Lecture 10

By Urooj Ainuddin

CS-218 Data Structures and Algorithms

Linked Structures

Book 1 Chapter 6

Things we know about arrays and lists…

An array provides easy and direct access to the individual elements, but it is limited in functionality and has a fixed size.

The Python list is implemented using the C array.

The Python list has a larger set of operations than an array and can adjust in size as items are added or removed.

The array and Python list can be used to implement many different abstract data types.

Binary search can be used with both structures when items are stored in sorted order to allow for quick searches.

Disadvantages of arrays and lists

Insertion and deletion

Insertion and deletion require items to be shifted to make room or close a gap, which consumes time.

Changing size

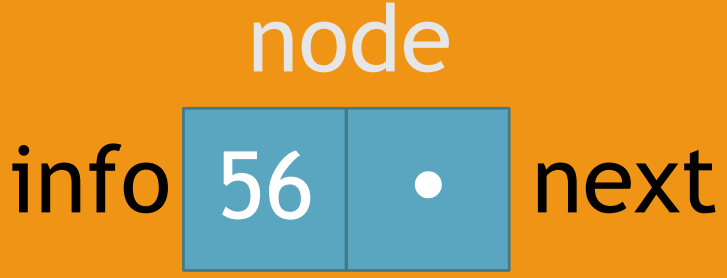
In the Python list, expansion requires the creation of a new larger array into which the elements of the original array must be copied. This take considerable time.

Memory allocation

Array elements are placed contiguously in memory, so we must find a block of memory large enough to hold the entire array. For large arrays, this can be difficult. This is especially true for a Python list since expansion requires even larger blocks of memory.

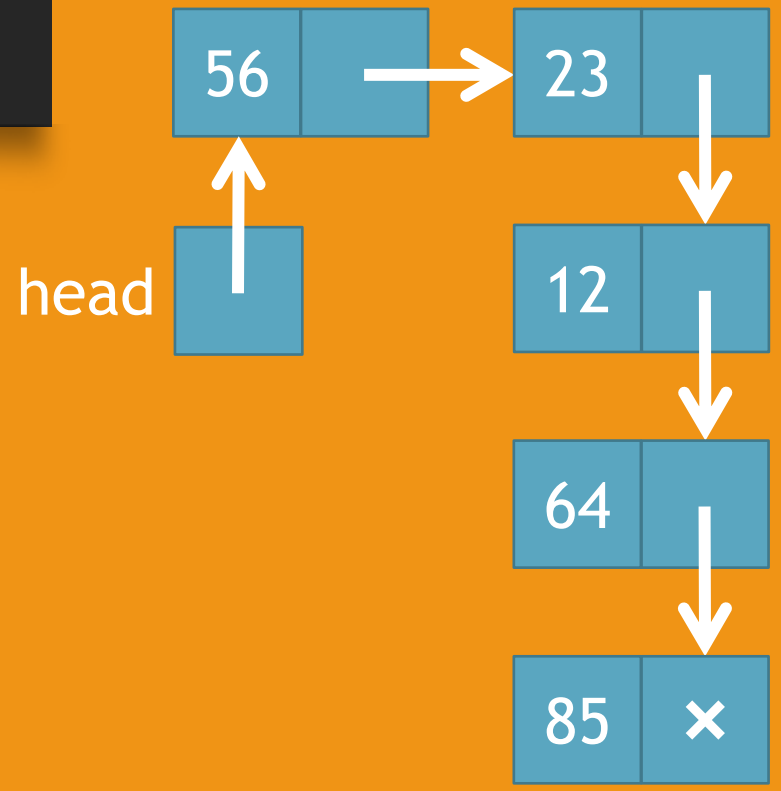
The singly linked list (SLL)

- The singly linked list is a structure composed of **nodes**.
- Each node has an **info** field and a **next** field.
- The info field carries the data item that needs to be stored in the data structure.



The singly linked list (SLL)

- The singly linked list is a structure composed of **nodes**.
- Each node has an **info** field and a **next** field.
- The info field carries the data item that needs to be stored in the data structure.
- The next field contains a pointer to the next node of the SLL.
- The entire data structure can be accessed via a **head** pointer.
- A pointer is the address of an object.
- The linked list data structure can be used to store a collection in linear order.



A node
A node
A node

56 •

A pointer points to the whole object, not the fields or elements inside it.

