
    else:
        prev.next = curr.next
```

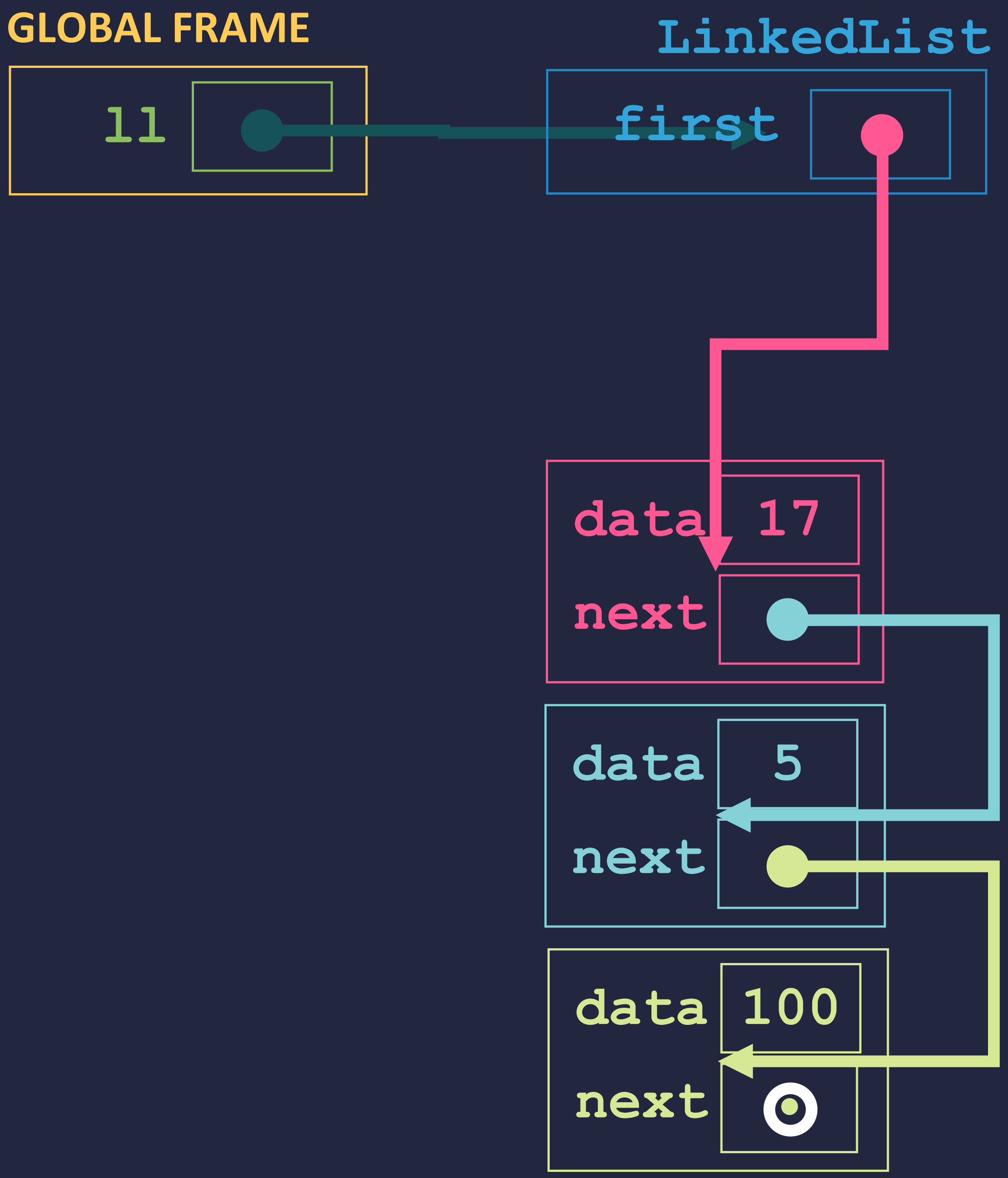
```
>>> ll.delete(22)
```



A LINKED LIST CLASS

```
class LinkedList:
    ...
