LINKED LISTS

LECTURE 9–1

JIM FIX, REED COLLEGE CSCI 121

COURSE INFO

- **Due tomorrow:**
  - **Homework 7**: Dispatcher exercise
- **To be assigned:**
  - **Homework 8** on higher order functions (2 exercises)
  - **Homework 9** on linked lists (**TODAY'S LECTURE**; 4 exercises)
- *Office Hours cancelled tonight* because of faculty candidate talk.
- **Next Monday:**
  - **Quiz 4** on recursion
- **Posted:**
  - **Project 3: hawk dove**, a bird simulation, *due in two weeks.*
  - **Project 4: adventure**, a text-based game, *proposal due one week from Friday.*
- **Today's lecture:**
  - We look at *linked lists*.

LIST DEMO

```
>>> xs = list(range(10000000))
>>> for _ in range(100000):
...     xs.append(1)
...
>>> for _ in range(100000):
...     xs.insert(0,0)
...
```

LIST DEMO

```
>>> xs = list(range(10000000))
>>> for _ in range(100000):
...     xs.append(1)
...
>>> for _ in range(100000):
...     xs.insert(0,0)
...
```

This runs somewhat instantaneously.

LIST DEMO

```
>>> xs = list(range(10000000))
>>> for _ in range(100000):
...     xs.append(1)
...
>>> for _ in range(100000):
...     xs.insert(0,0)
...
```

This loop sure takes a while. Why?

LIST DEMO

```
>>> xs = list(range(10000000))
>>> for _ in range(100000):
...     xs.append(1)
...
>>> for _ in range(100000):
...     xs.insert(0,0)
...
```

This loop sure takes a while. Why?

The items stored at 0 onward have to be copied right to make room for the new item #0.

LIST DEMO

```
>>> xs = list(range(10000000))
>>> for _ in range(100000):
...     xs.append(1)
...
>>> for _ in range(100000):
...     xs.insert(0,0)
...
>>> for _ in range(100000):
...     del xs[0]
...
```

This loop also takes a while. Why?

LIST DEMO

```
>>> xs = list(range(10000000))
>>> for _ in range(100000):
...     xs.append(1)
...
>>> for _ in range(100000):
...     xs.insert(0,0)
...
>>> for _ in range(100000):
...     del xs[0]
...
```

This loop also takes a while. Why?

The items stored at 1 onward have to be copied left to when we eliminate item #0.

LINKED DATA STRUCTURES

- Rather than storing data as a sequence in memory…
- …we can house the data into individual storage units called ***nodes.***
- The sequencing happens by having each node* know some successor node.
  - ➡ Each node refers to, or *links to*, some other node*.
- We can change the sequence order, or add things, or remove things, by changing the links.
- You don't have to move the data around.

** NOTE: except the last node in the sequence.*

A NODE CLASS

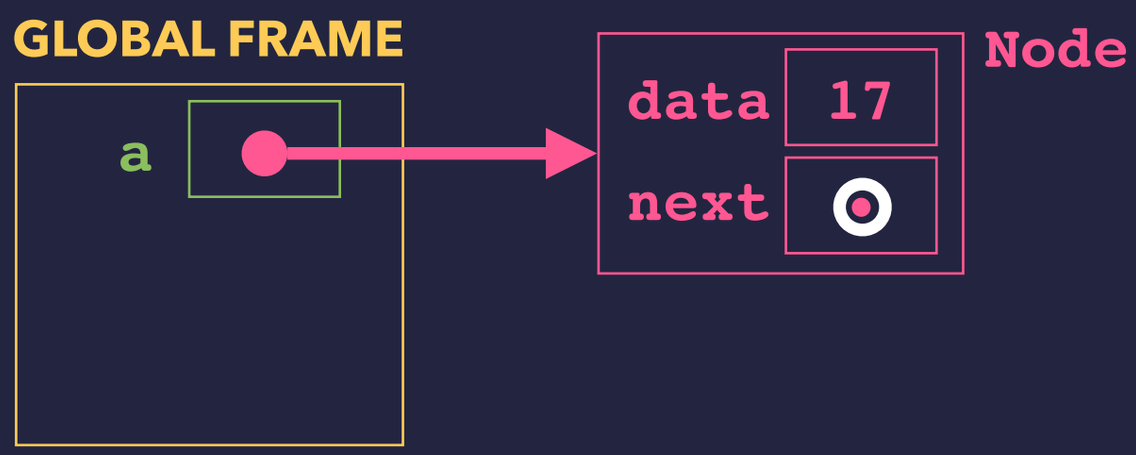
```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None
```

```
>>>
```

A NODE CLASS

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

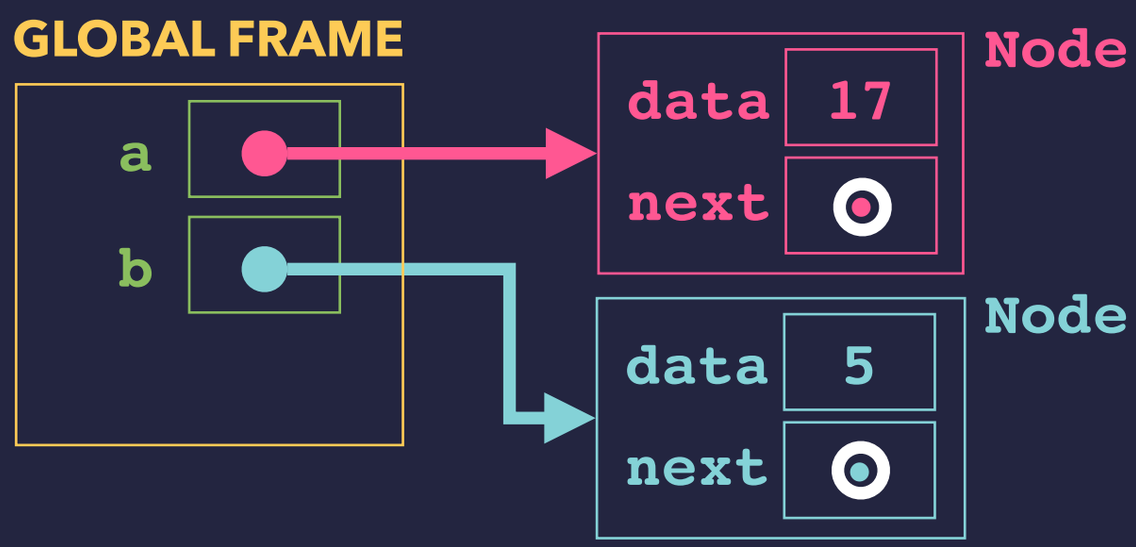
```
>>> a = Node(17)
>>>
```



A NODE CLASS

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

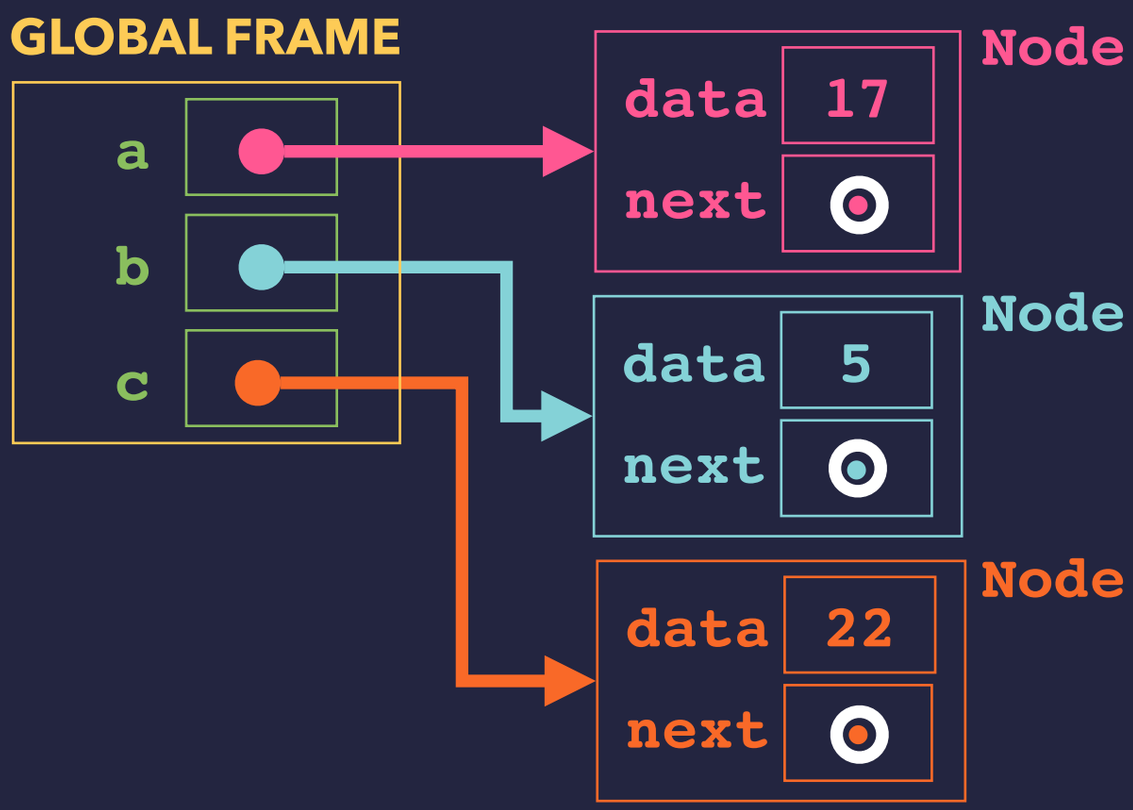
```
>>> a = Node(17)
>>> b = Node(5)
>>>
```



A NODE CLASS

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

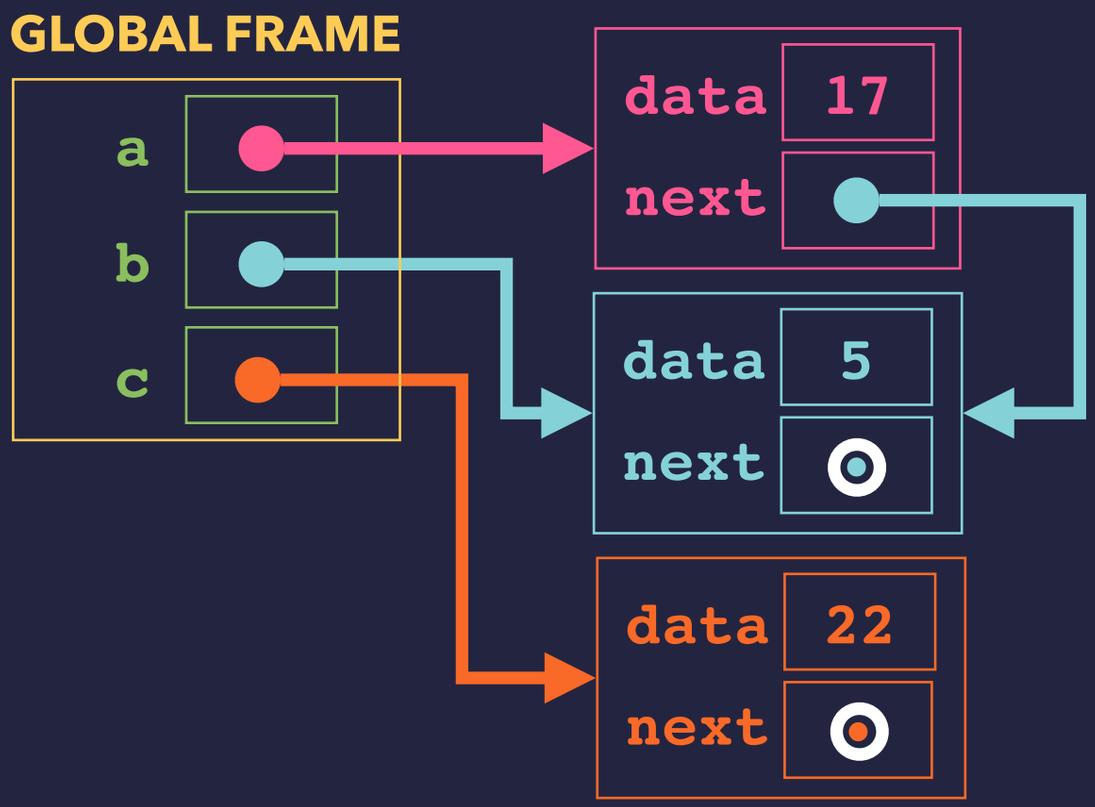
```
>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>>
```



LINKING NODES IN SERIES

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

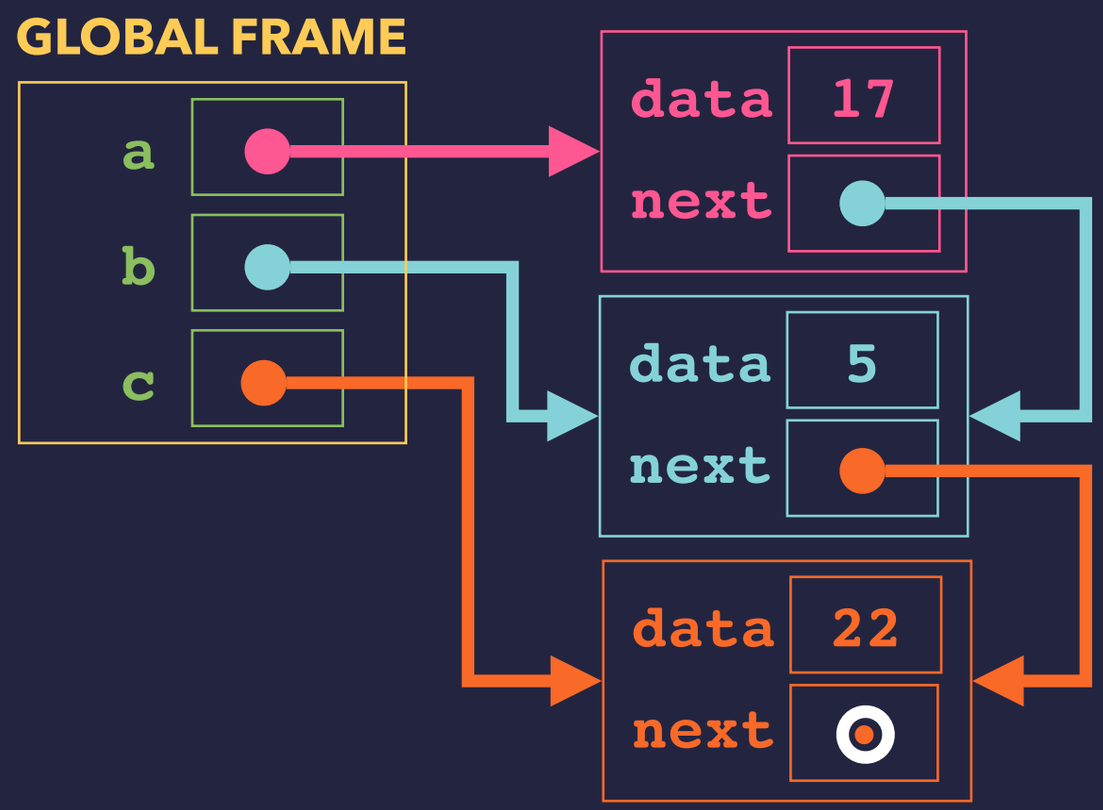
```
>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> a.next = b
>>>
```



LINKING NODES IN SERIES

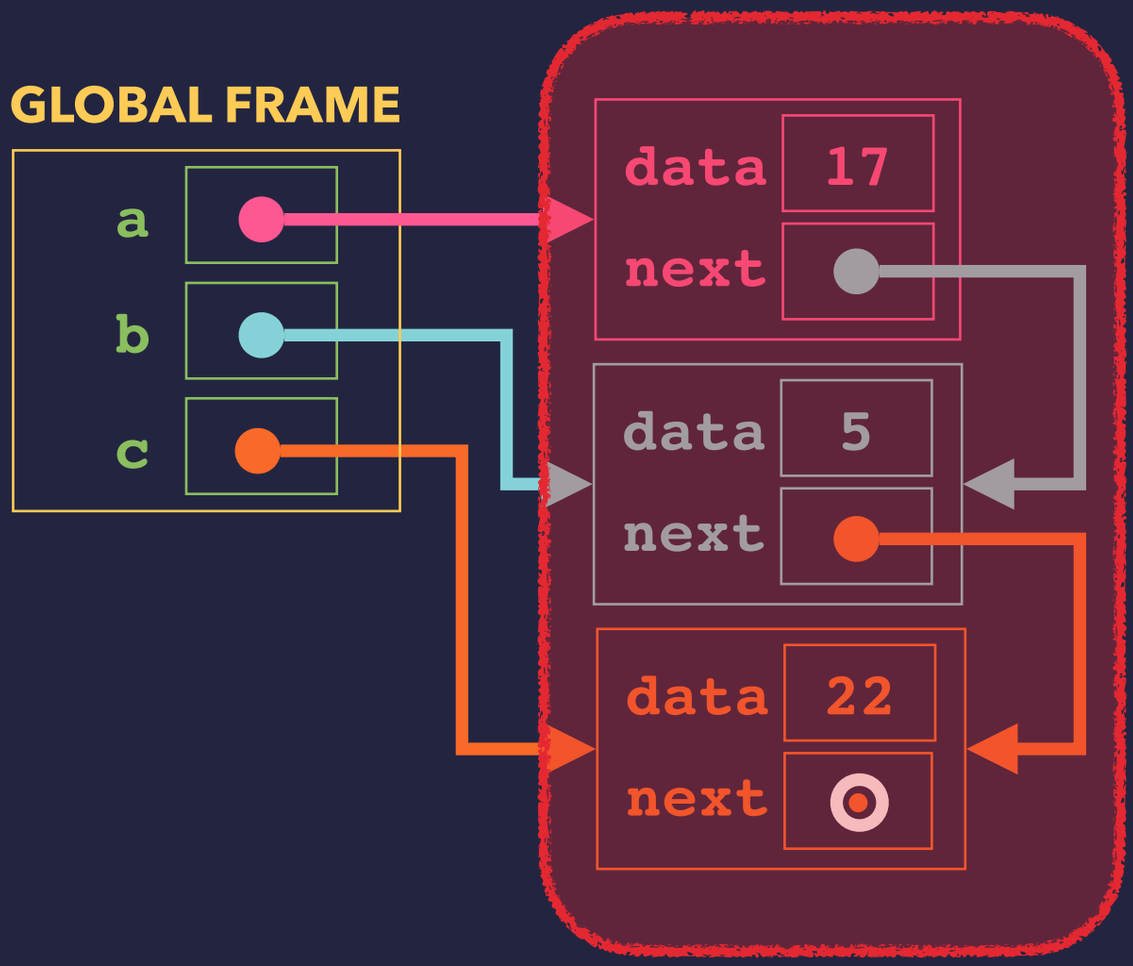
```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None
```

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



LINKING NODES IN SERIES

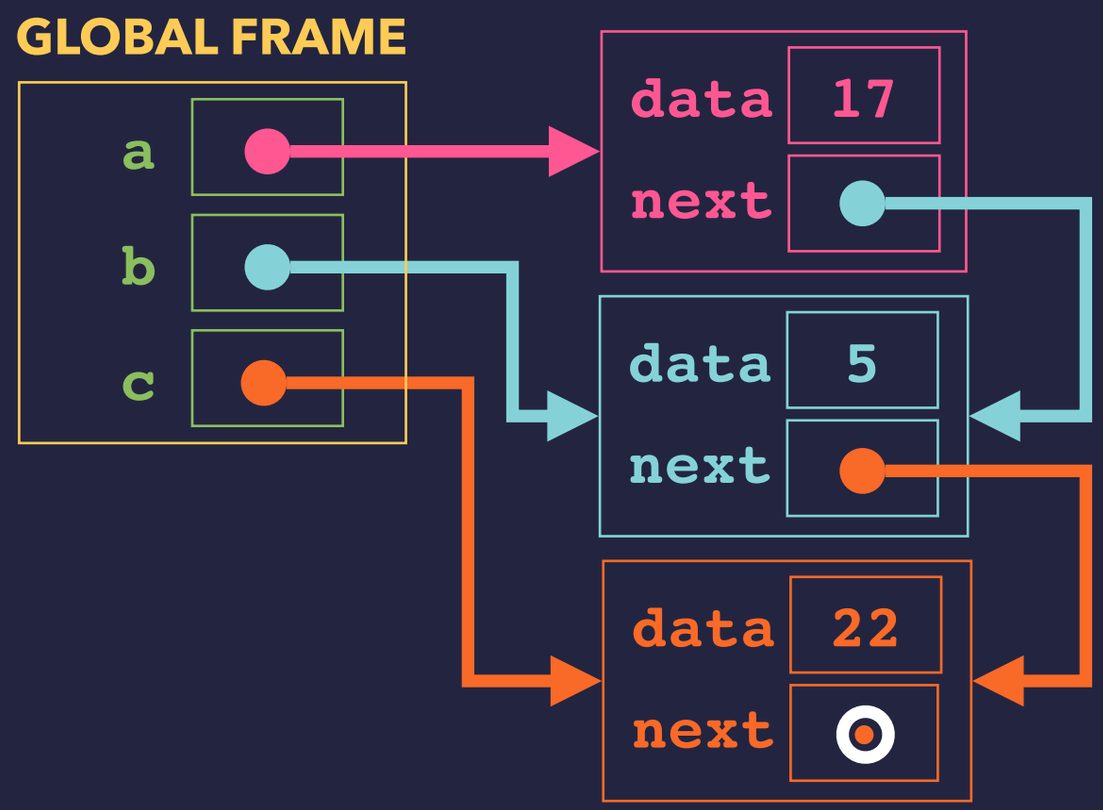
```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None

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



THIS STRUCTURE IS CALLED A LINKED LIST

ACCESSING VALUES THROUGH SEVERAL 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
>>> print(a.value)
17
>>> print(a.next.value)
5
>>> print(a.next.next.value)
22
>>>
```