The singlylinkedlist.py file - the class ListNode

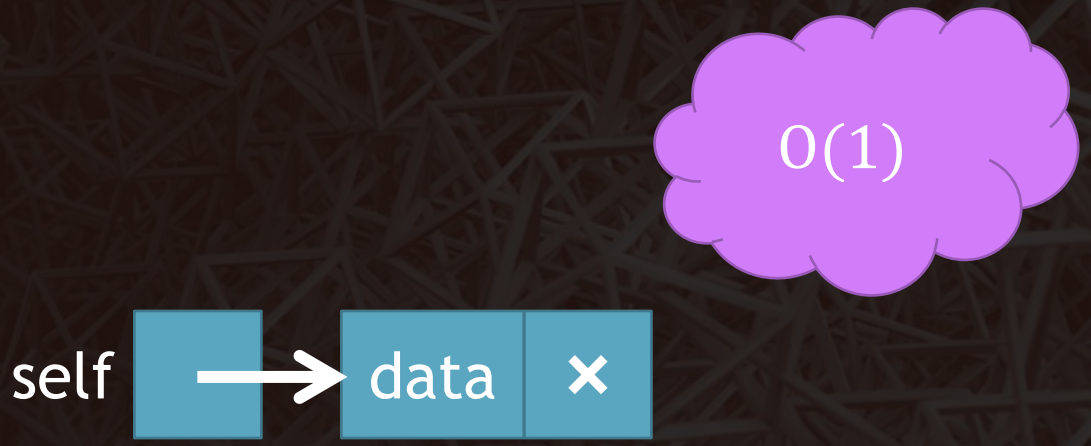
```
class ListNode:
    def __init__(self,data):
        self.data=data
        self.next=None
```

The singlylinkedlist.py file - the class ListNode

```
class ListNode:
    def __init__(self,data):
        self.data=data
        self.next=None
```

The singlylinkedlist.py file - the class ListNode

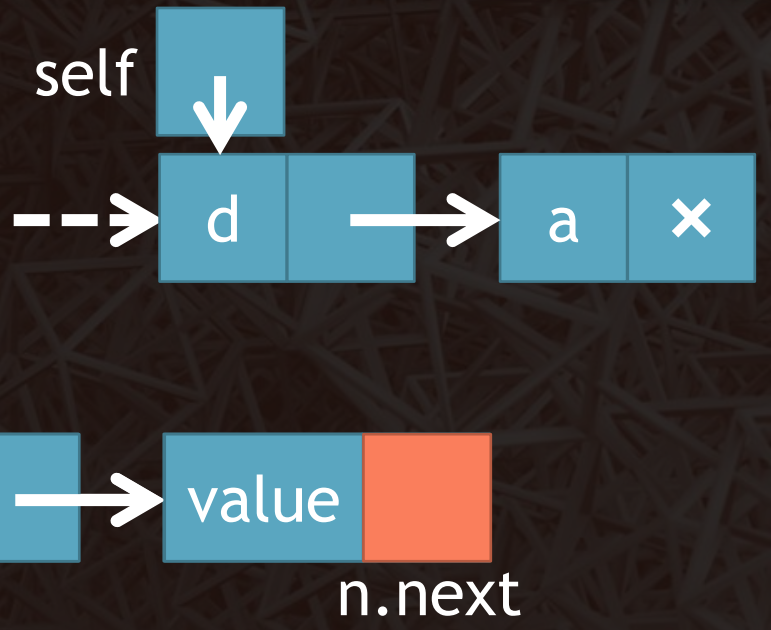
```
class ListNode:
    def __init__(self,data):
        self.data=data
        self.next=None
```



The singlylinkedlist.py file - insert

This code inserts a node after the node pointed to by self.

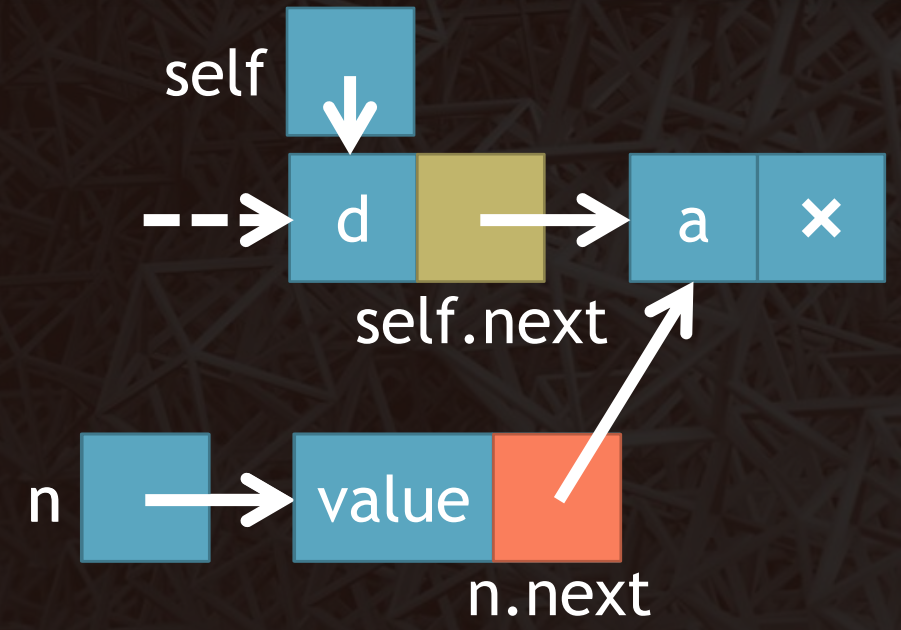
```
def insert(self, value):
        n = ListNode(value)
        n.next=self.next
        self.next=n
```