    def delete(self, value):
        prev = None
        curr = self.first
        if curr is None:
            return None
        while curr.value != value:
            prev = curr
            curr = curr.next
        if prev is None:
            self.first = curr.next
        else:
            prev.next = curr.next
```

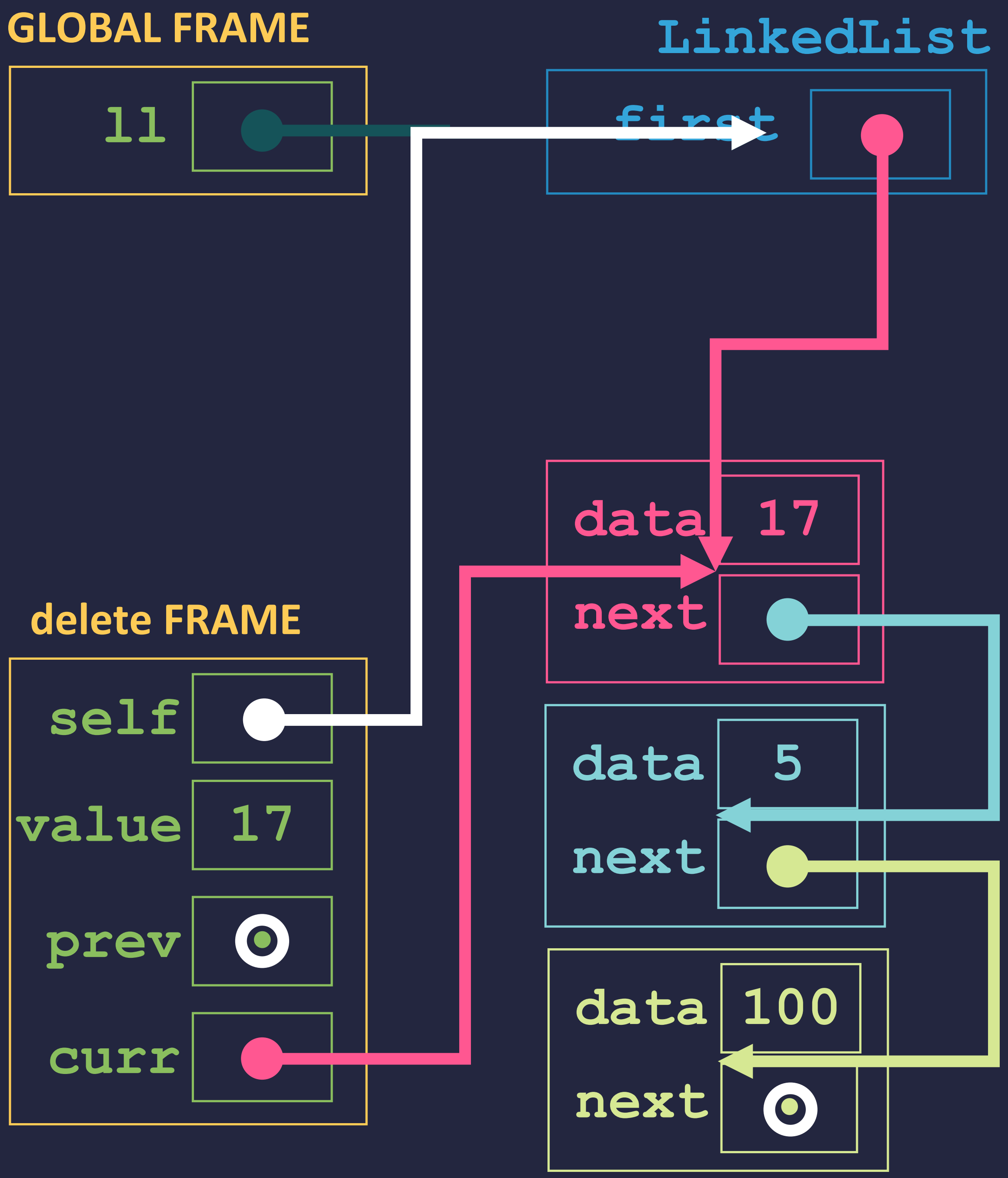
```
>>> ll.delete(22)
>>> ll.delete(17)
```



A LINKED LIST CLASS

```
class LinkedList:
    ...