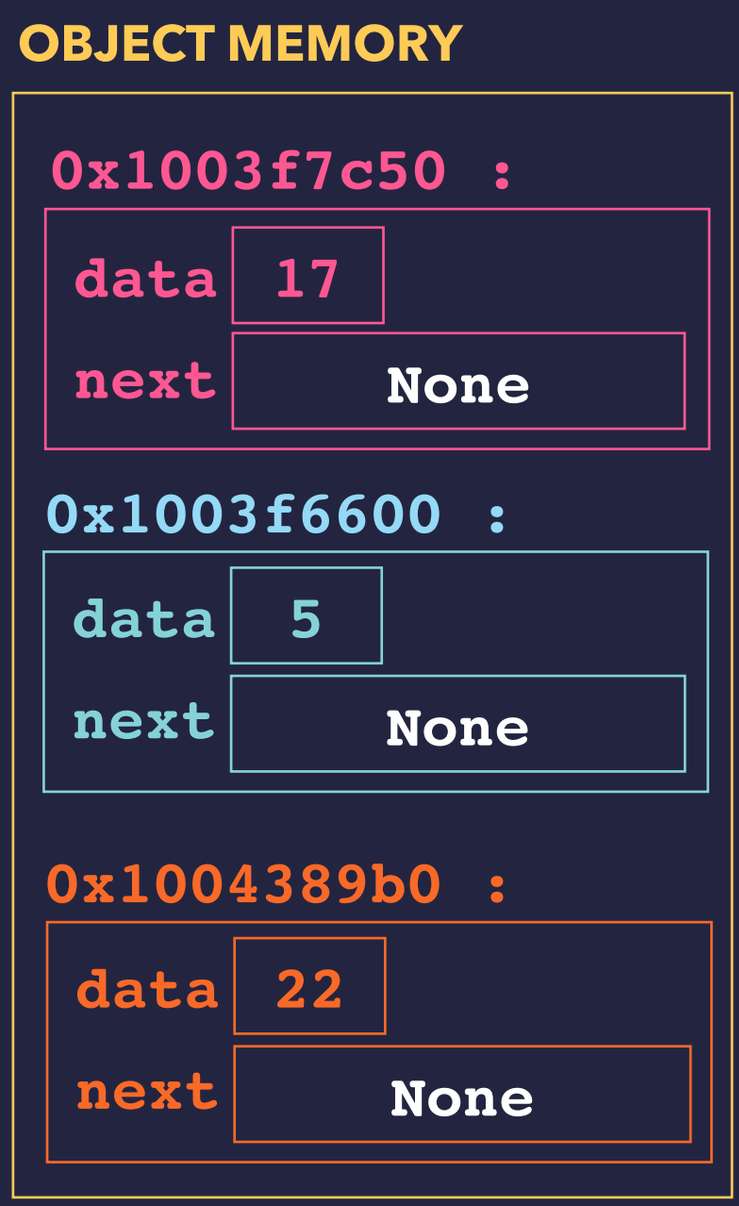
A WAY OF THINKING ABOUT REFERENCES

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

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> print(a)
<__main__.Node object at 0x1003f7c50>
>>> print(b)
<__main__.Node object at 0x1003f6600>
>>> print(c)
<__main__.Node object at 0x1004389b0>
>>>
```

A WAY OF THINKING ABOUT REFERENCES

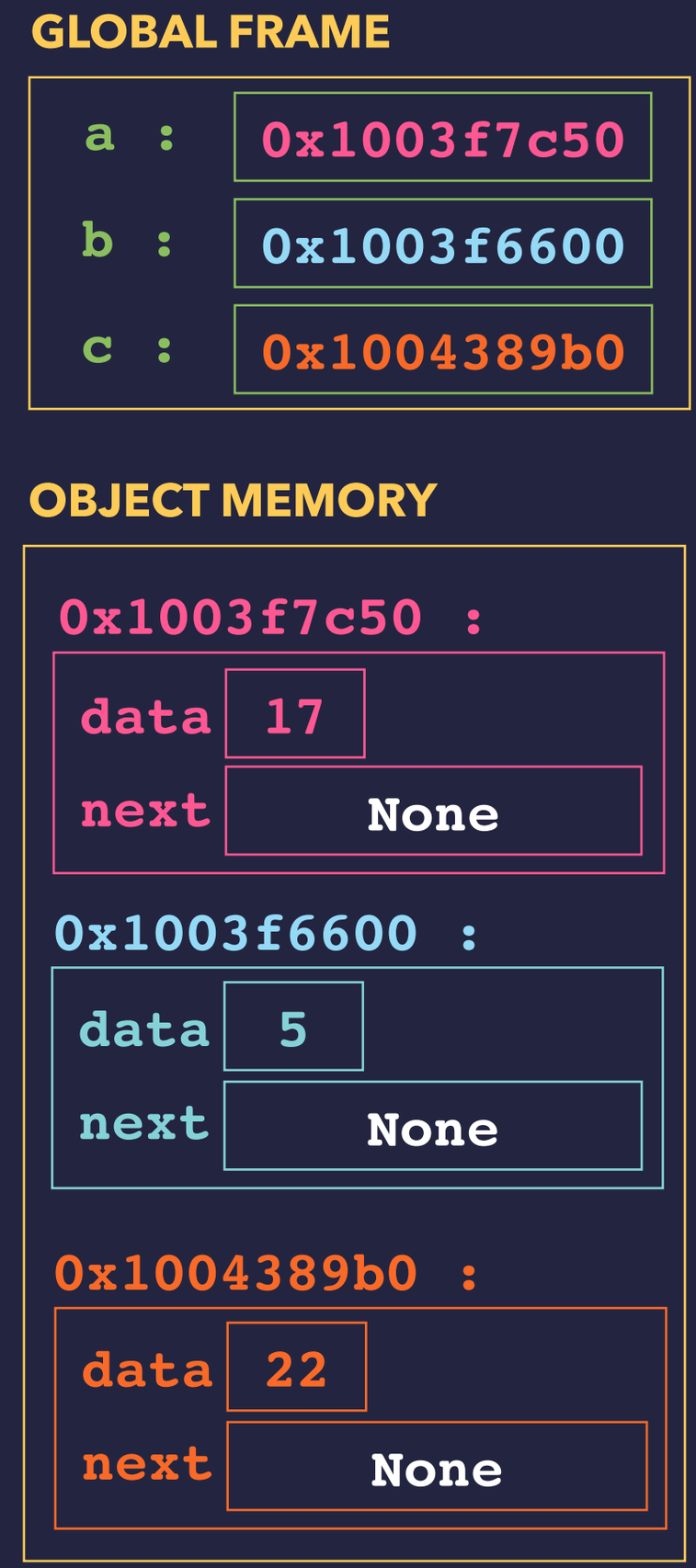
```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None
```

```
>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> print(a)
<__main__.Node object at 0x1003f7c50>
>>> print(b)
<__main__.Node object at 0x1003f6600>
>>> print(c)
<__main__.Node object at 0x1004389b0>
>>>
```



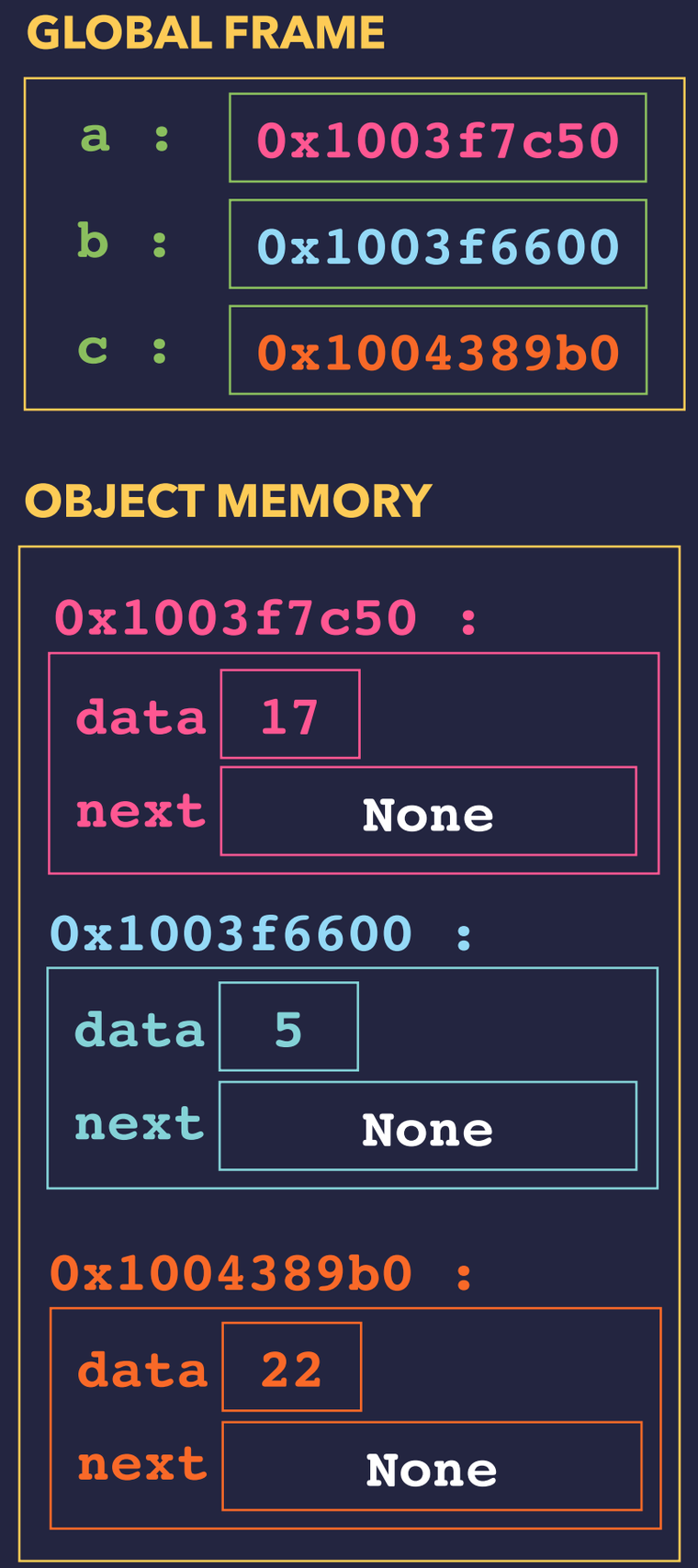
A WAY OF THINKING ABOUT REFERENCES

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

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> print(a)
<__main__.Node object at 0x1003f7c50>
>>> print(b)
<__main__.Node object at 0x1003f6600>
>>> print(c)
<__main__.Node object at 0x1004389b0>
>>>
```



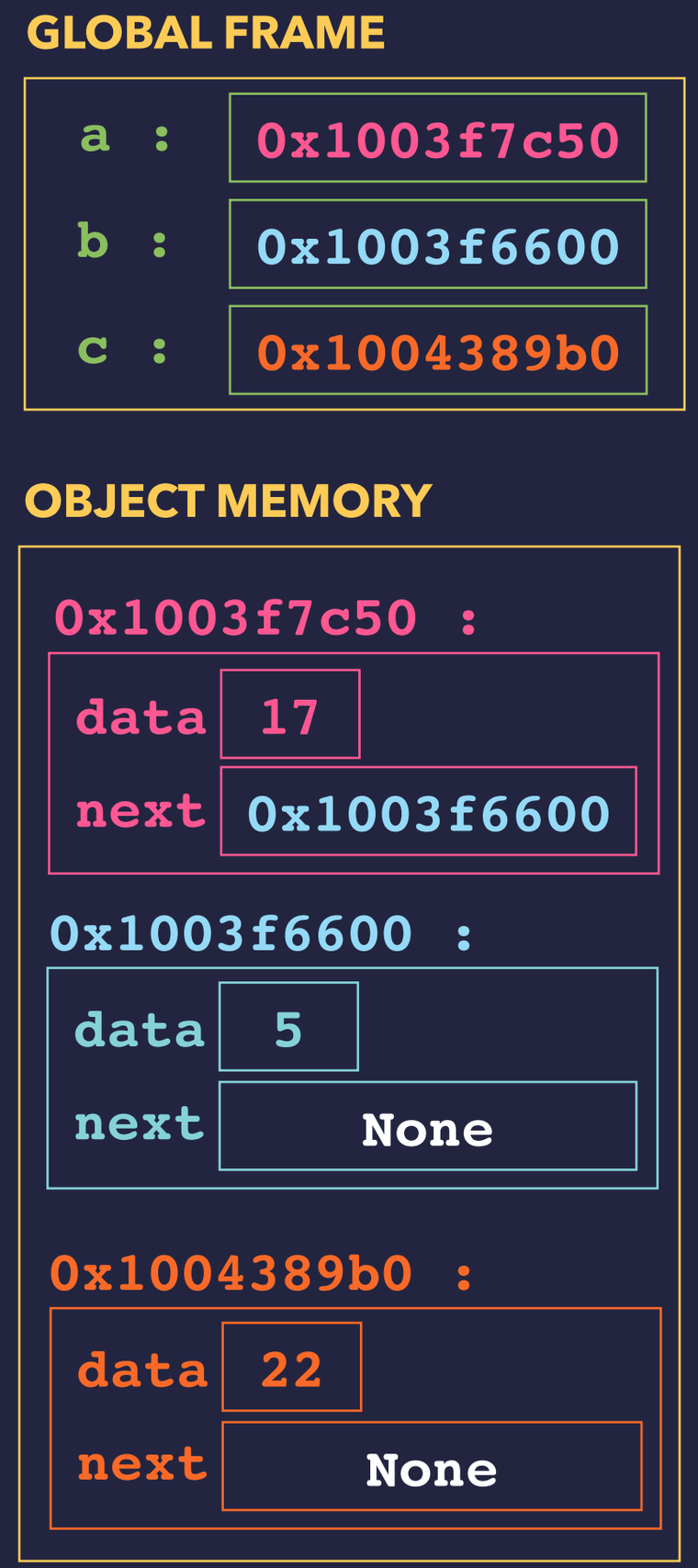
A WAY OF THINKING ABOUT REFERENCES

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

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> print(a)
<__main__.Node object at 0x1003f7c50>
>>> print(b)
<__main__.Node object at 0x1003f6600>
>>> print(c)
<__main__.Node object at 0x1004389b0>
>>> a.next = b
```



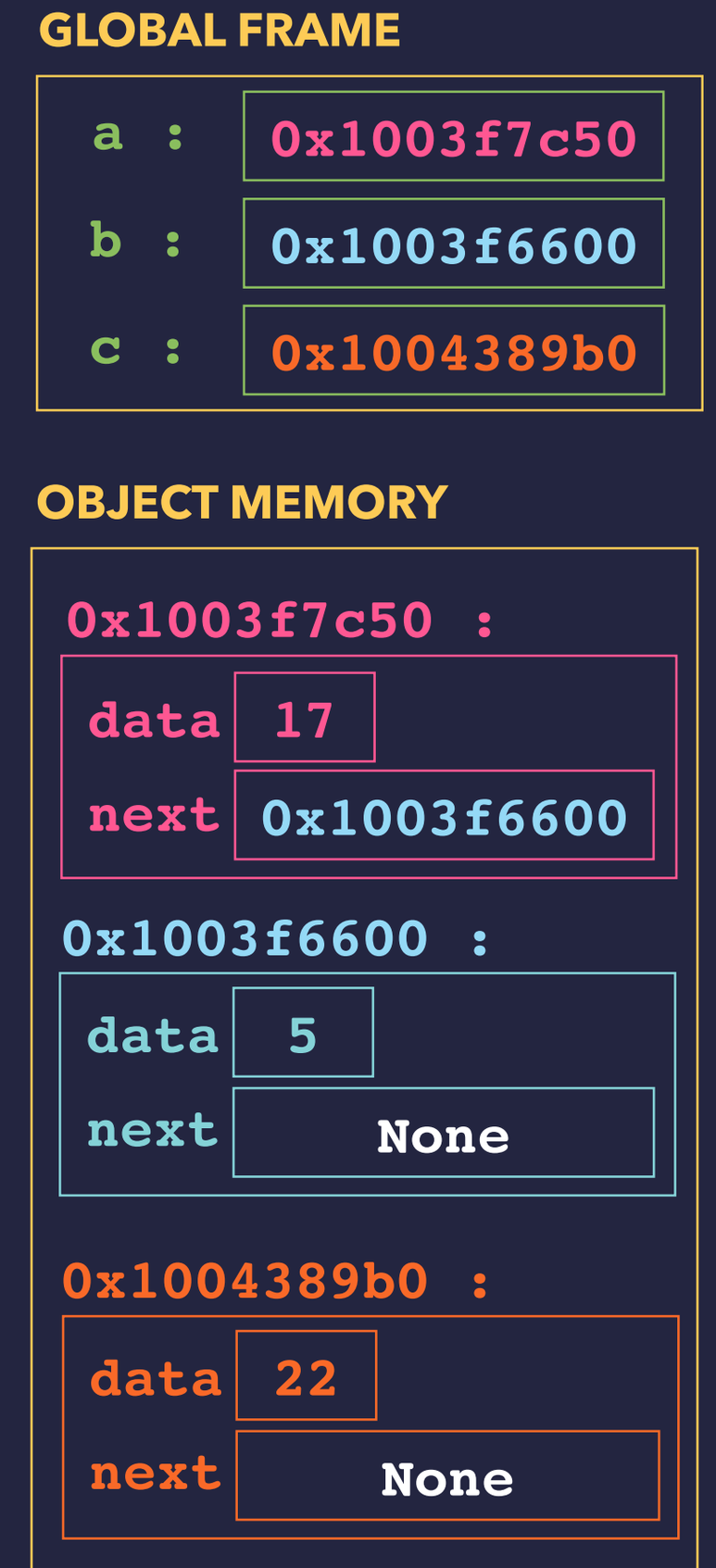
A WAY OF THINKING ABOUT REFERENCES

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

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> print(a)
<__main__.Node object at 0x1003f7c50>
>>> print(b)
<__main__.Node object at 0x1003f6600>
>>> print(c)
<__main__.Node object at 0x1004389b0>
>>> a.next = b
>>>
```