    def delete(self, value):
        prev = None
        curr = self.first
        if curr is None:
            return None
        while curr.value != value:
            prev = curr
            curr = curr.next
        if prev is None:
            self.first = curr.next
        else:
            prev.next = curr.next
```

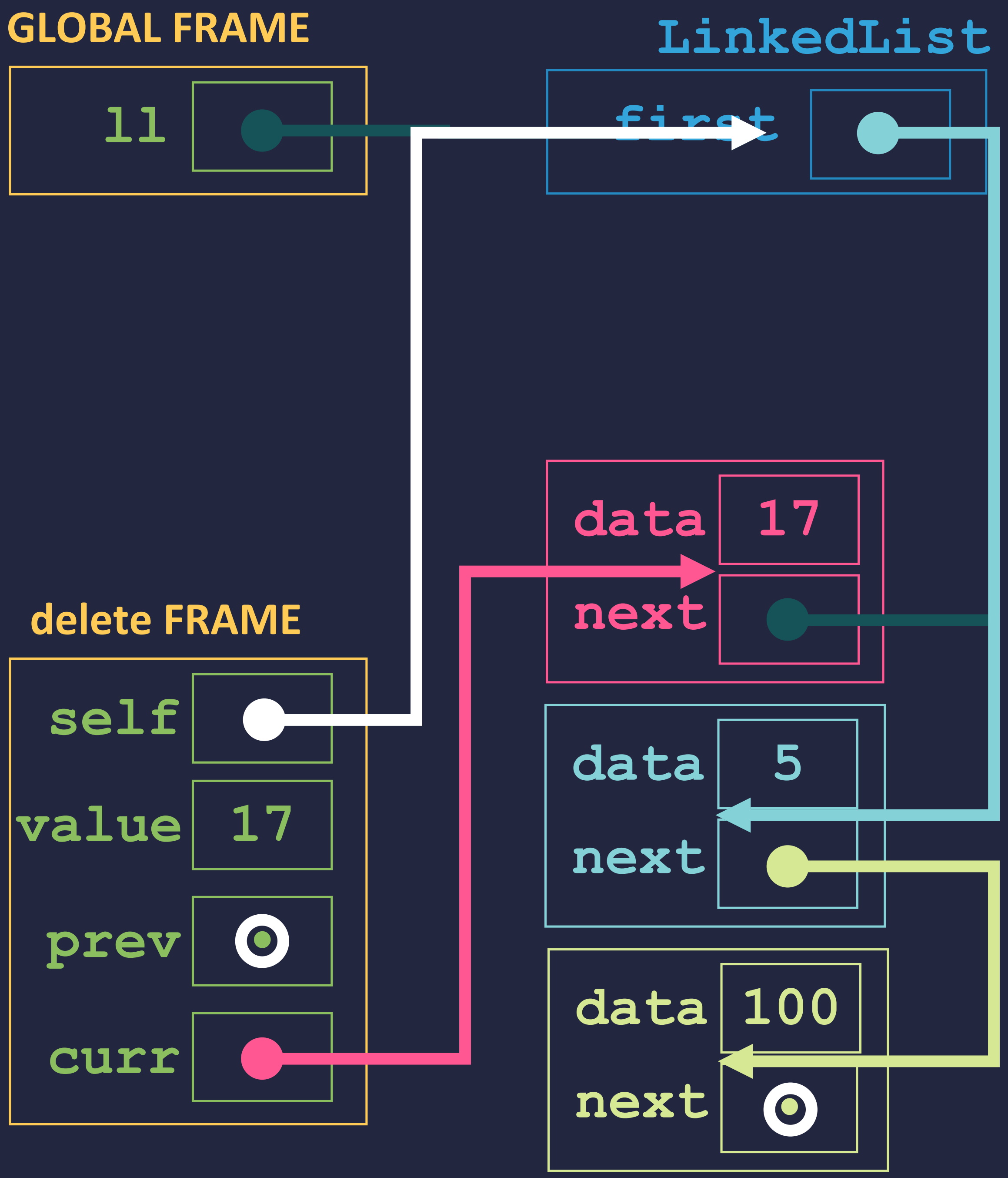
```
>>> ll.delete(22)
>>> ll.delete(17)
```



A LINKED LIST CLASS

```
class LinkedList:
    ...