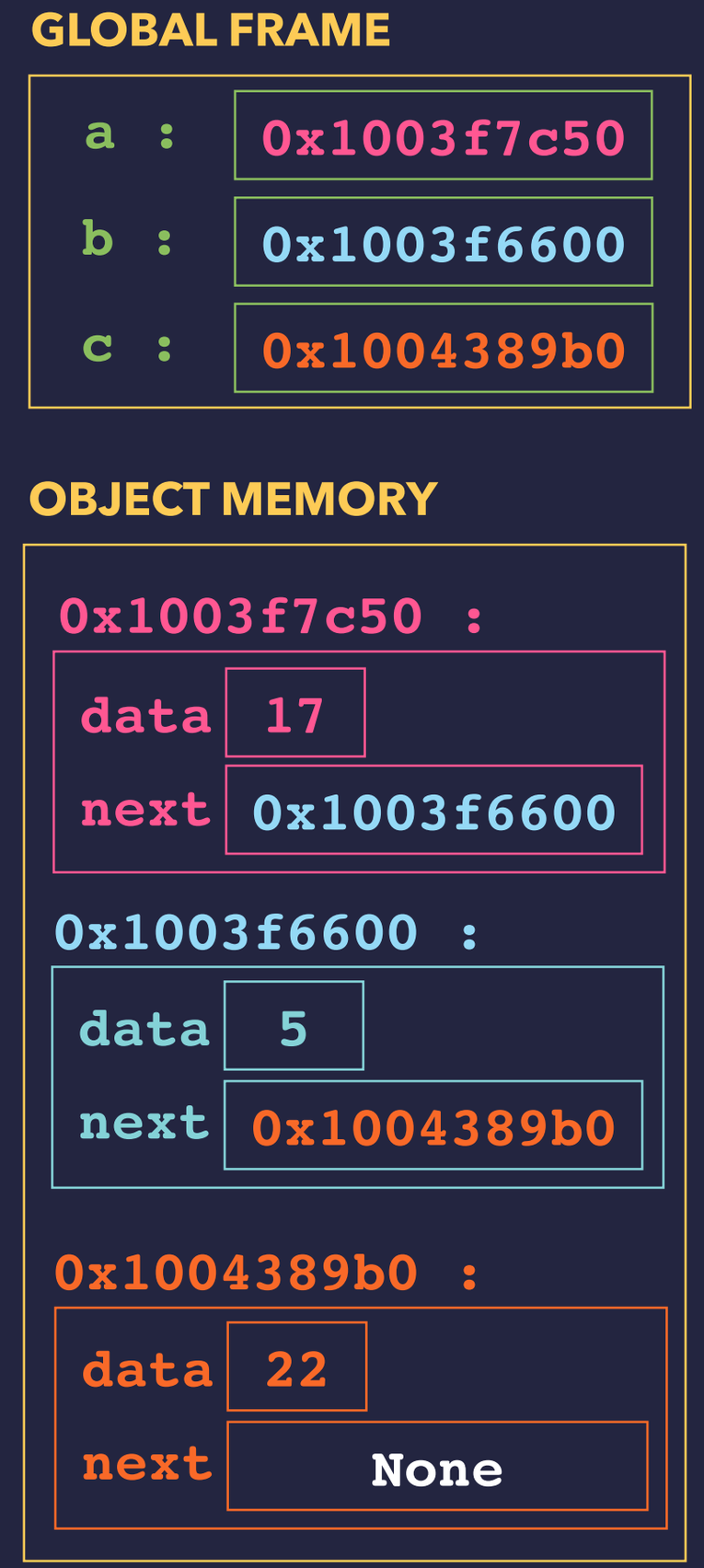
A WAY OF THINKING ABOUT REFERENCES

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

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> print(a)
<__main__.Node object at 0x1003f7c50>
>>> print(b)
<__main__.Node object at 0x1003f6600>
>>> print(c)
<__main__.Node object at 0x1004389b0>
>>> a.next = b
>>> b.next = c
```



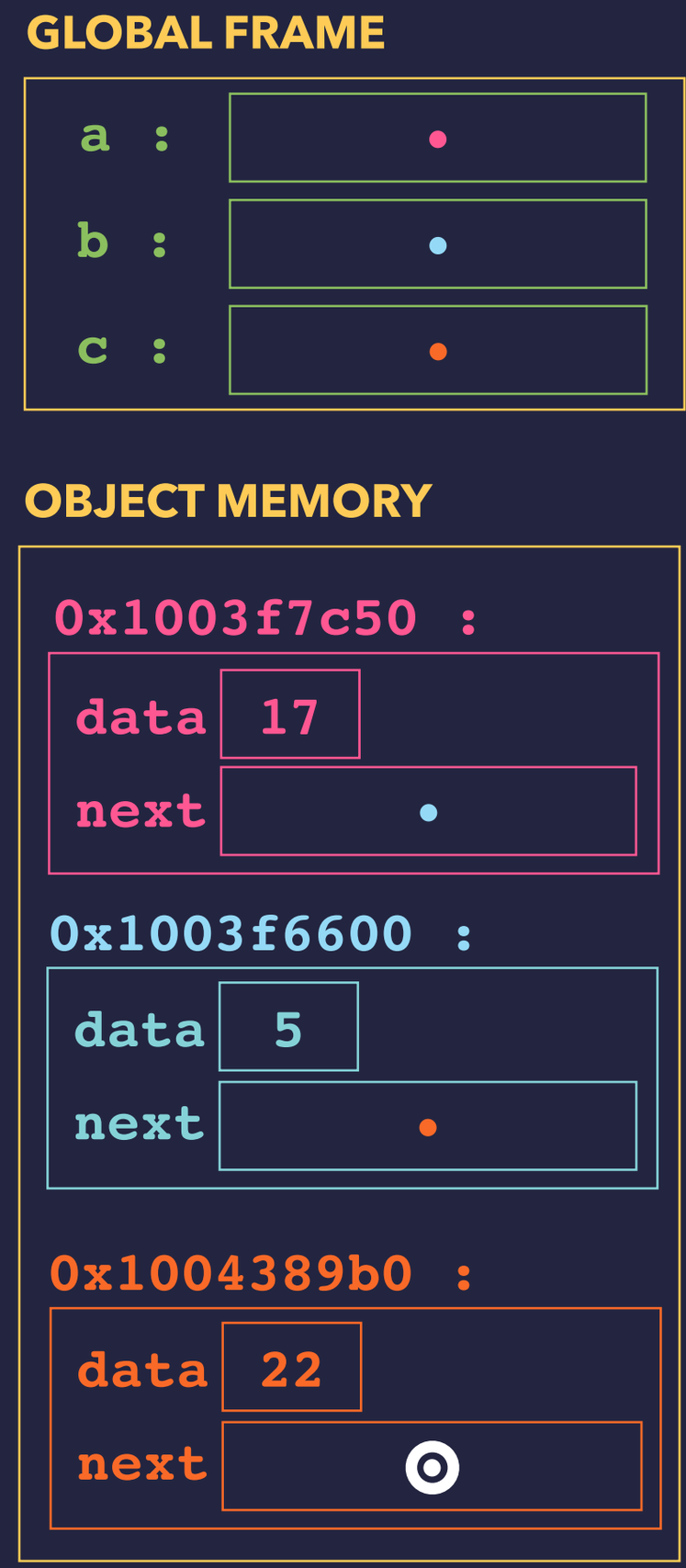
A WAY OF THINKING ABOUT REFERENCES

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

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> print(a)
<__main__.Node object at 0x1003f7c50>
>>> print(b)
<__main__.Node object at 0x1003f6600>
>>> print(c)
<__main__.Node object at 0x1004389b0>
>>> a.next = b
>>> b.next = c
>>>
```



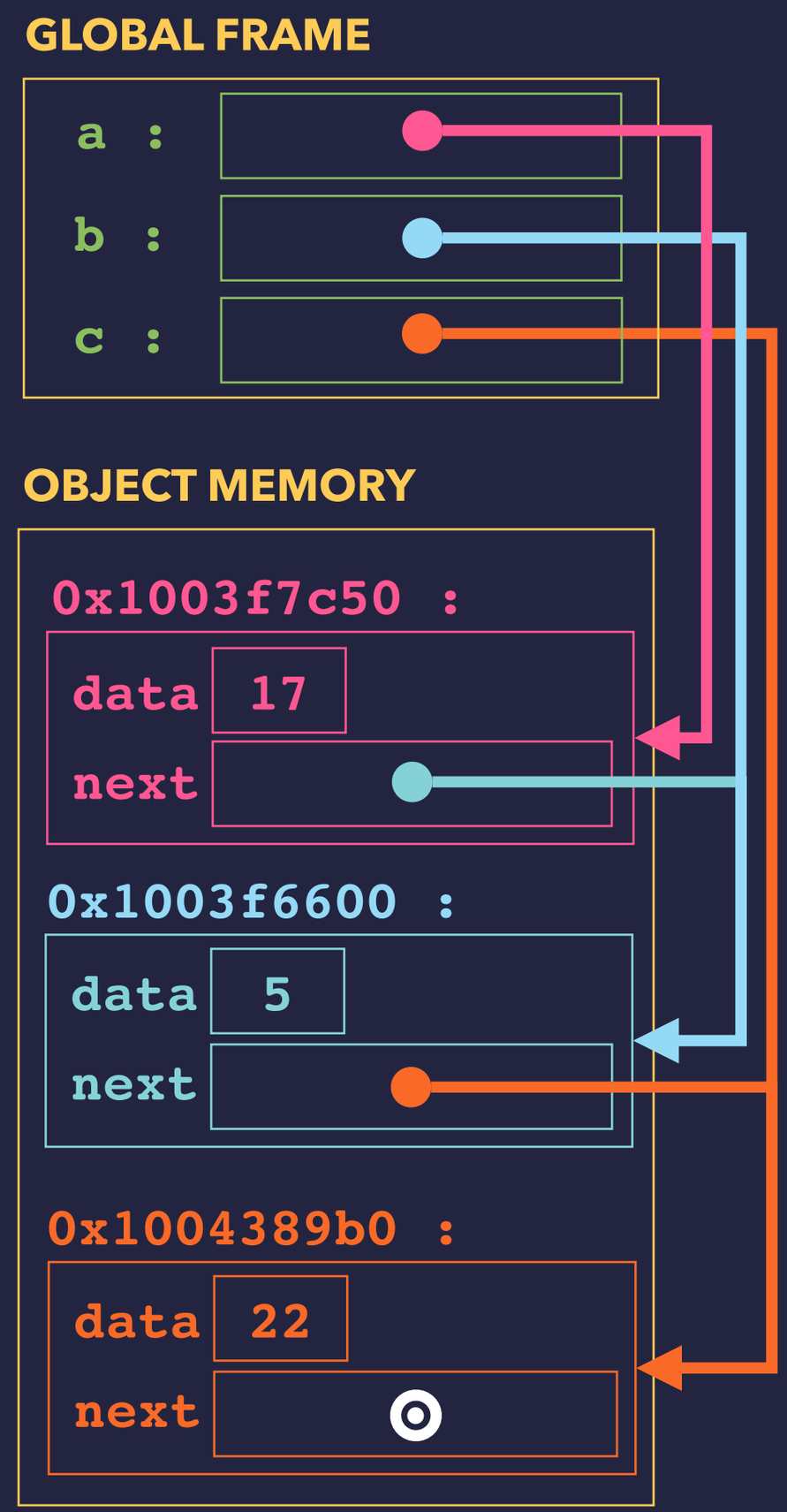
A WAY OF THINKING ABOUT REFERENCES

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

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> print(a)
<__main__.Node object at 0x1003f7c50>
>>> print(b)
<__main__.Node object at 0x1003f6600>
>>> print(c)
<__main__.Node object at 0x1004389b0>
>>> a.next = b
>>> b.next = c
>>>
```



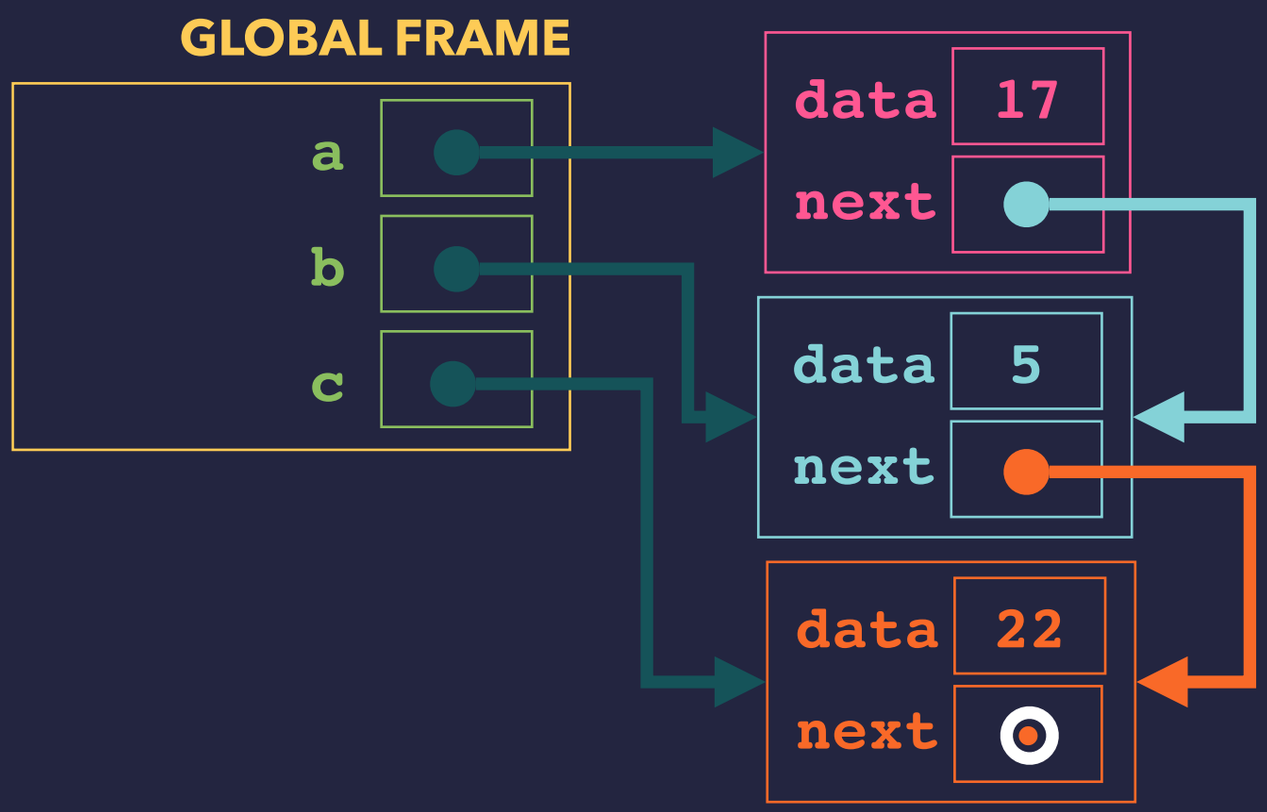
A WAY OF THINKING ABOUT REFERENCES

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

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> print(a)
<__main__.Node object at 0x1003f7c50>
>>> print(b)
<__main__.Node object at 0x1003f6600>
>>> print(c)
<__main__.Node object at 0x1004389b0>
>>> a.next = b
>>> b.next = c
>>>
```



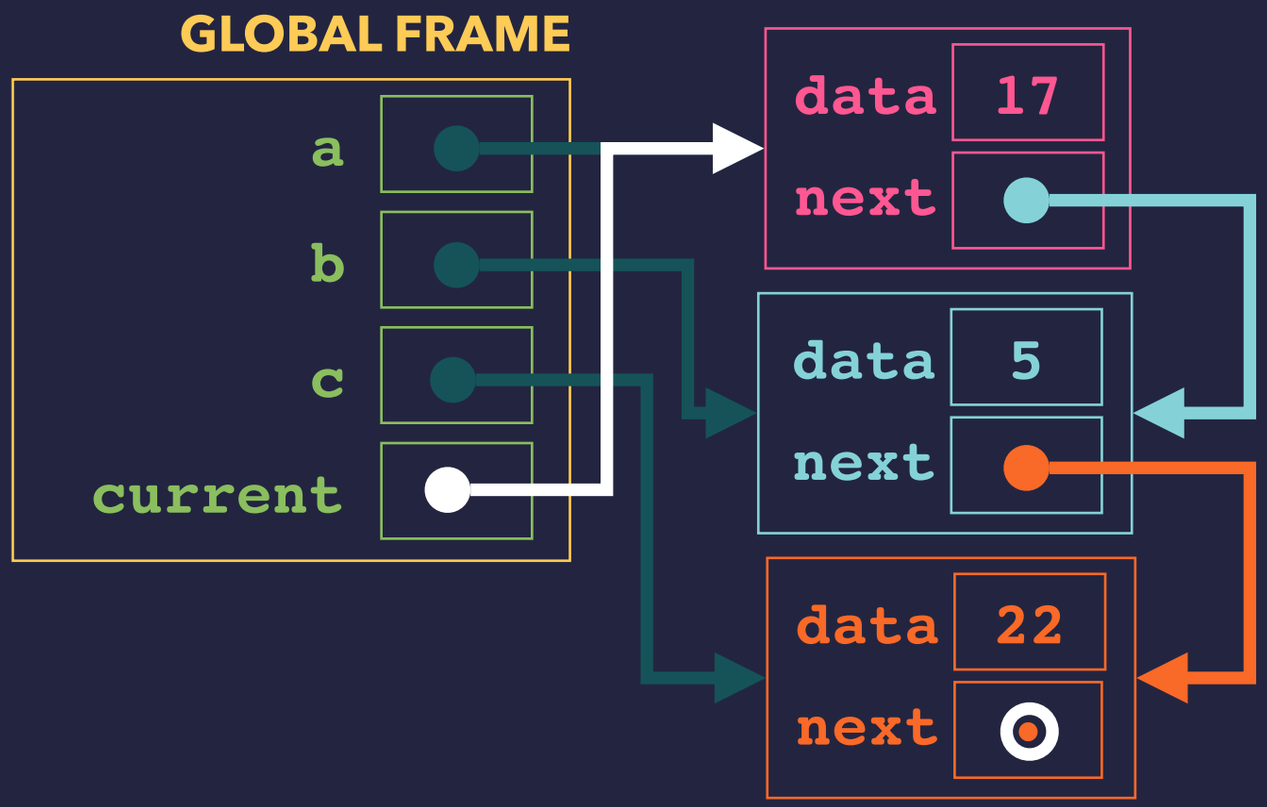
MOVING A REFERENCE DOWN THE LIST

```
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
>>> current = a
```



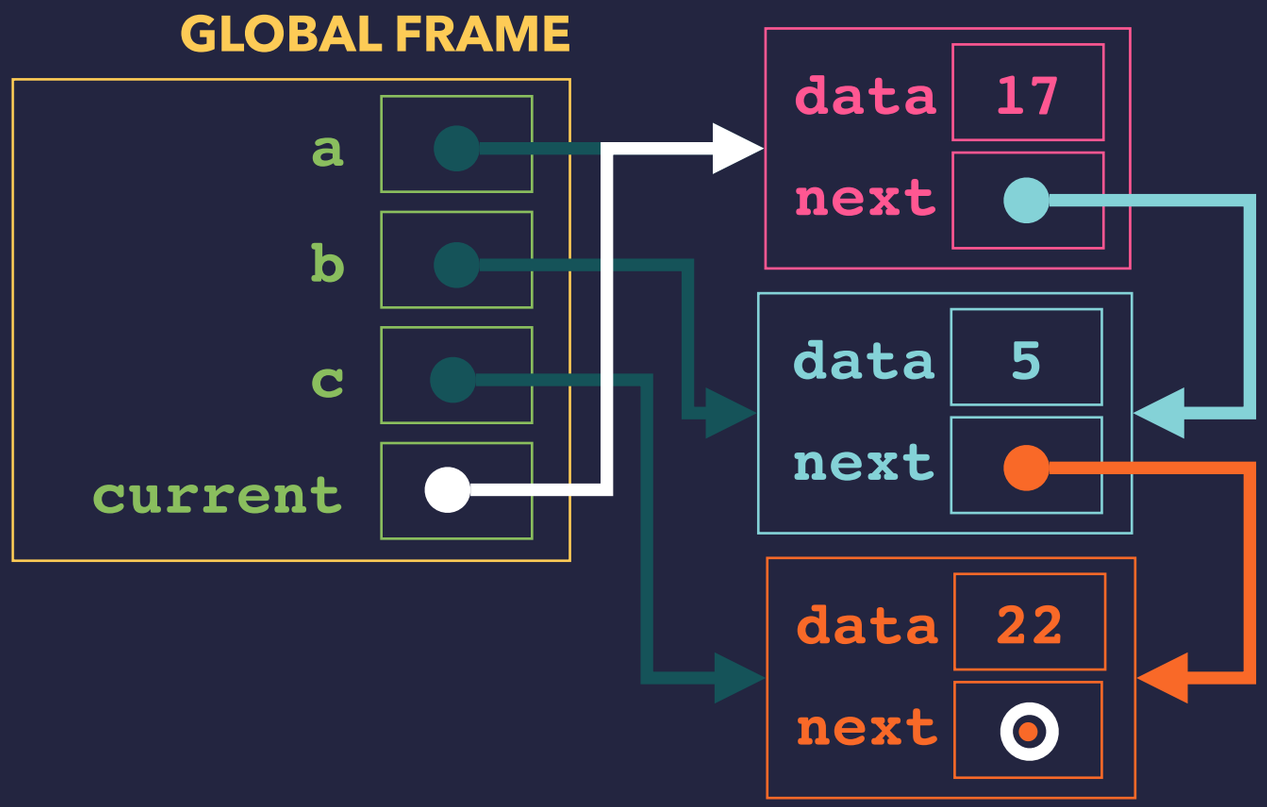
MOVING A REFERENCE DOWN THE LIST

```
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
>>> current = a
>>>
```



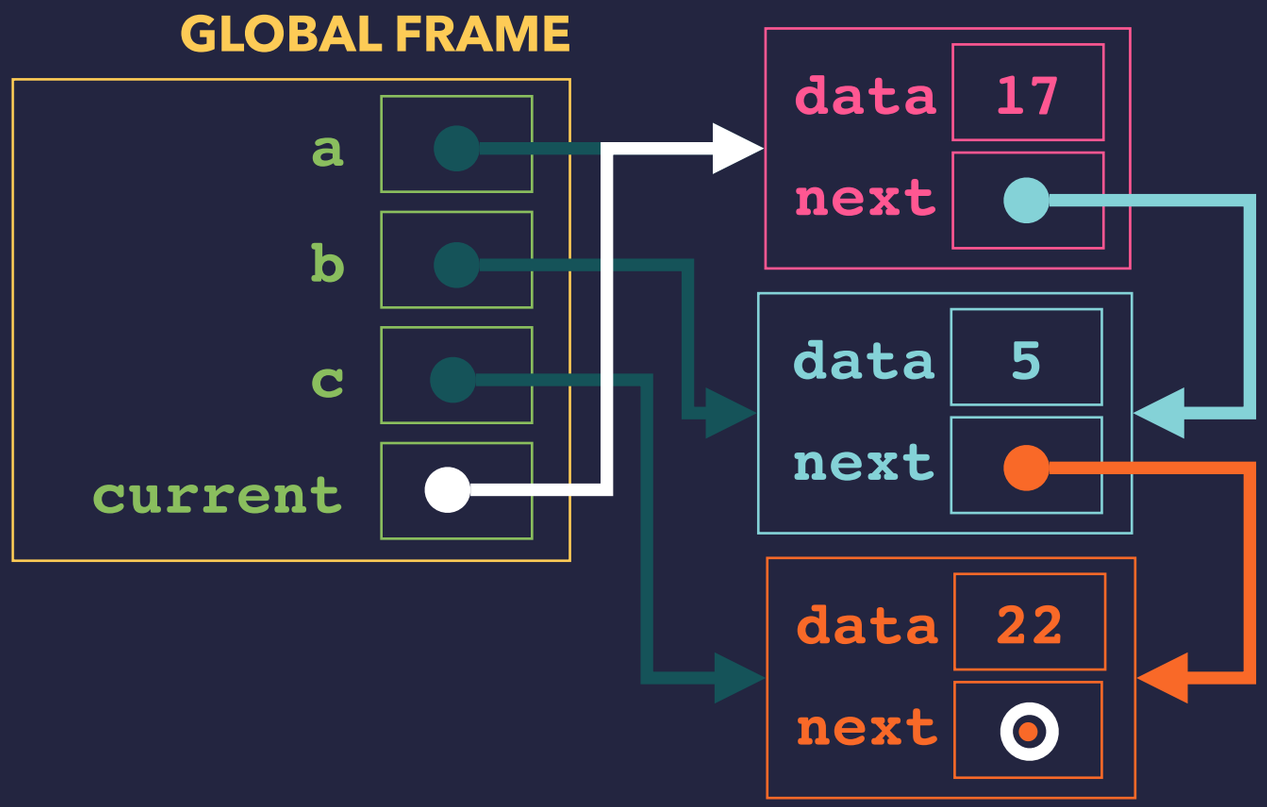
MOVING A REFERENCE DOWN THE LIST

```
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
>>> current = a
>>> print(current.value)
17
>>> |
```



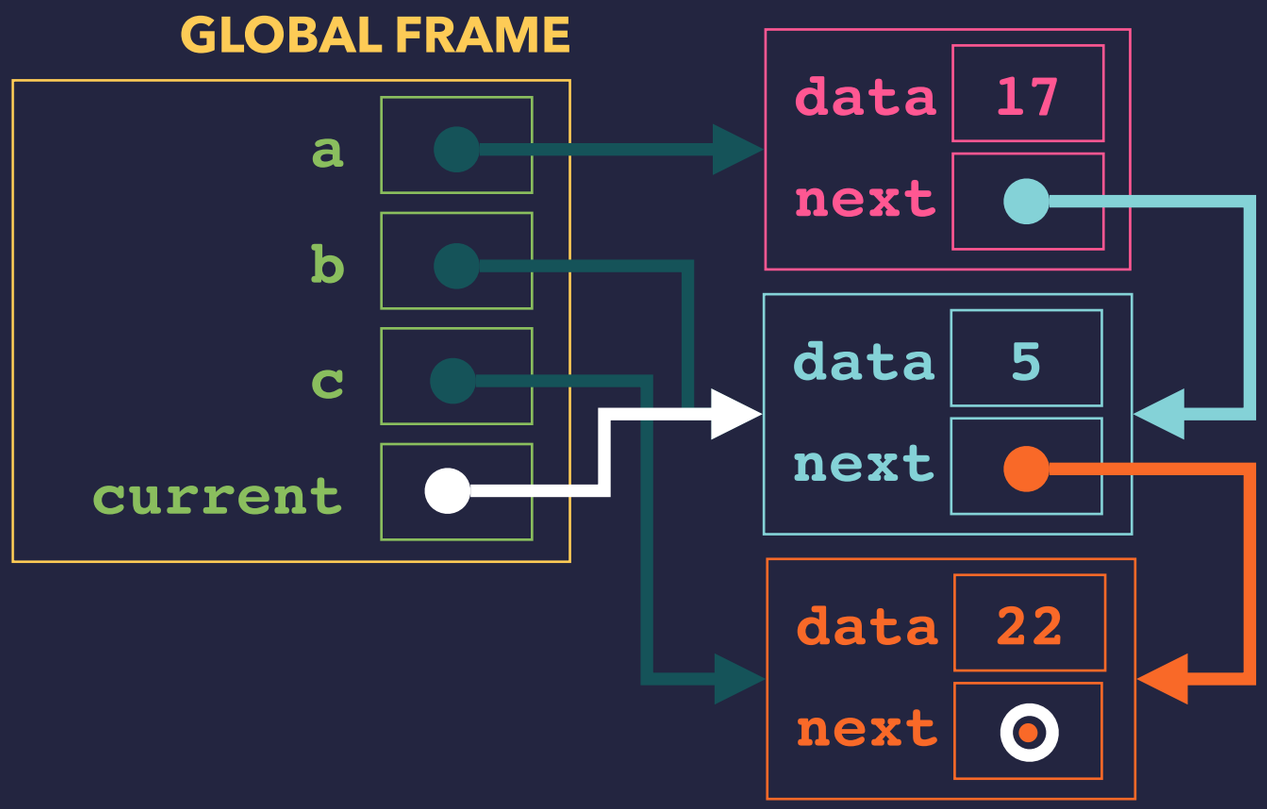
MOVING A REFERENCE DOWN THE LIST

```
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
>>> current = a
>>> print(current.value)
17
>>> current = current.next
```



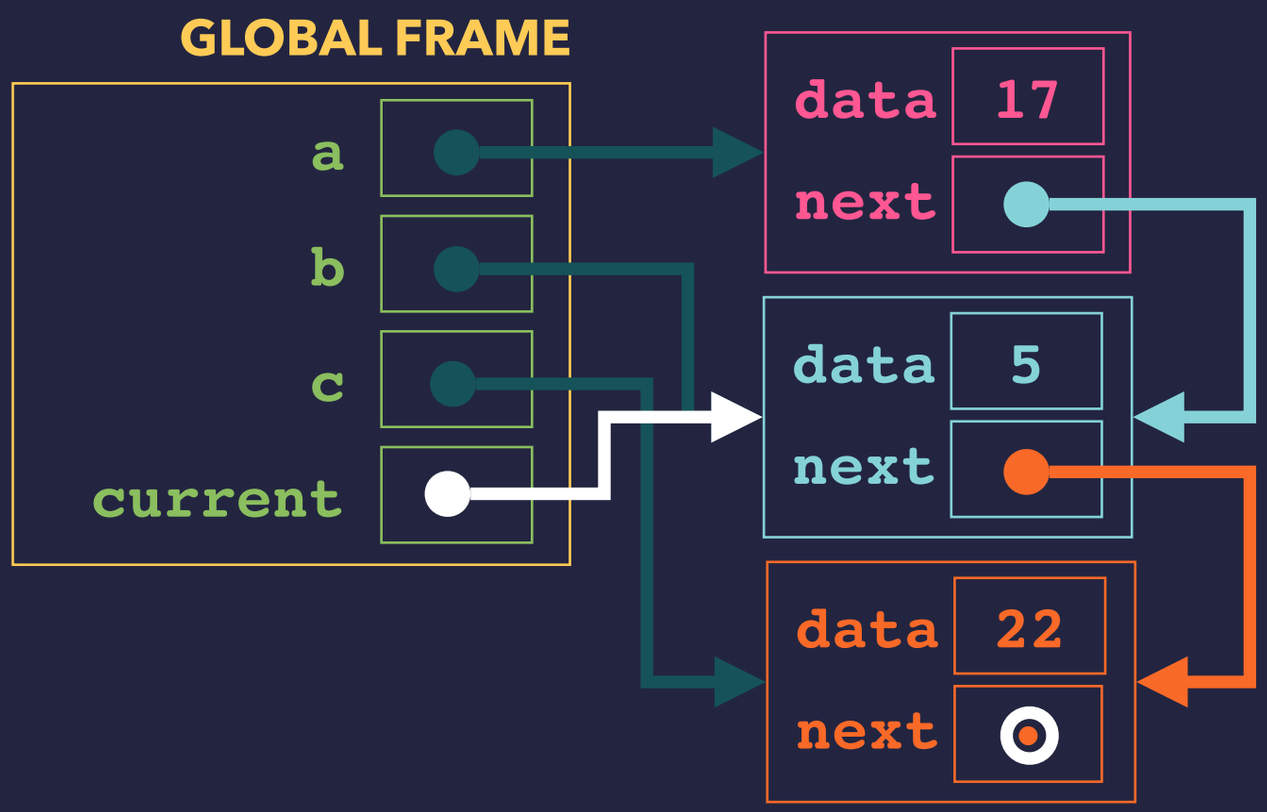
MOVING A REFERENCE DOWN THE LIST

```
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
>>> current = a
>>> print(current.value)
17
>>> current = current.next
>>> |
```



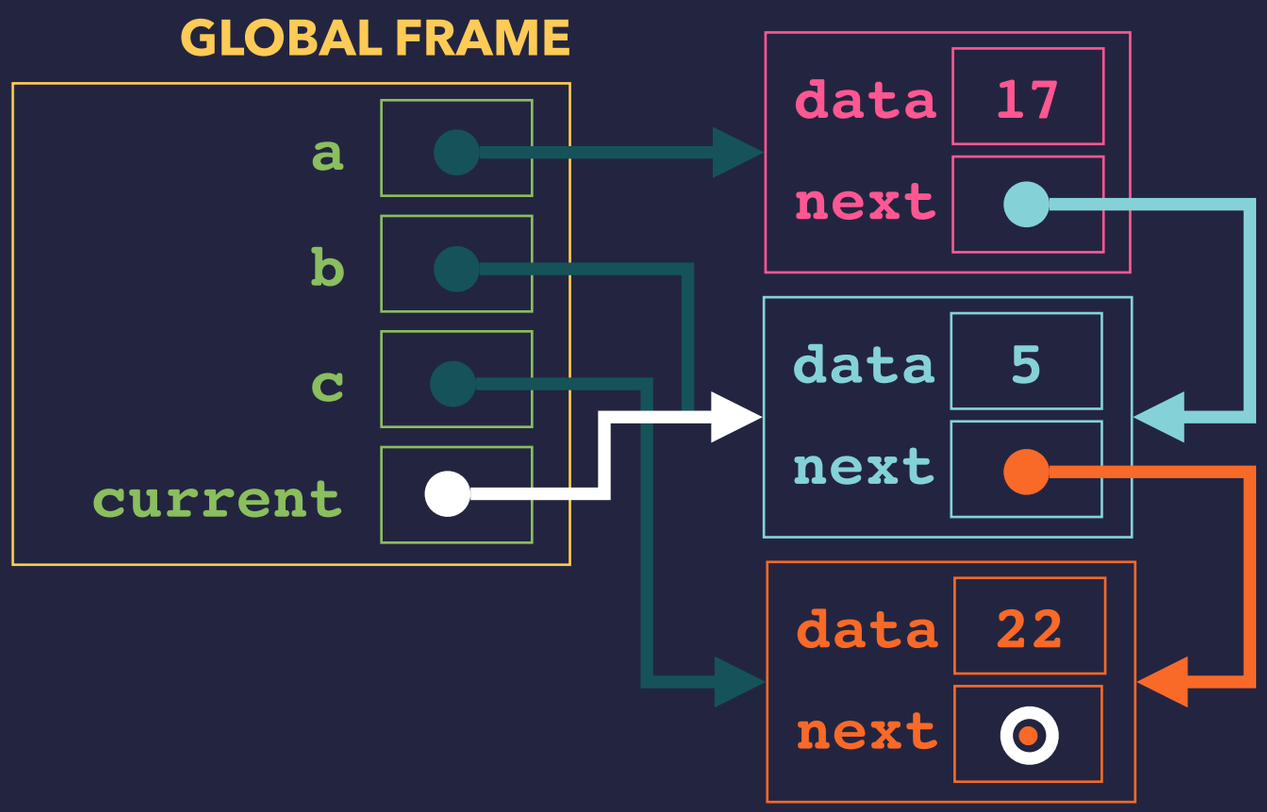
MOVING A REFERENCE DOWN THE LIST

```
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
>>> current = a
>>> print(current.value)
17
>>> current = current.next
>>> print(current.value)
5
>>>
```



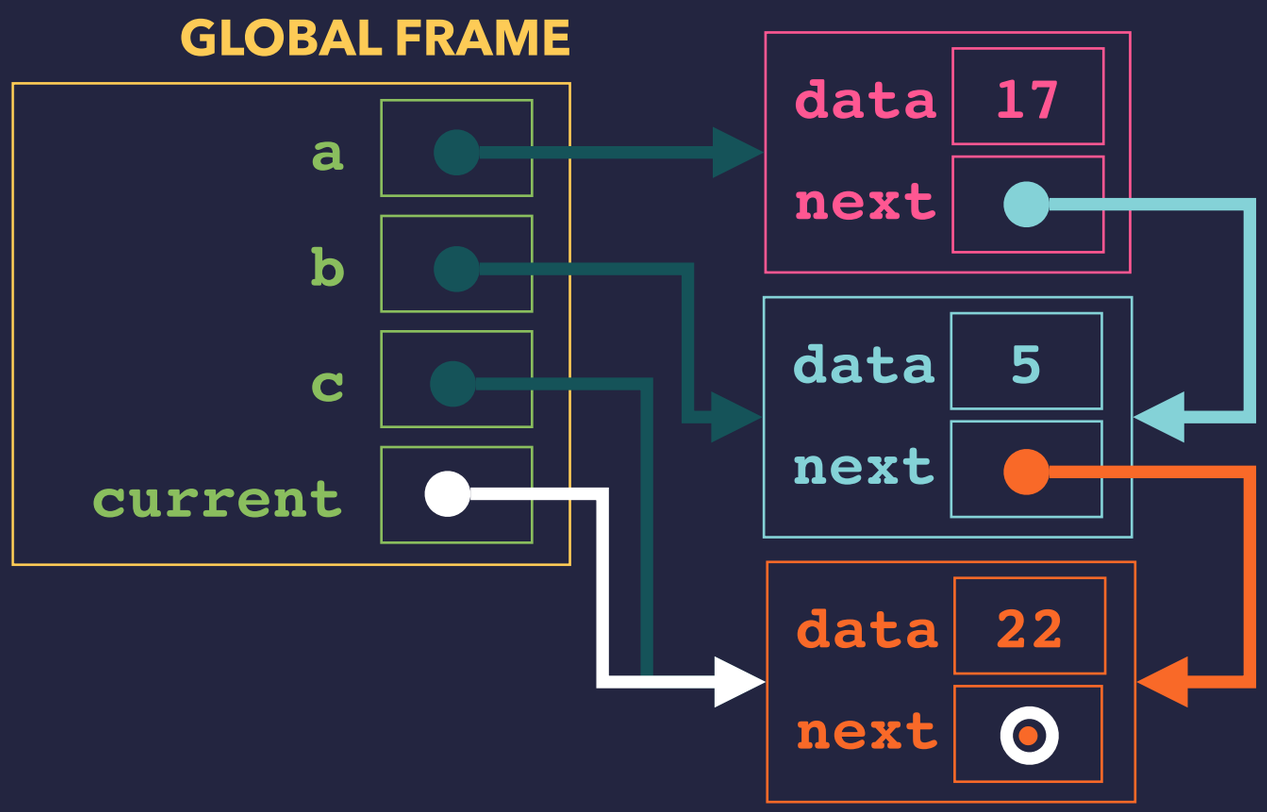
MOVING A REFERENCE DOWN THE LIST

```
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
>>> current = a
>>> print(current.value)
17
>>> current = current.next
>>> print(current.value)
5
>>> current = current.next
```



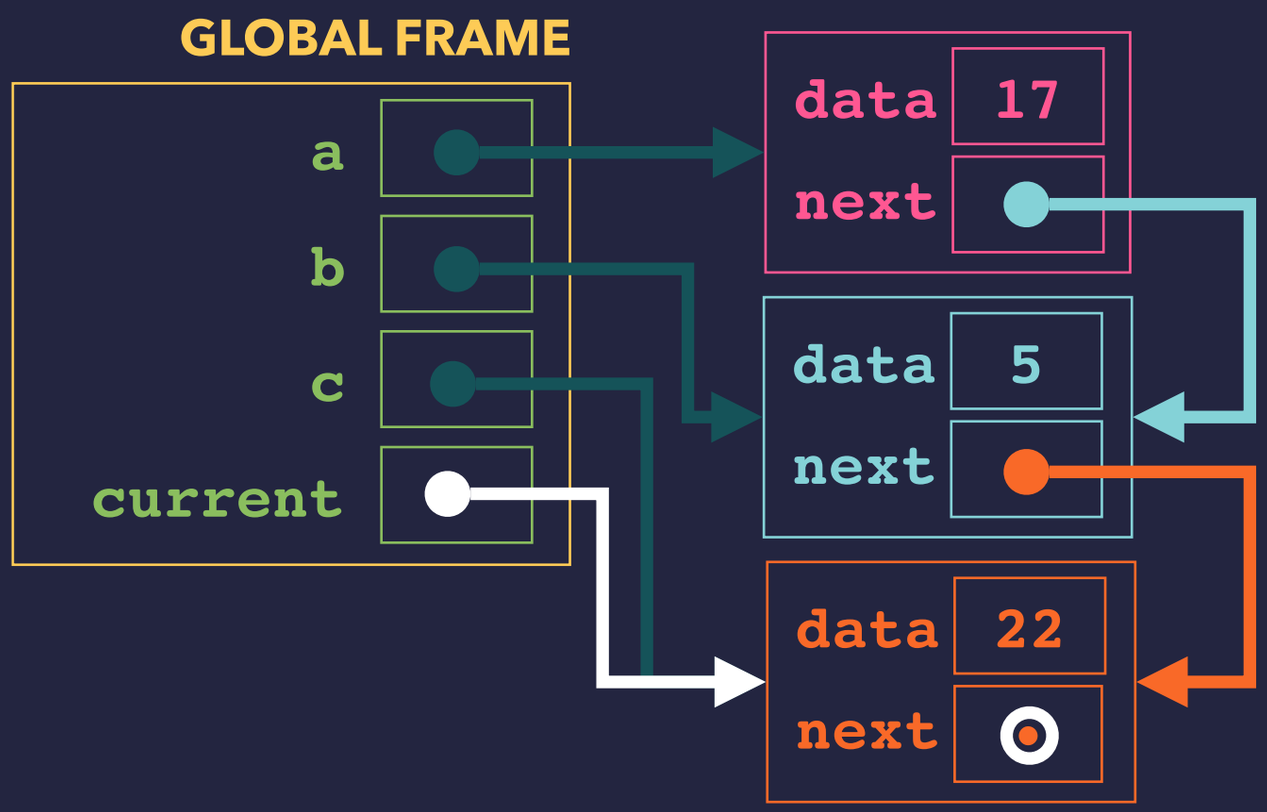
MOVING A REFERENCE DOWN THE LIST

```
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
>>> current = a
>>> print(current.value)
17
>>> current = current.next
>>> print(current.value)
5
>>> current = current.next
>>> |
```