    def delete(self, value):
        prev = None
        curr = self.first
        if curr is None:
            return None
        while curr.value != value:
            prev = curr
            curr = curr.next
        if prev is None:
            self.first = curr.next
        else:
            prev.next = curr.next
```

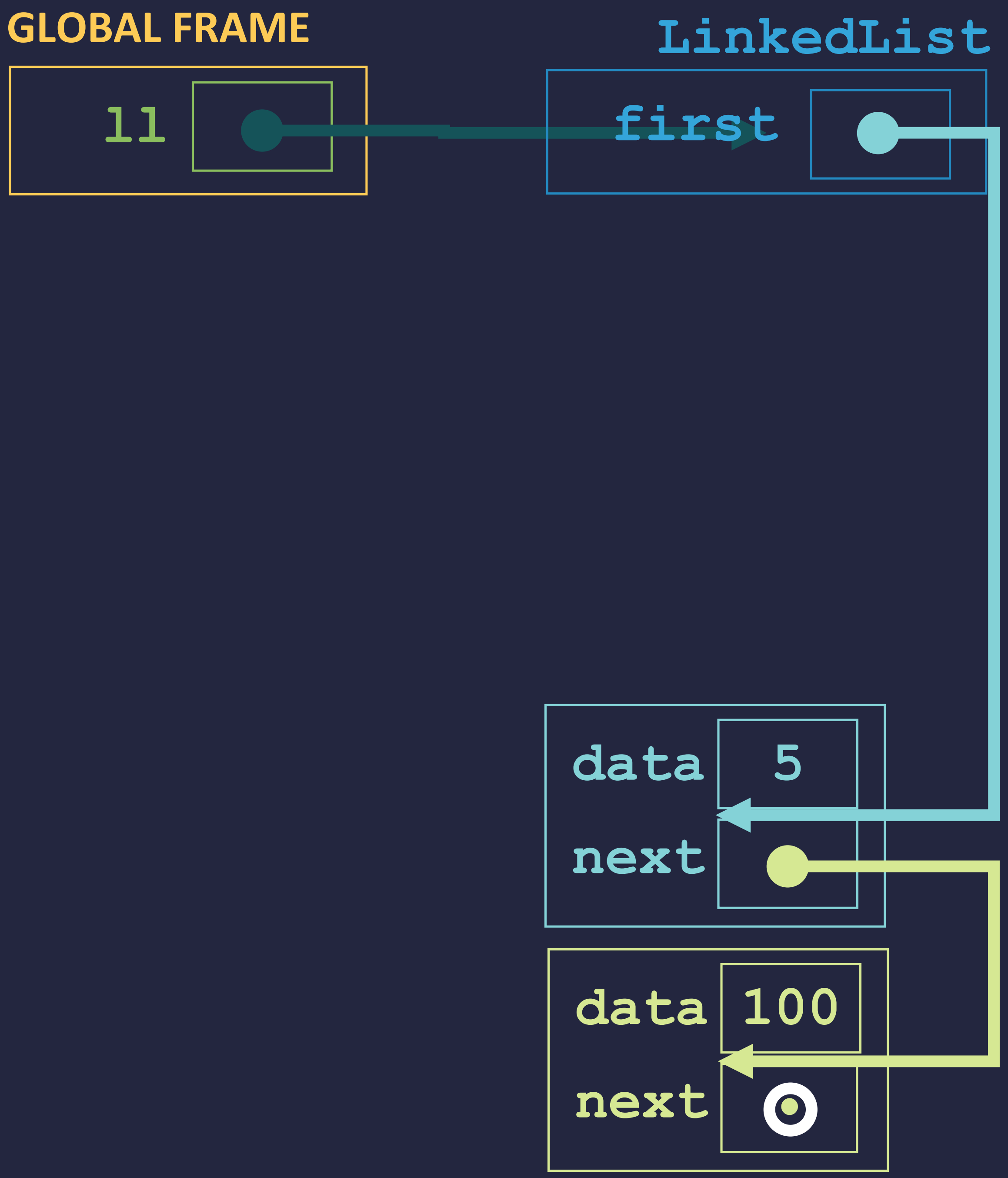
```
>>> ll.delete(22)
>>> ll.delete(17)
```



A LINKED LIST CLASS

```
class LinkedList:
    ...
    def delete(self, value):
        prev = None
        curr = self.first
        if curr is None:
            return None
        while curr.value != value:
            prev = curr
            curr = curr.next
        if prev is None:
            self.first = curr.next
        else:
            prev.next = curr.next
```

```
>>> ll.delete(22)
>>> ll.delete(17)
>>>
```



GLOBAL FRAME
ll
LinkedList
first
data
5
next
data
100
next