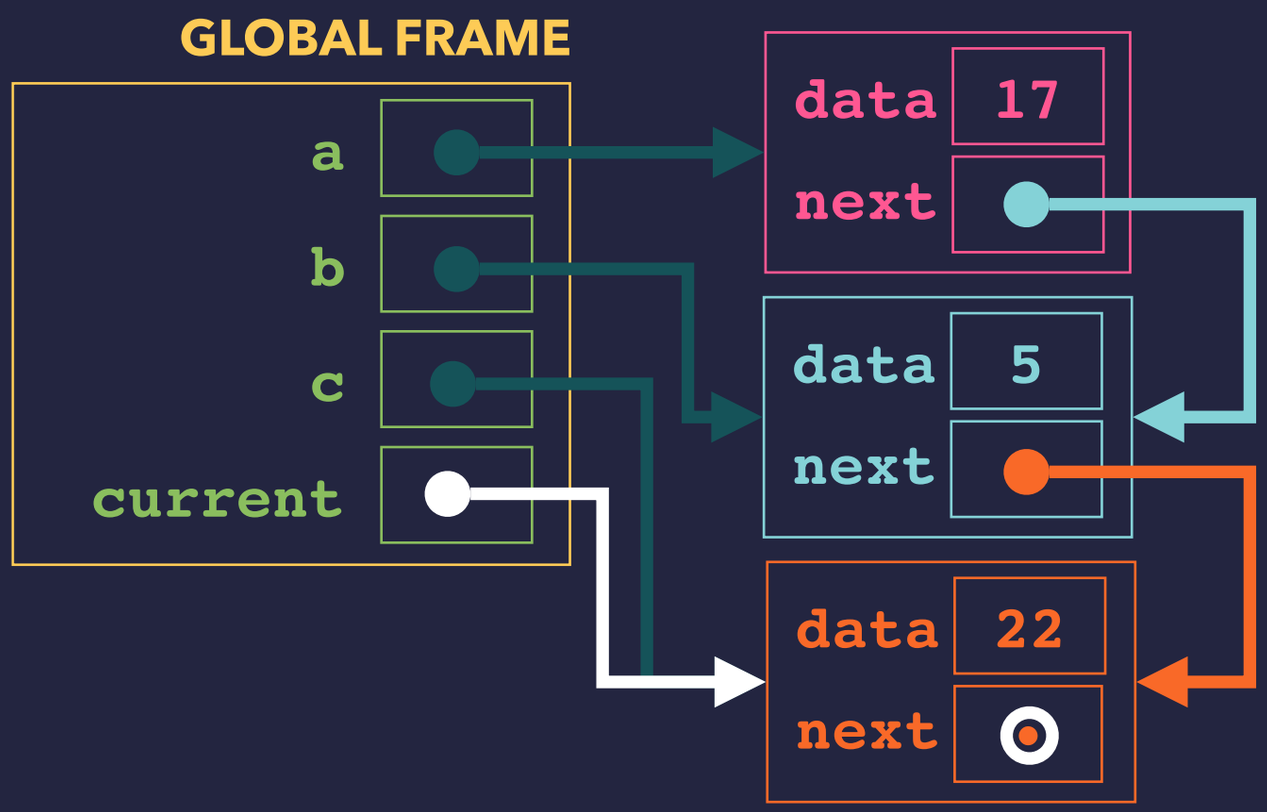
MOVING A REFERENCE DOWN THE LIST

```
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
>>> current = a
>>> print(current.value)
17
>>> current = current.next
>>> print(current.value)
5
>>> current = current.next
>>> print(current.value)
22
>>>
```



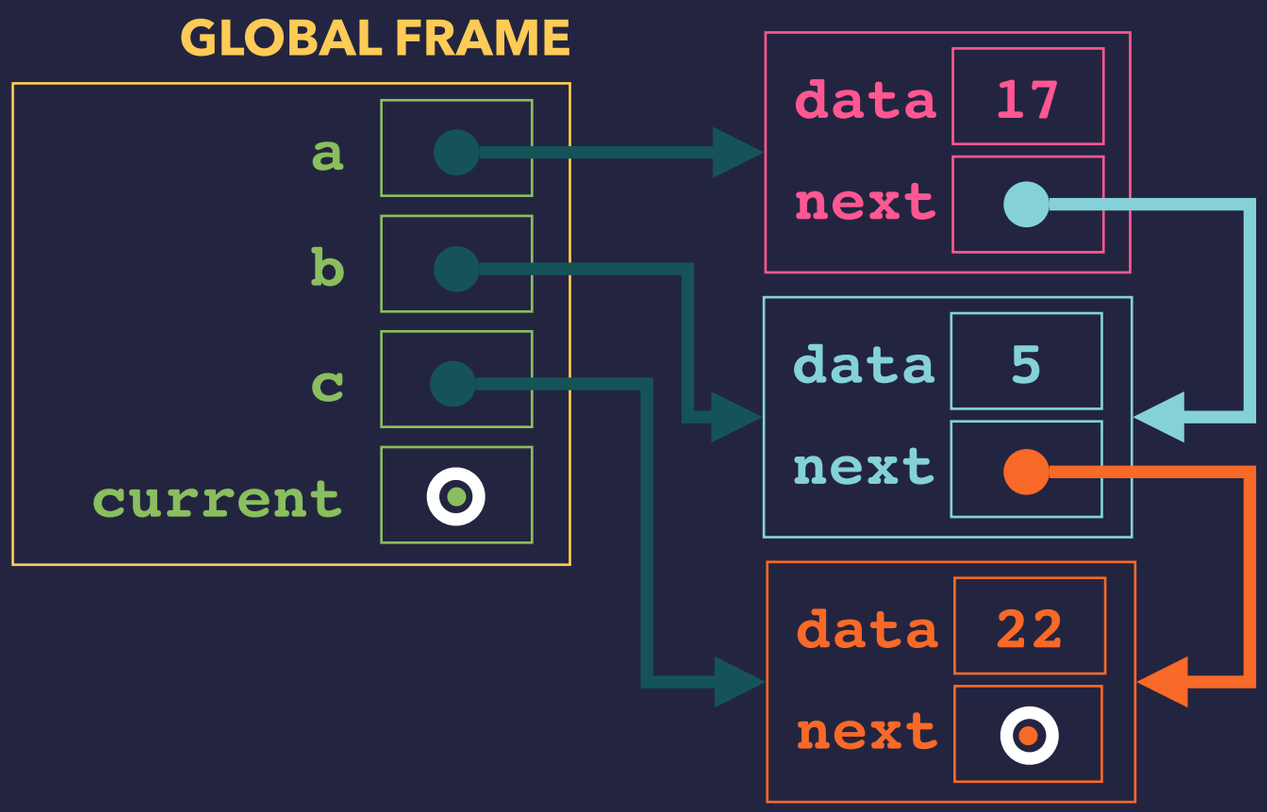
MOVING A REFERENCE DOWN THE LIST

```
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
>>> current = a
>>> print(current.value)
17
>>> current = current.next
>>> print(current.value)
5
>>> current = current.next
>>> print(current.value)
22
>>> current = current.next
```



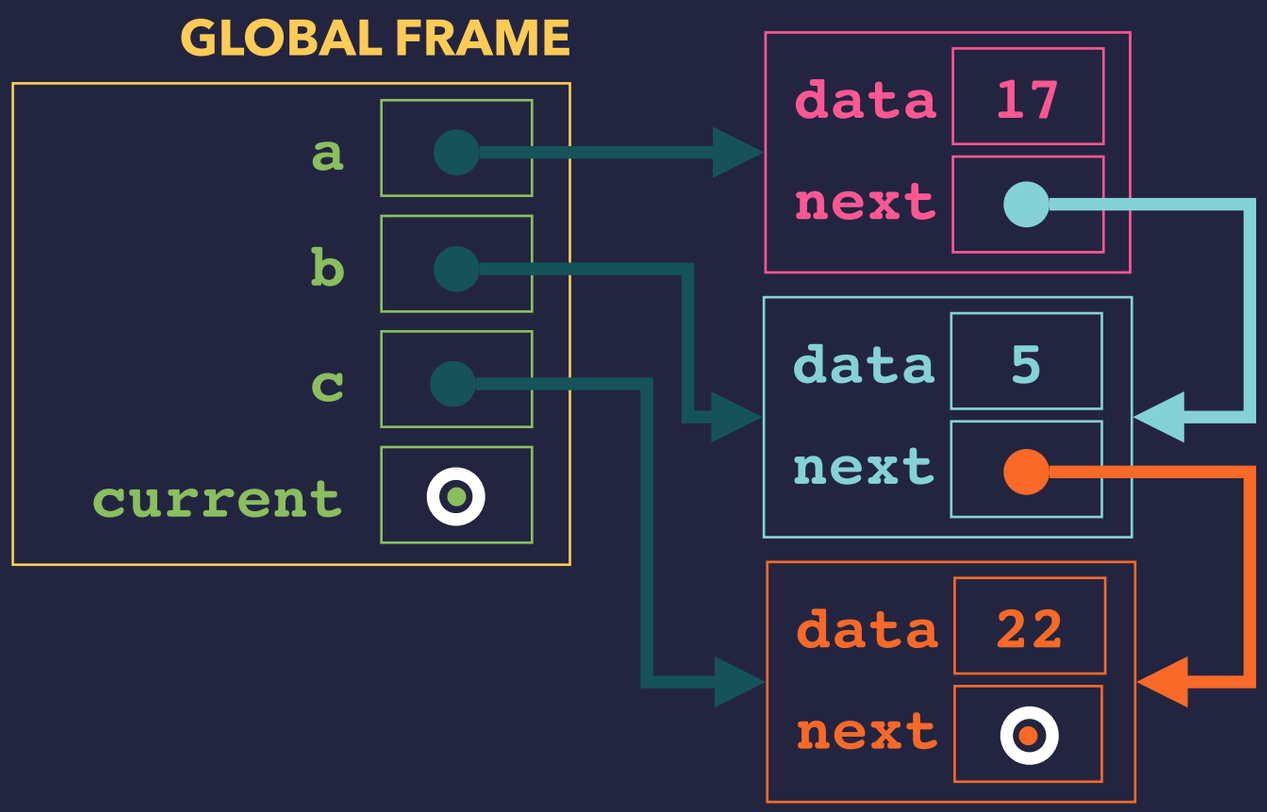
MOVING A REFERENCE DOWN THE LIST

```
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
>>> current = a
>>> print(current.value)
17
>>> current = current.next
>>> print(current.value)
5
>>> current = current.next
>>> print(current.value)
22
>>> current = current.next
>>>
```



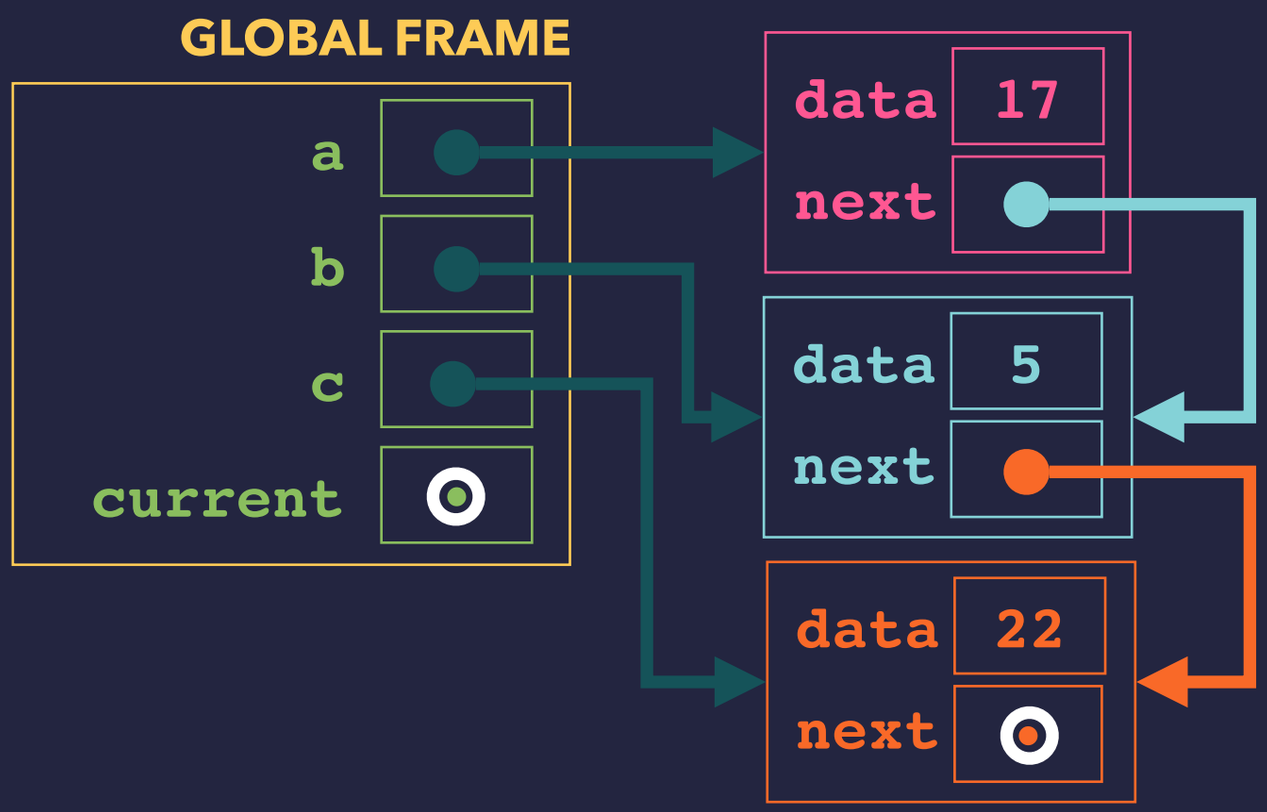
MOVING A REFERENCE DOWN THE LIST

```
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
>>> current = a
>>> print(current.value)
17
>>> current = current.next
>>> print(current.value)
5
>>> current = current.next
>>> print(current.value)
22
>>> current = current.next
>>> print(current)
```



MOVING A REFERENCE DOWN THE LIST

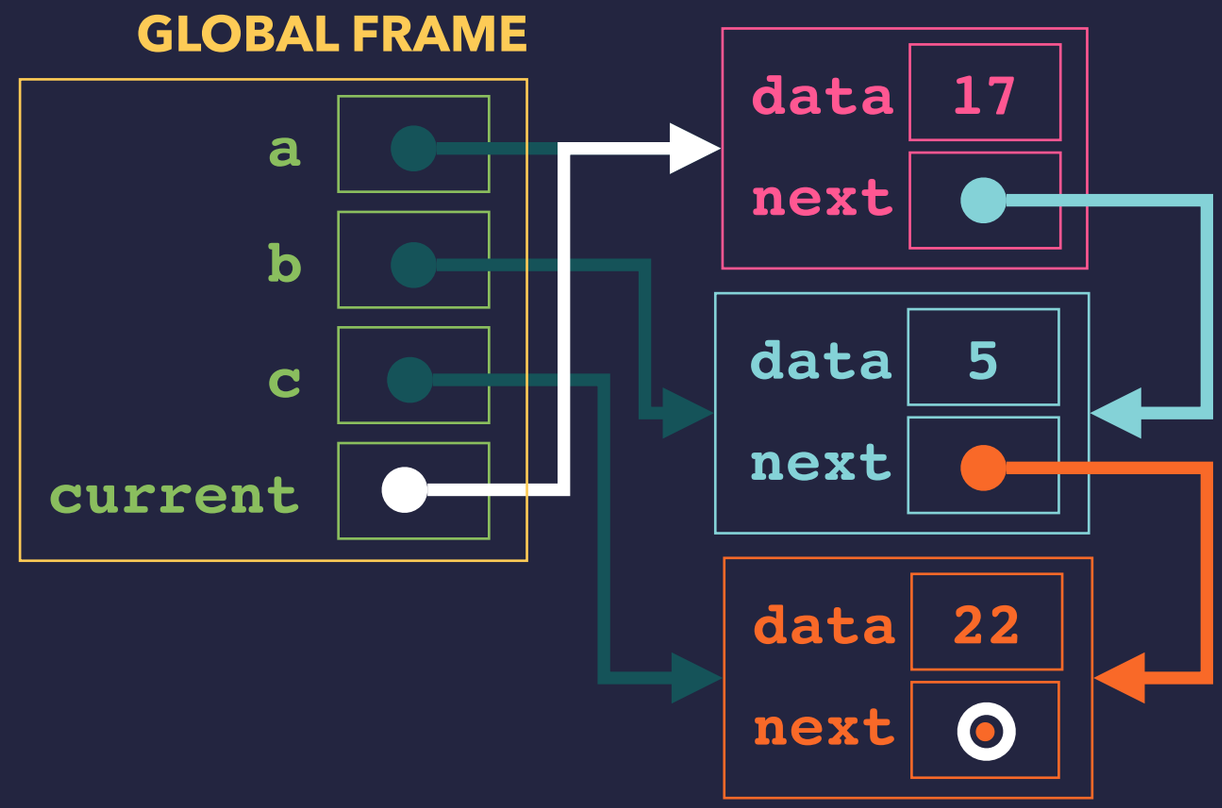
```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> a.next = b
>>> b.next = c
>>> current = a
>>> print(current.value)
17
>>> current = current.next
>>> print(current.value)
5
>>> current = current.next
>>> print(current.value)
22
>>> current = current.next
>>> print(current)
None
>>>
```



LOOPING THROUGH A LINKED LIST

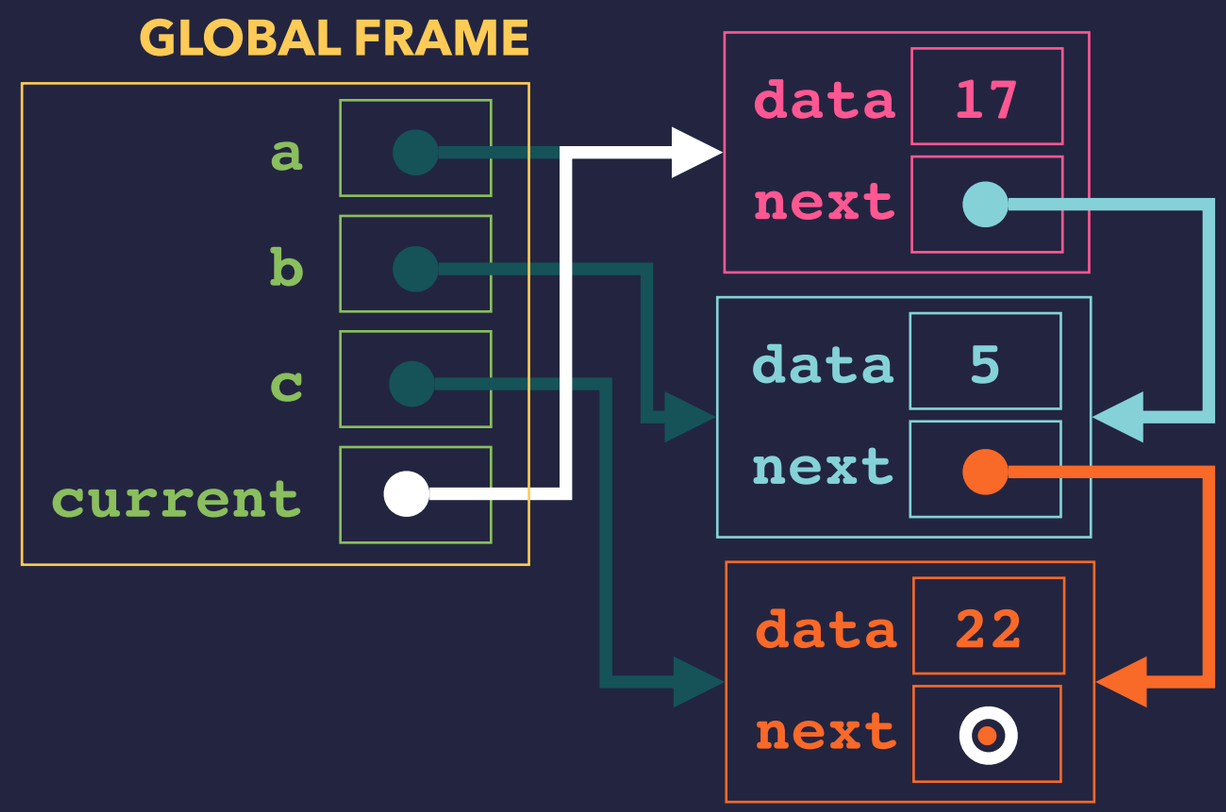
WALKING DOWN A LIST: TRAVERSAL

```
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
>>> current = a
>>> |
```



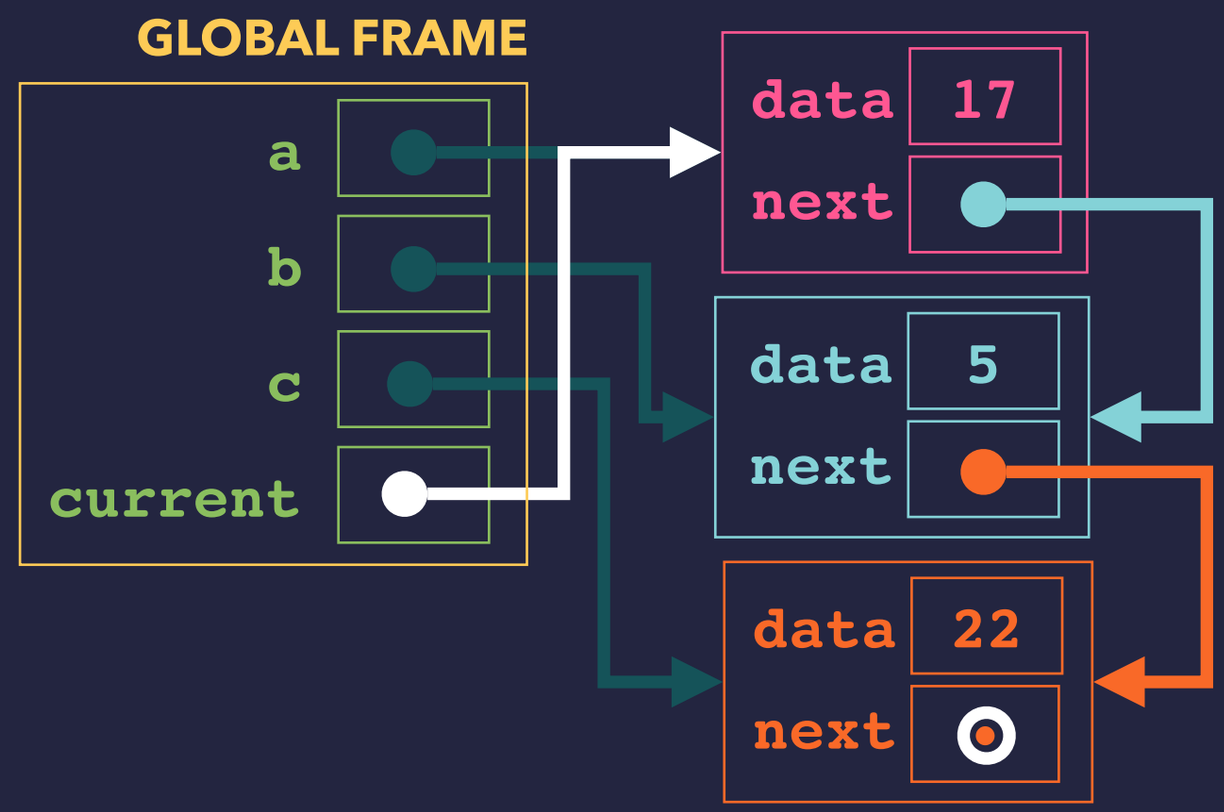
WALKING DOWN A LIST: TRAVERSAL

```
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
>>> current = a
>>> while current != None:
....    print(current.value)
....    current = current.next
....
```



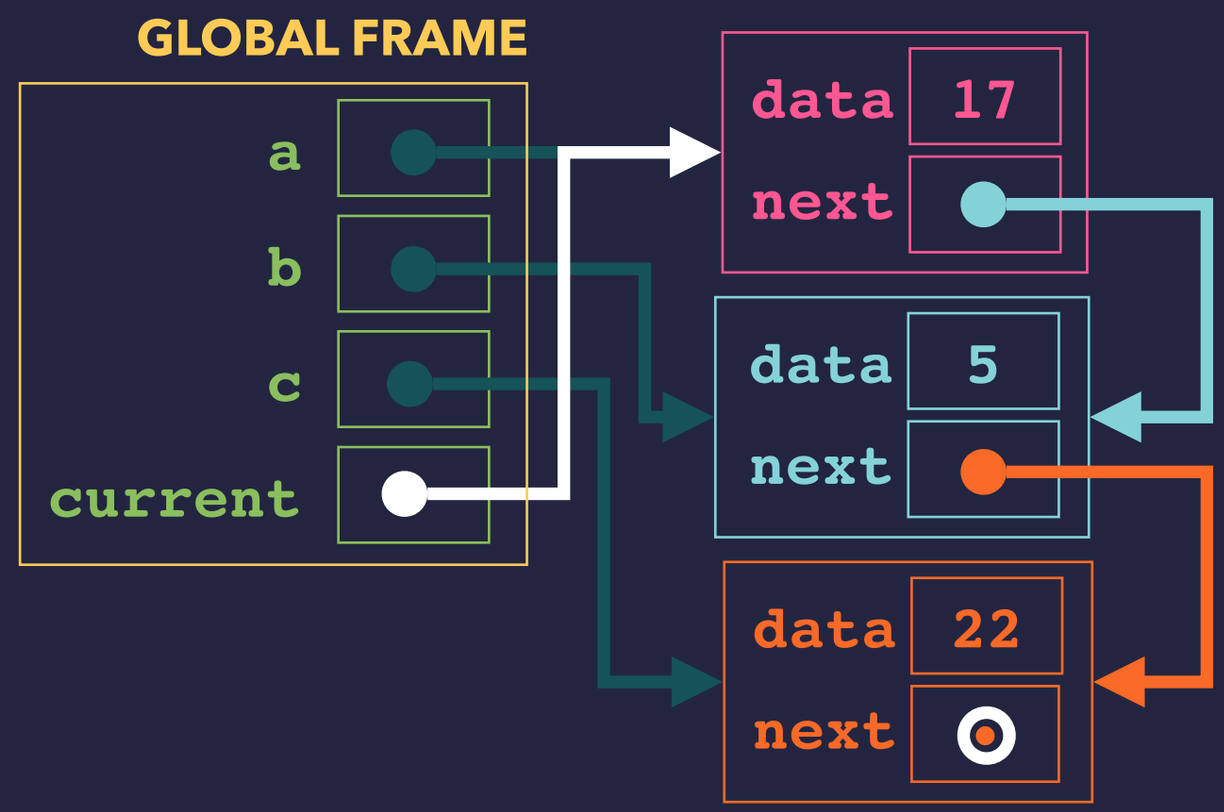
WALKING DOWN A LIST: TRAVERSAL

```
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
>>> current = a
>>> while current != None:
....    print(current.value)
....    current = current.next
....
```



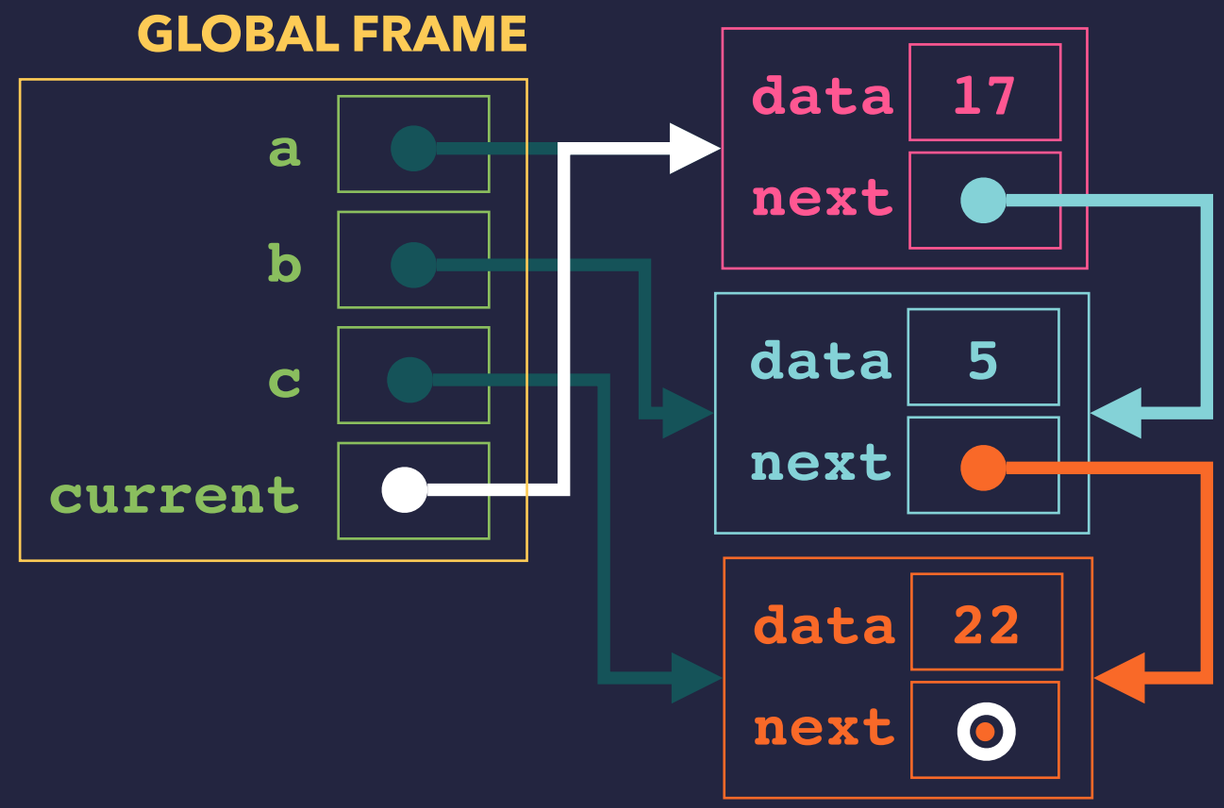
WALKING DOWN A LIST: TRAVERSAL

```
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
>>> current = a
>>> while current != None:
....    print(current.value)
....    current = current.next
....
```



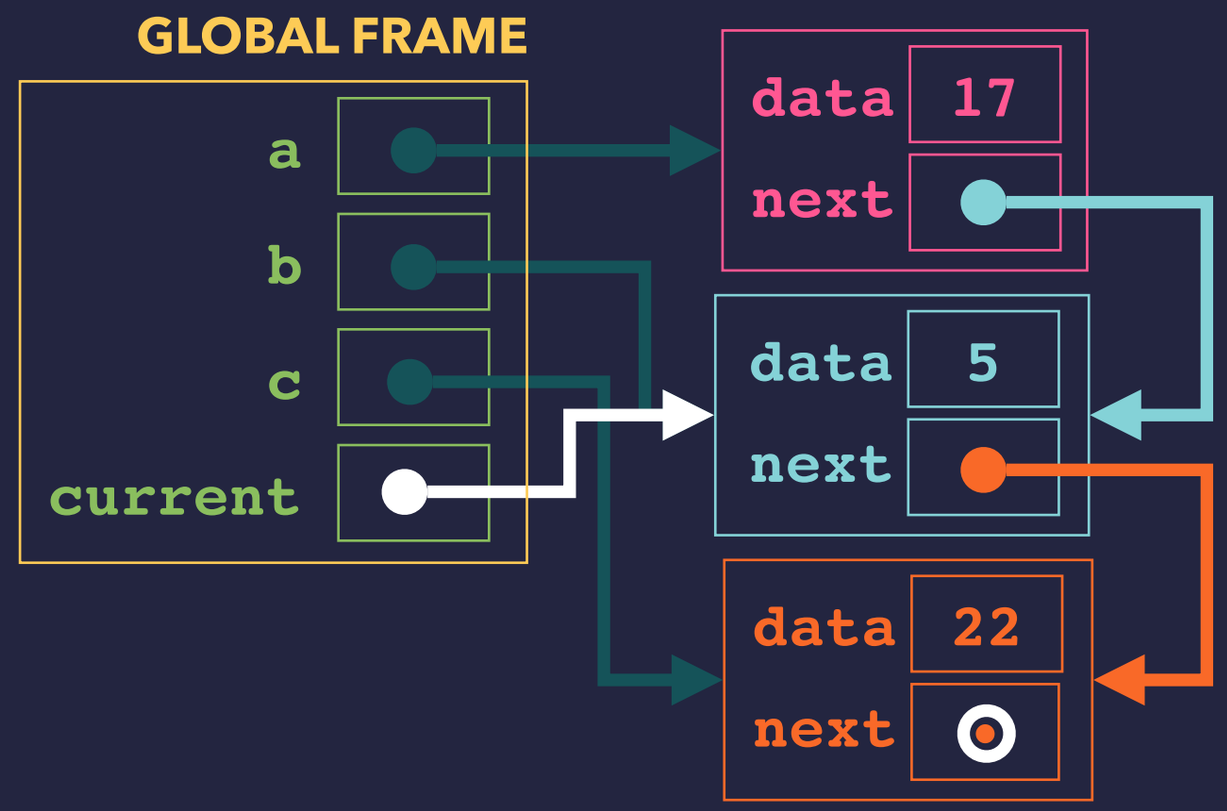
WALKING DOWN A LIST: TRAVERSAL

```
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
>>> current = a
>>> while current != None:
....     print(current.value)
....     current = current.next
....
17
|
```



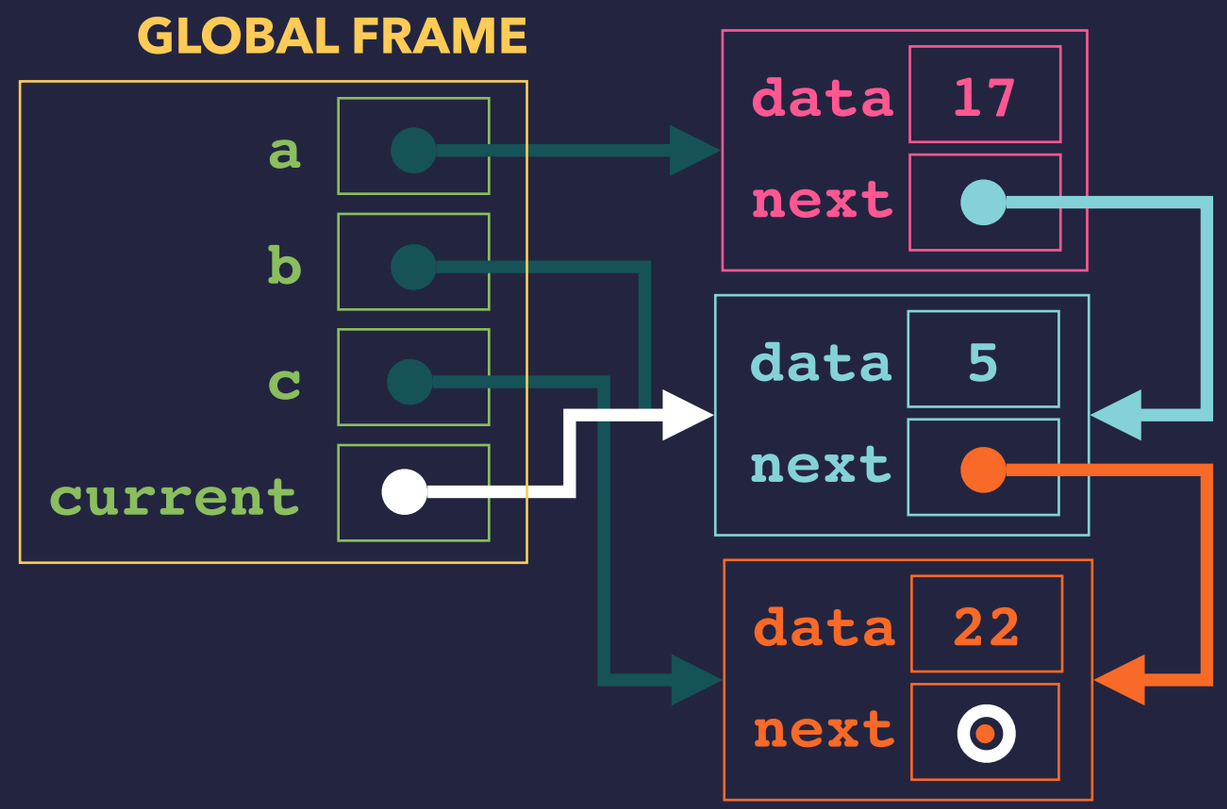
WALKING DOWN A LIST: TRAVERSAL

```
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
>>> current = a
>>> while current != None:
....     print(current.value)
....     current = current.next
....
17
|
```



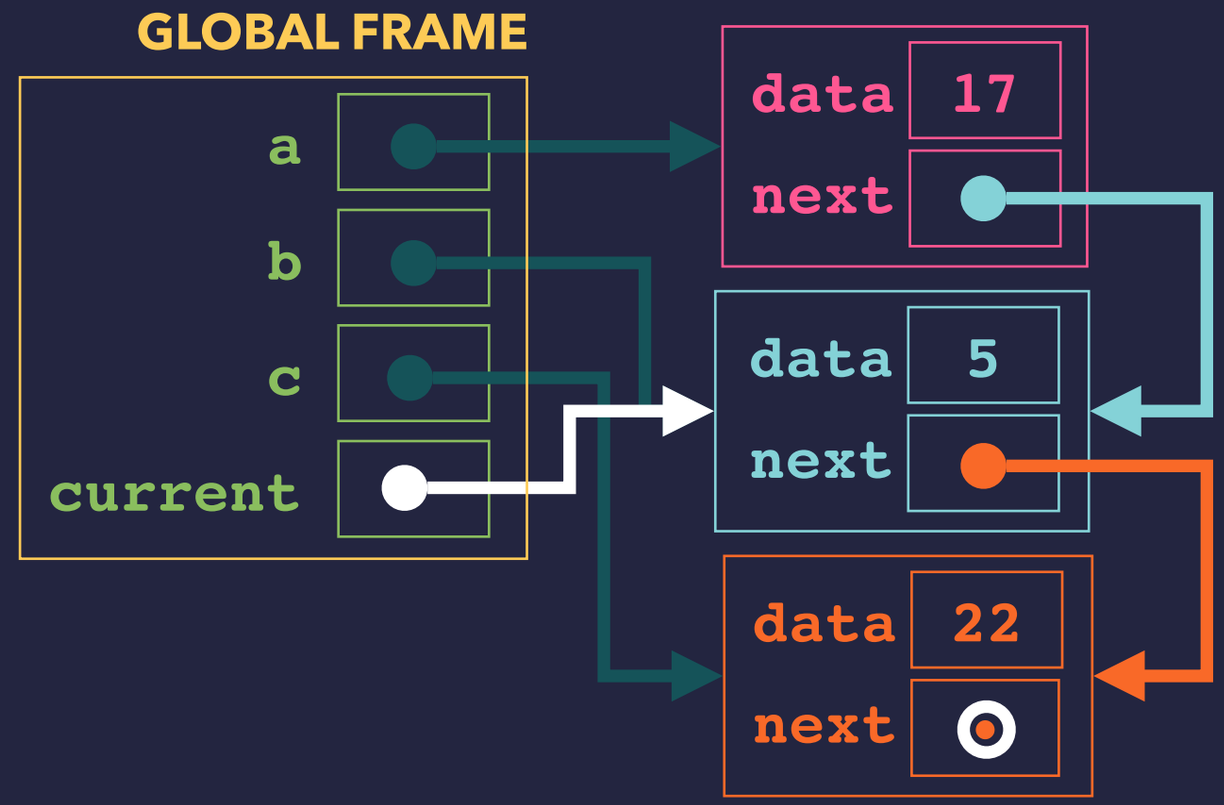
WALKING DOWN A LIST: TRAVERSAL

```
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
>>> current = a
>>> while current != None:
....    print(current.value)
....    current = current.next
....
17
|
```



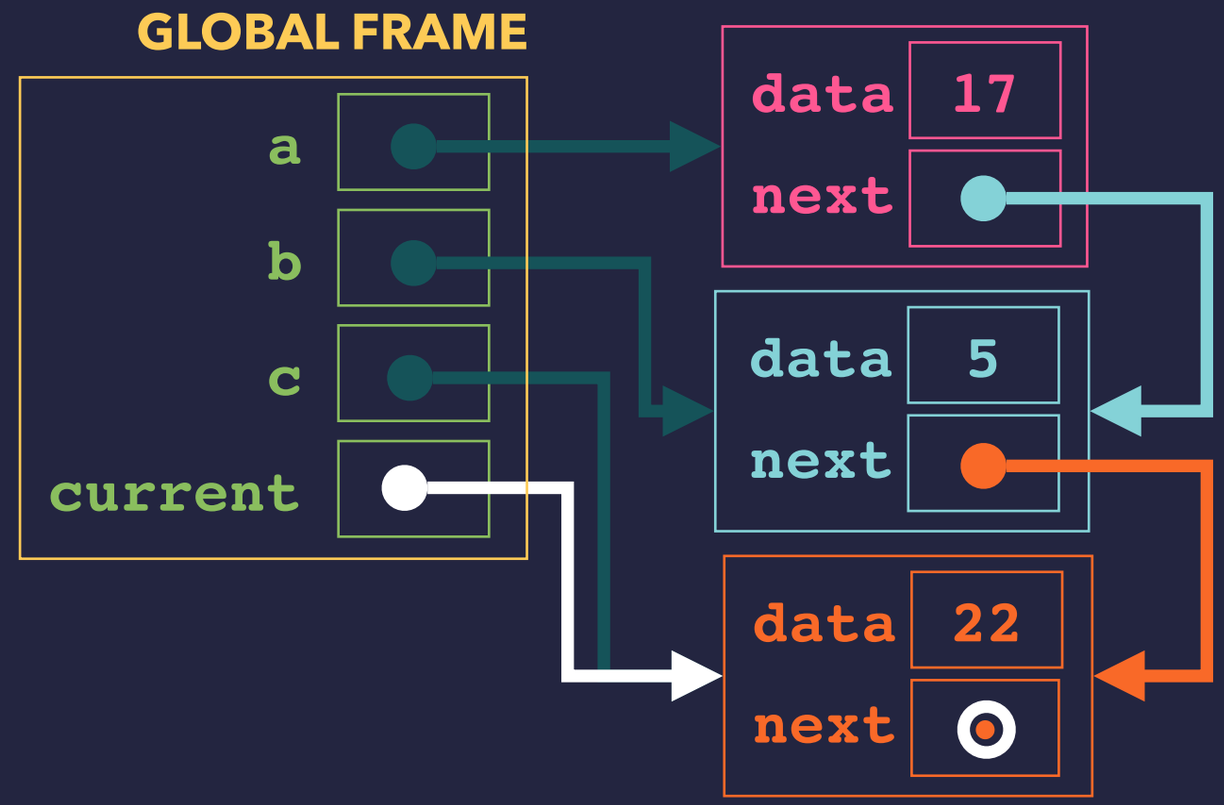
WALKING DOWN A LIST: TRAVERSAL

```
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
>>> current = a
>>> while current != None:
....    print(current.value)
....    current = current.next
....
17
5
|
```



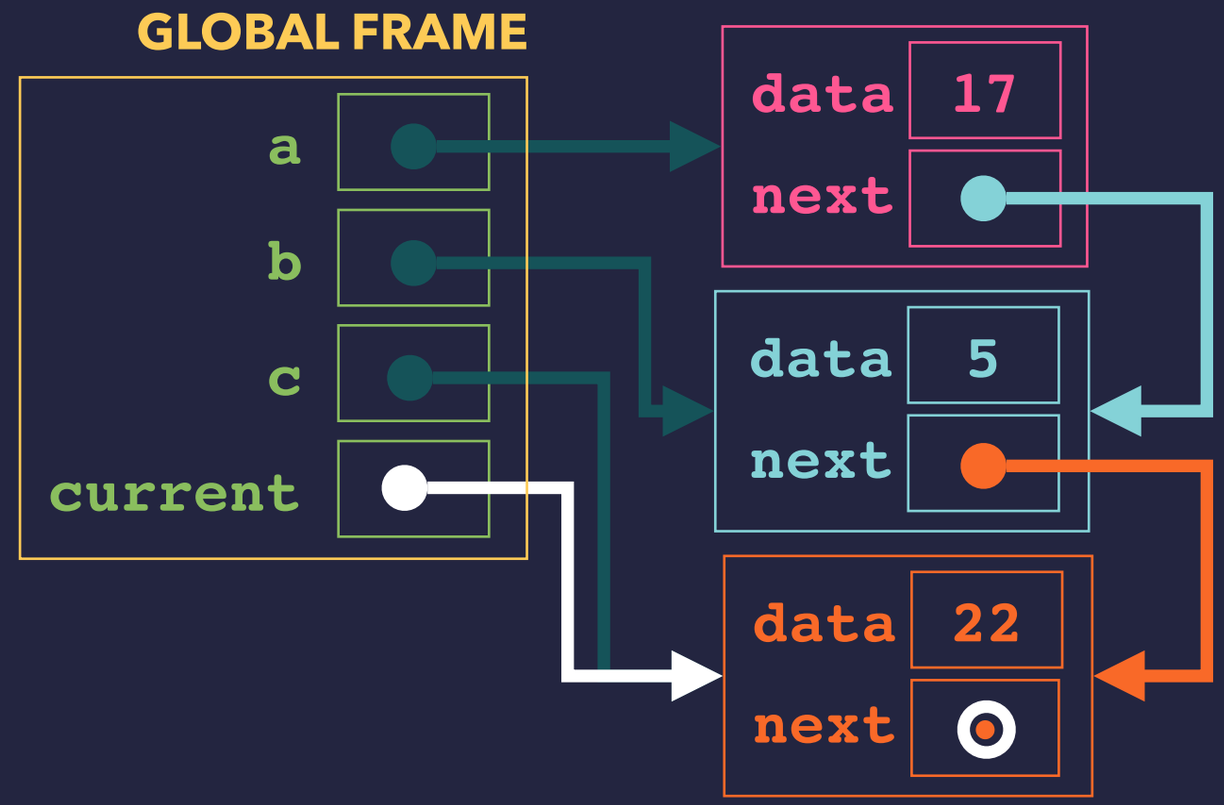
WALKING DOWN A LIST: TRAVERSAL

```
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
>>> current = a
>>> while current != None:
....     print(current.value)
....     current = current.next
....
17
5
|
```



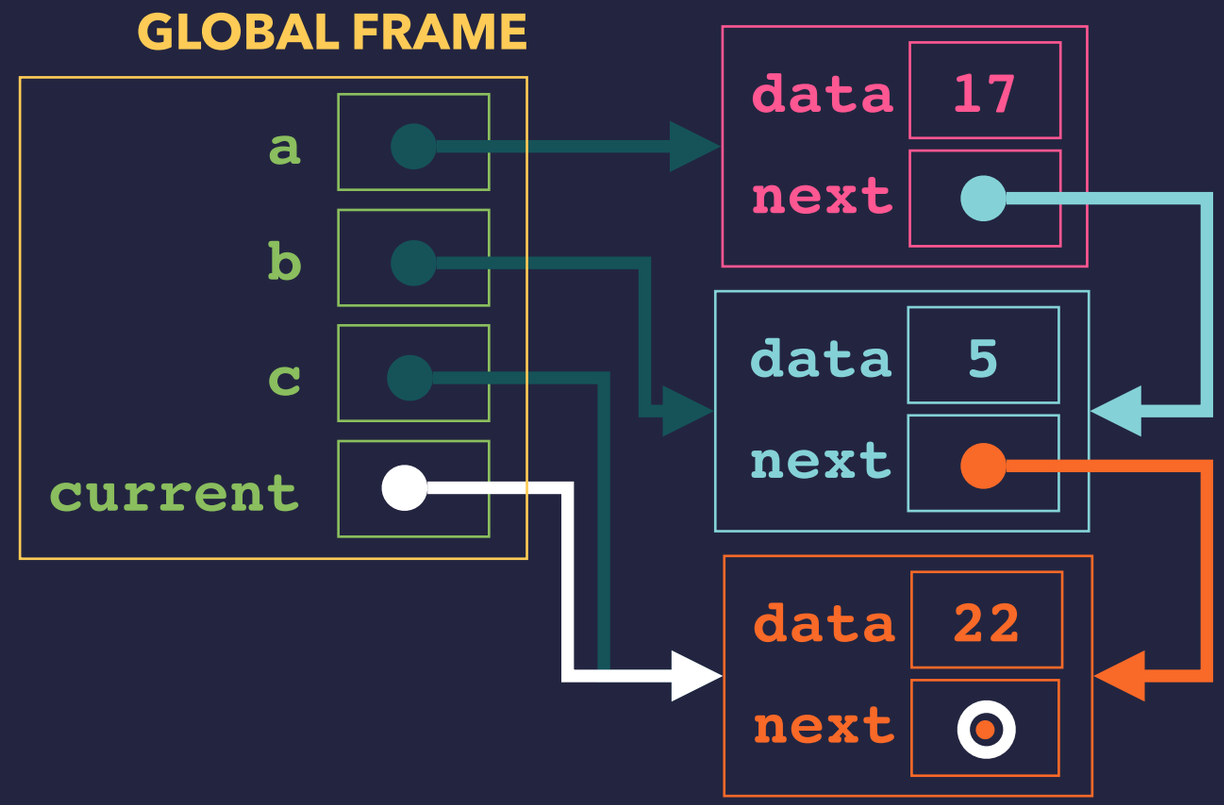
WALKING DOWN A LIST: TRAVERSAL

```
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
>>> current = a
>>> while current != None:
....    print(current.value)
....    current = current.next
....
17
5
|
```



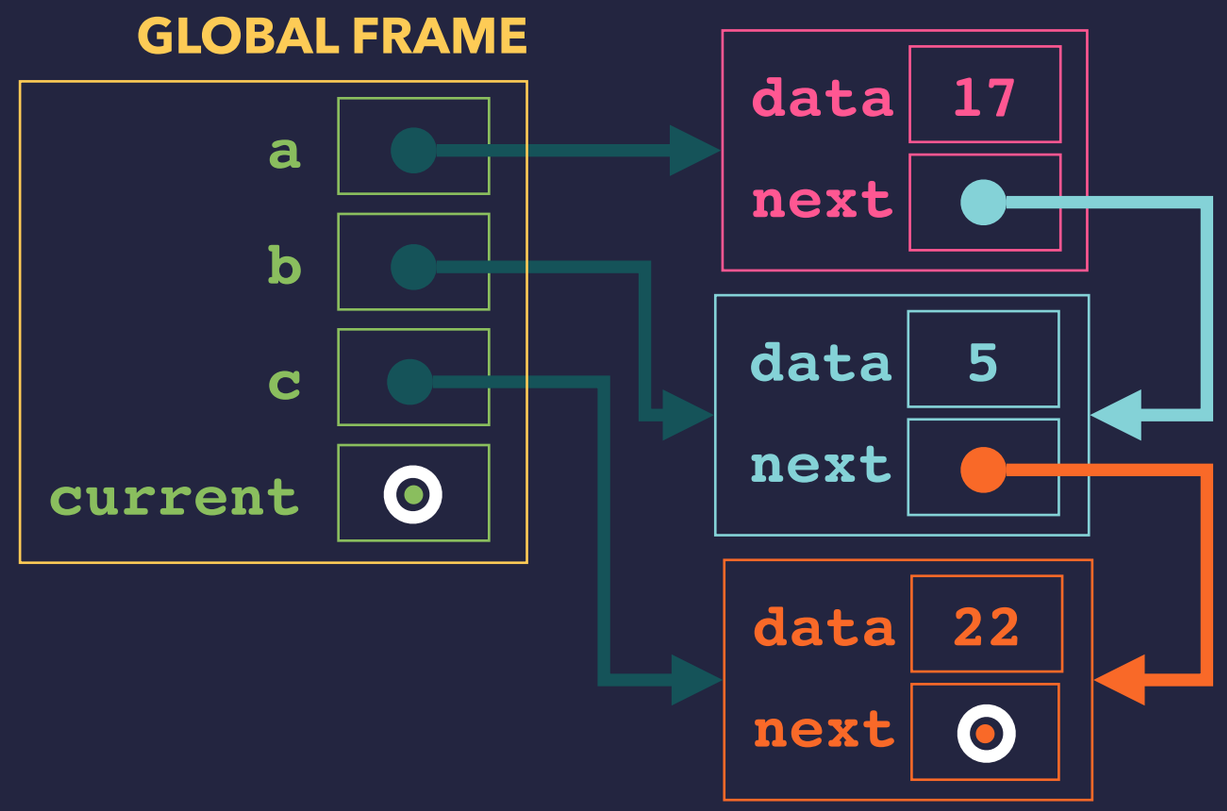
WALKING DOWN A LIST: TRAVERSAL

```
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
>>> current = a
>>> while current != None:
....     print(current.value)
....     current = current.next
....
17
5
22
```



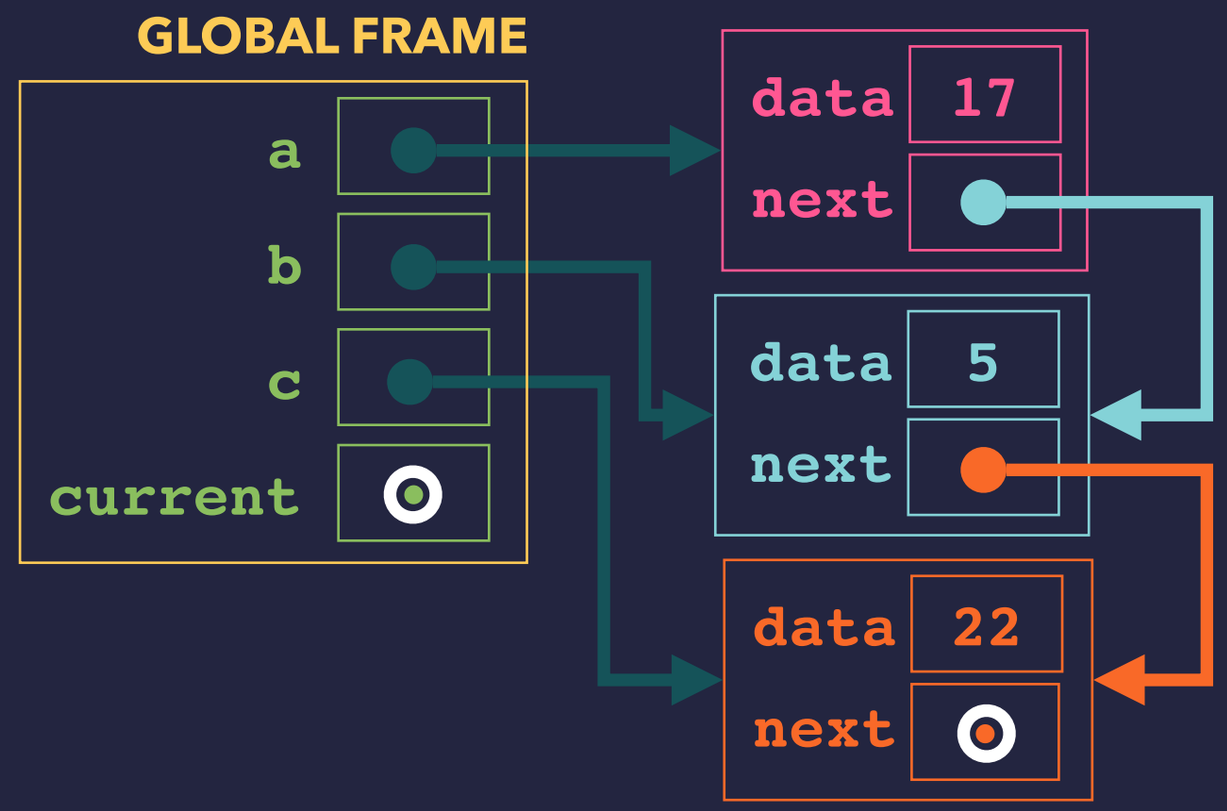
WALKING DOWN A LIST: TRAVERSAL

```
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
>>> current = a
>>> while current != None:
....     print(current.value)
....     current = current.next
....
17
5
22
```



WALKING DOWN A LIST: TRAVERSAL

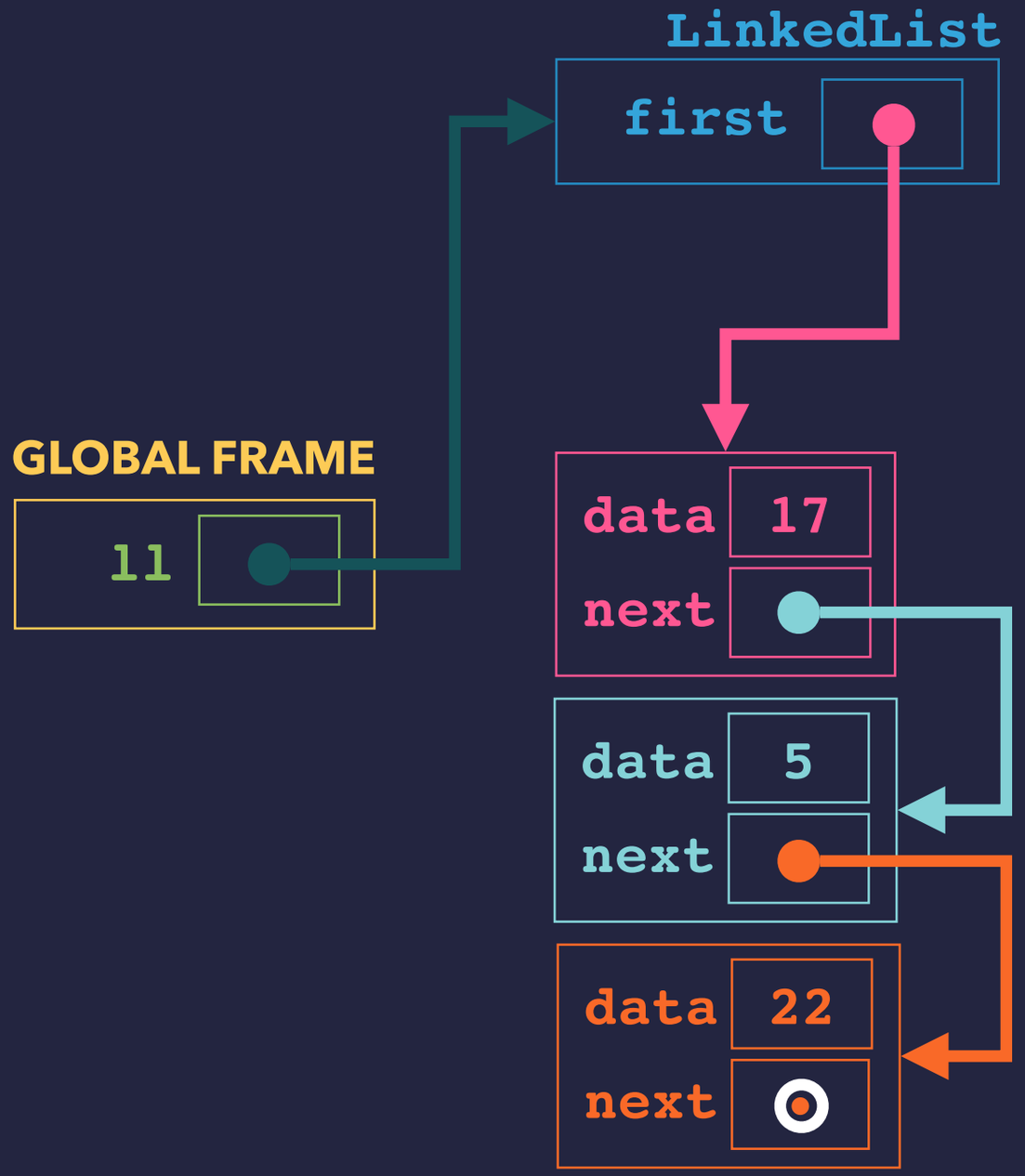
```
class Node:
    def __init__(self, value):
        self.value = value
        self.next  = None

>>> a = Node(17)
>>> b = Node(5)
>>> c = Node(22)
>>> a.next = b
>>> b.next = c
>>> current = a
>>> while current != None:
....     print(current.value)
....     current = current.next
....
17
5
22
>>>
```



A LINKED LIST CLASS

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

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

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

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

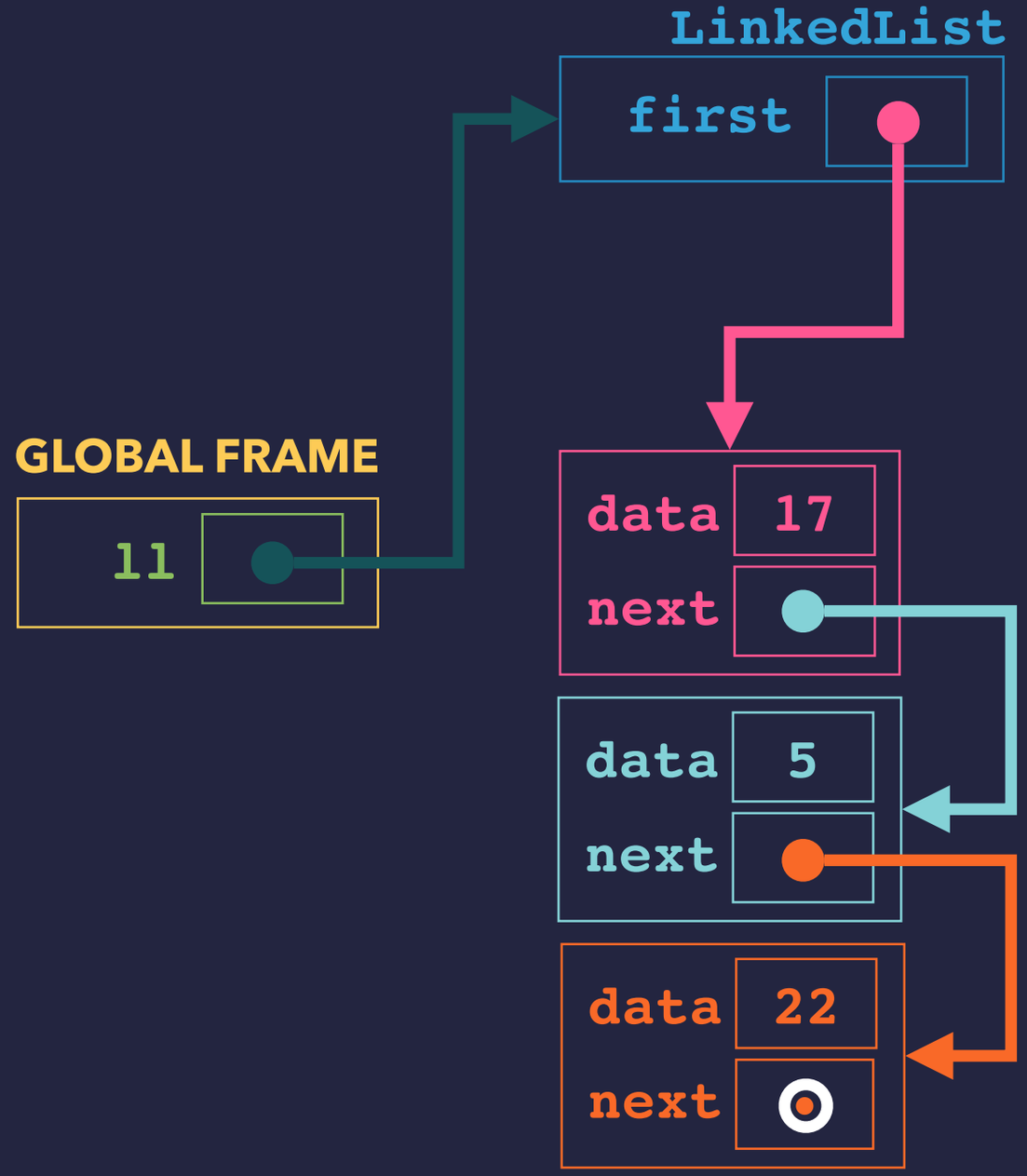


LINKED LIST APPEND

A LINKED LIST CLASS WITH APPEND

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

class LinkedList:
    ...
    def append(self, value):
        if self.first == None:
            self.first = Node(value)
        else:
            curr = self.first
            while curr.next != None:
                curr = curr.next
            curr.next = Node(value)
```

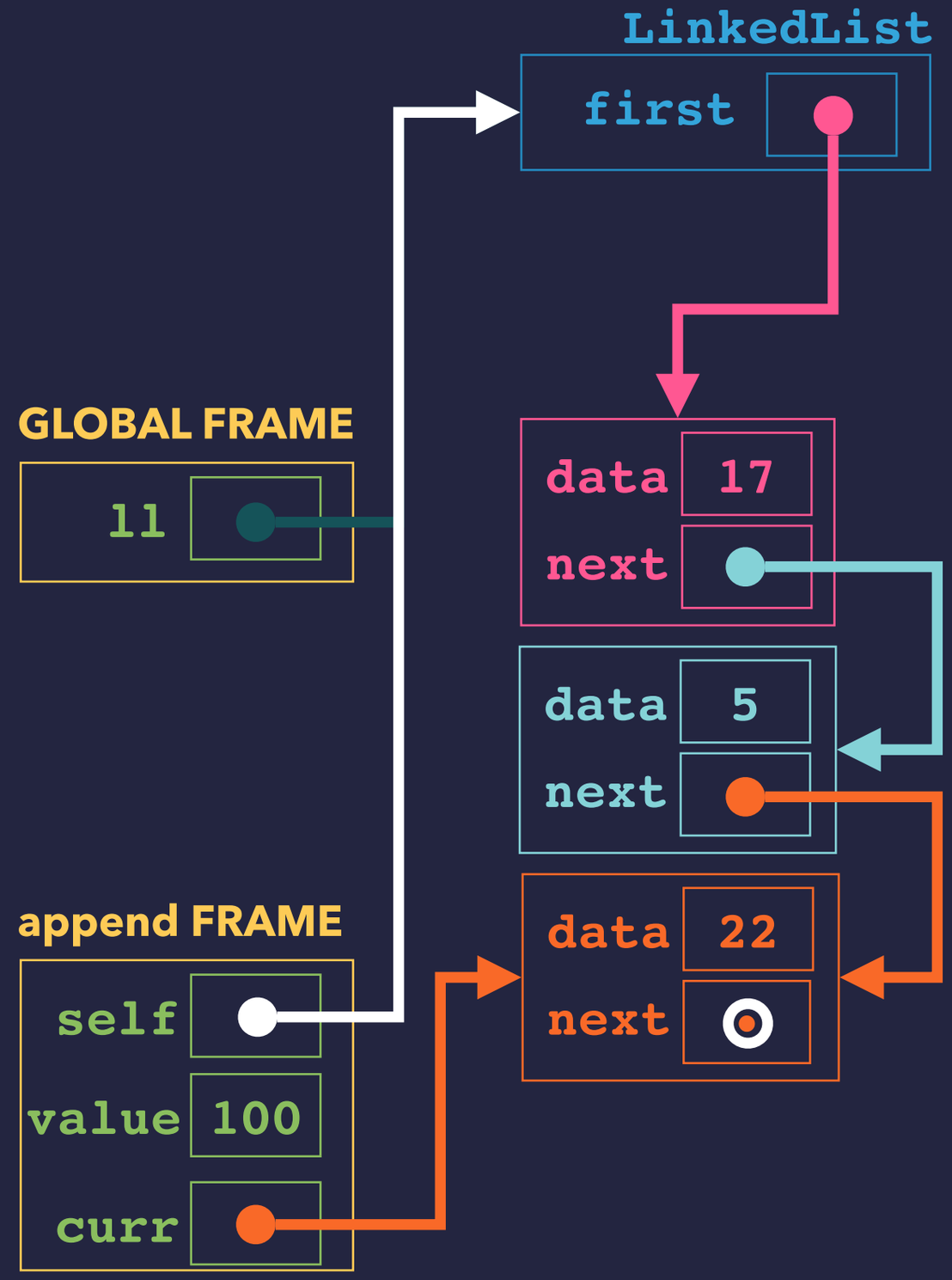
```
>>> ll = LinkedList()
>>> ll.append(17)
>>> ll.append(5)
>>> ll.append(22)
>>>
```



LINKED LIST APPEND

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

class LinkedList:
    ...
    def append(self, value):
        if self.first == None:
            self.first = Node(value)
        else:
            curr = self.first
            while curr.next != None:
                curr = curr.next
            curr.next = Node(value)
```

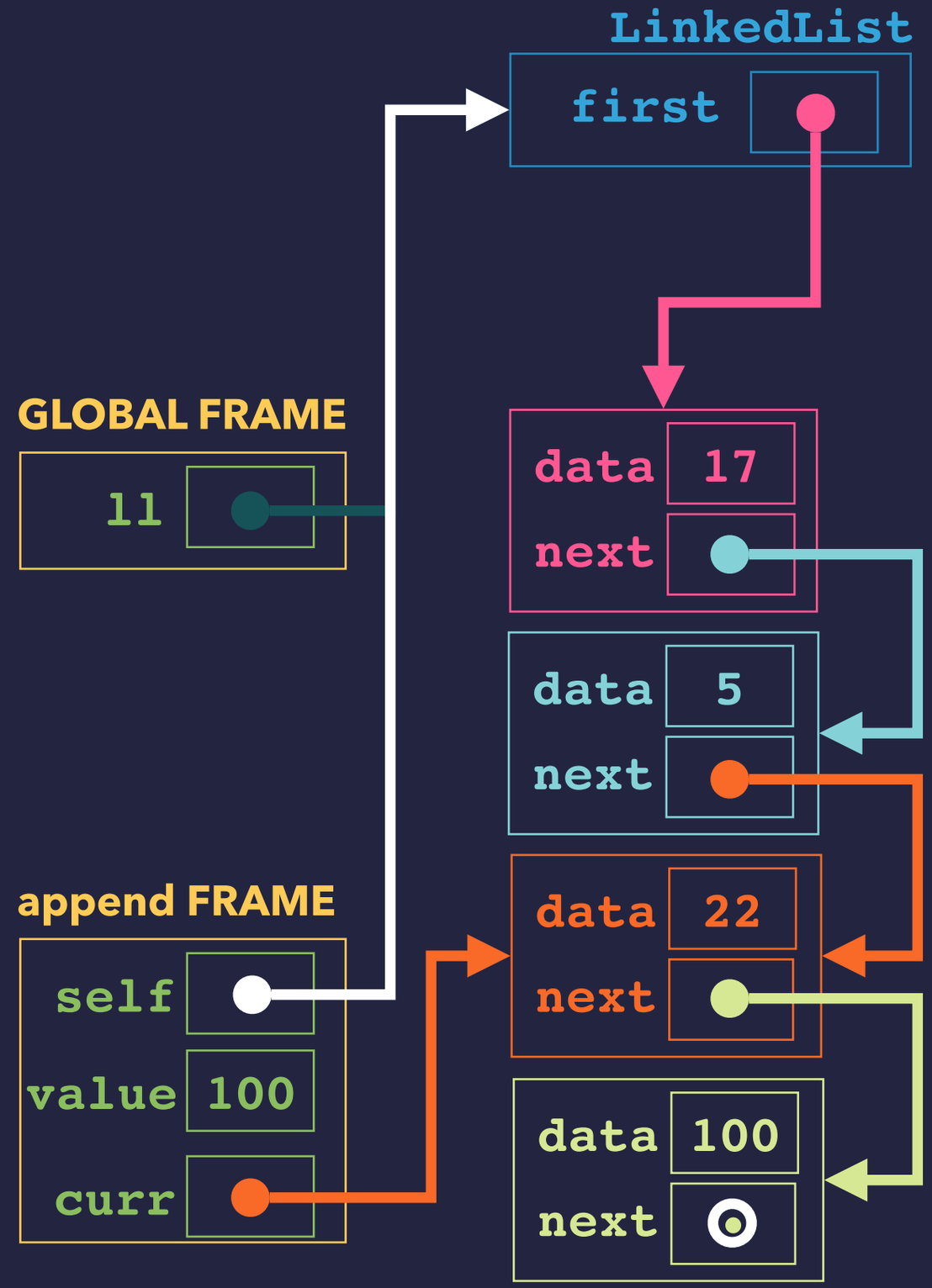
```
>>> ll = LinkedList()
>>> ll.append(17)
>>> ll.append(5)
>>> ll.append(22)
>>> ll.append(100)
```



LINKED LIST APPEND

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

class LinkedList:
    ...
    def append(self, value):
        if self.first == None:
            self.first = Node(value)
        else:
            curr = self.first
            while curr.next != None:
                curr = curr.next
            curr.next = Node(value)
```

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

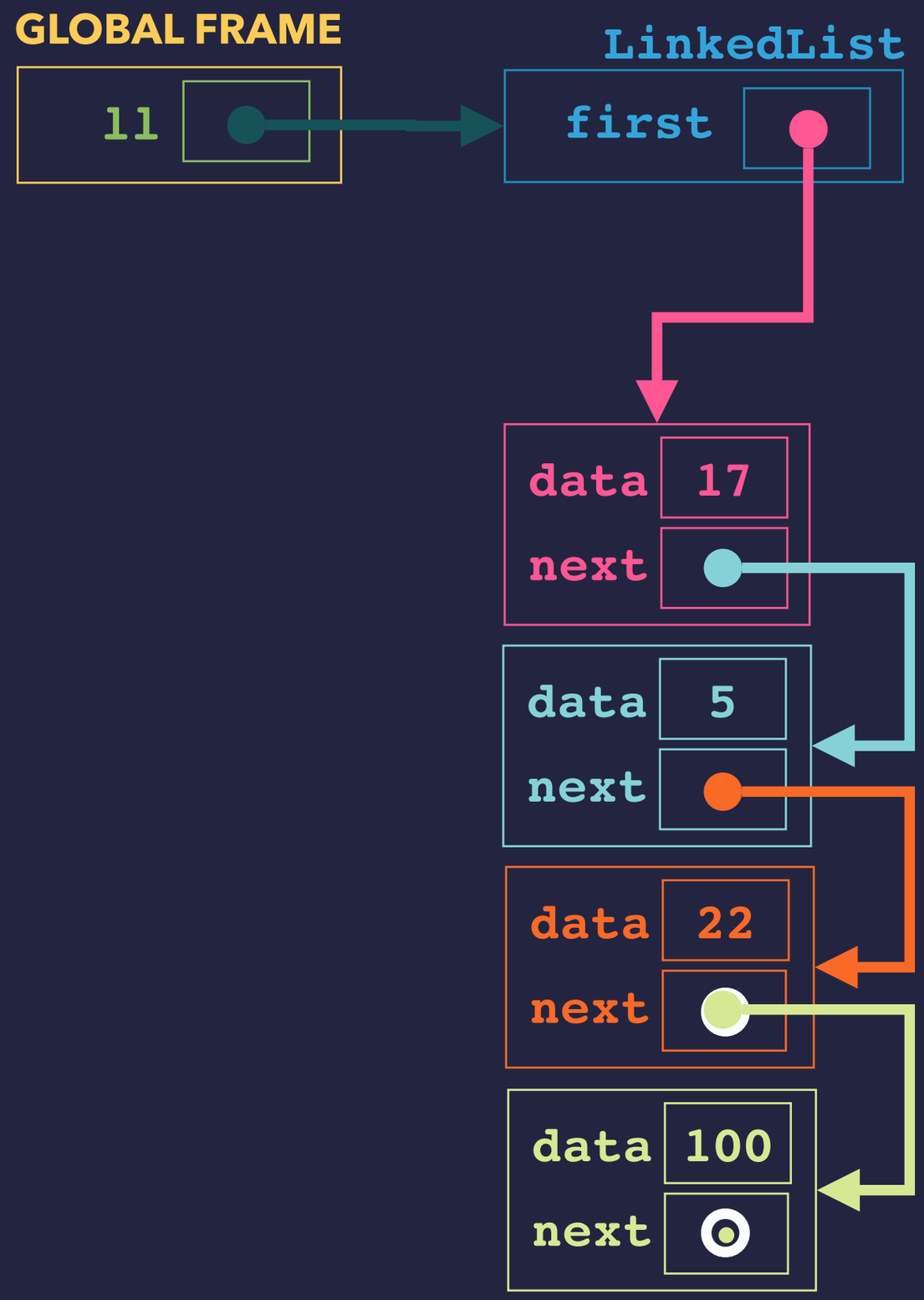


PRINTING THE VALUES IN A LINKED LIST

LINKED LIST OUTPUT

```
class LinkedList:
    ...
    def output(self):
        curr = self.first
        while curr != None:
            print(curr.value)
            curr = curr.next
```

```
>>> ll.output()
17
5
22
100
>>>
```

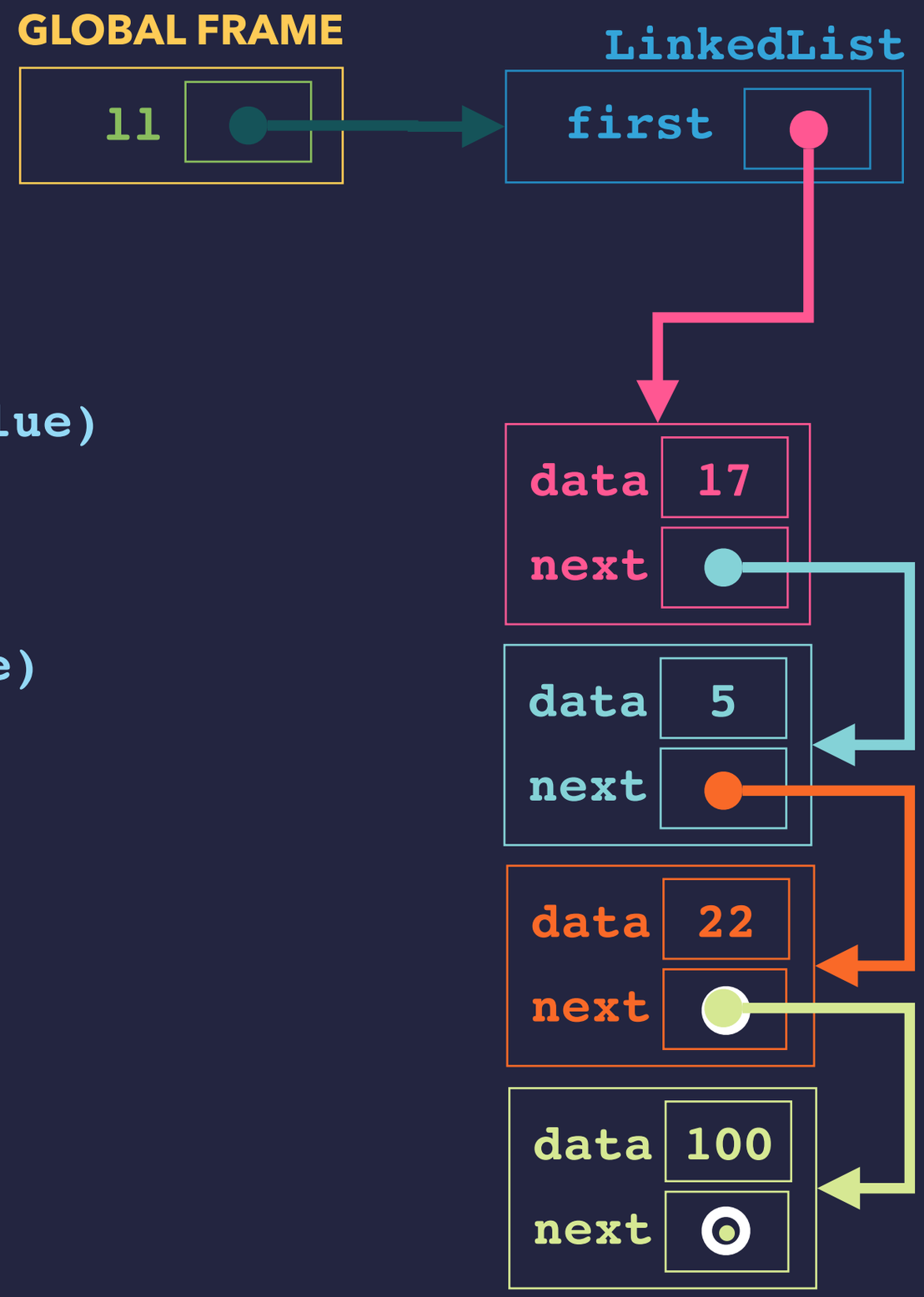


STRING REPRESENTING A LINKED LIST

LINKEDLIST ASSTRING

```
class LinkedList:
    ...
    def asString(self):
        s = "<"
        if self.first != None:
            s += str(self.first.value)
            curr = self.first.next
            while curr != None:
                s += ", "
                s += str(curr.value)
                curr = curr.next
        s += ">"
        return s
```

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



CHECKING WHETHER A LIST CONTAINS A VALUE

LINKED LIST CONTAINS CHECK

```
class LinkedList:
    ...
    def contains(self, value):
        curr = self.first
        while curr != None:
            if curr.value == value:
                return True
            curr = curr.next
        return False
```

```
>>> ll.contains(5)
True
>>> ll.contains(88)
False
>>>
```