The singlylinkedlist.py file - insert

This code inserts a node after the node pointed to by self.

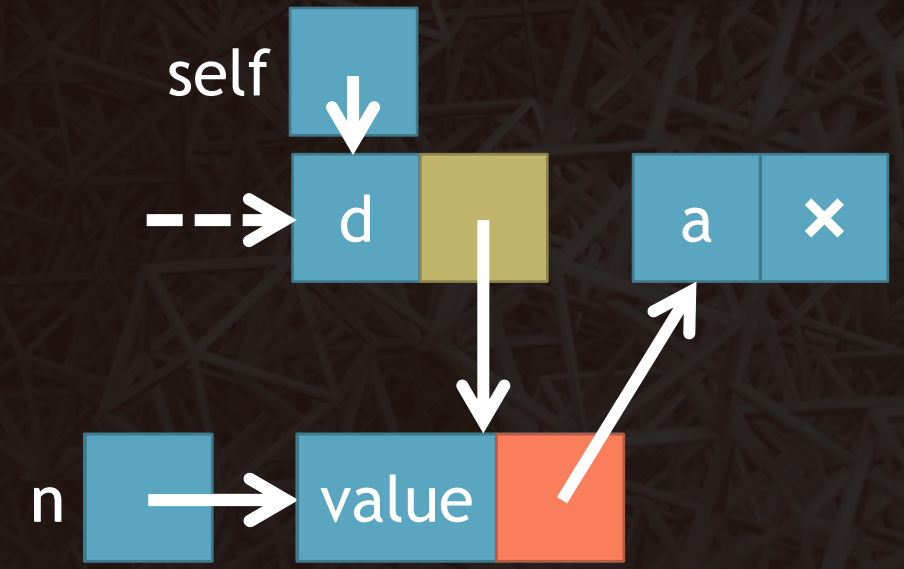
```
def insert(self, value):
        n = ListNode(value)
        n.next=self.next
        self.next=n
```



The singlylinkedlist.py file - insert

This code inserts a node after the node pointed to by self.

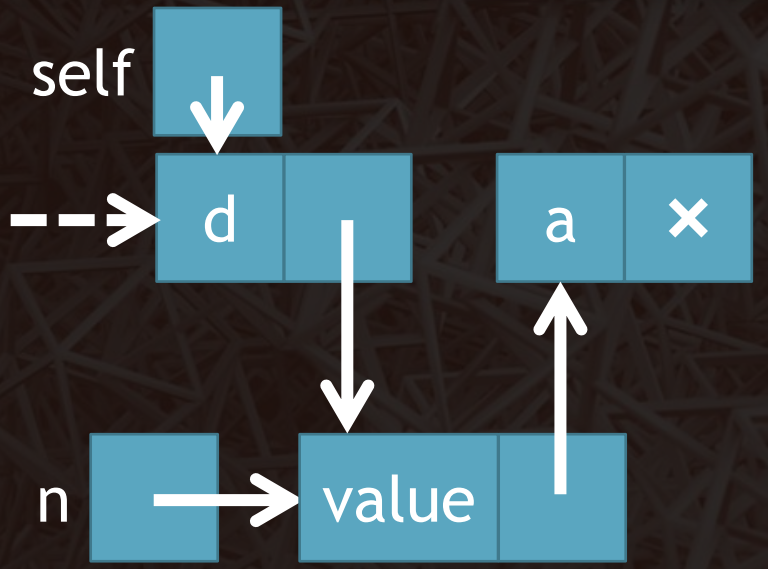
```
def insert(self, value):
        n = ListNode(value)
        n.next=self.next
        self.next=n
```



The singlylinkedlist.py file - insert

This code inserts a node after the node pointed to by self.

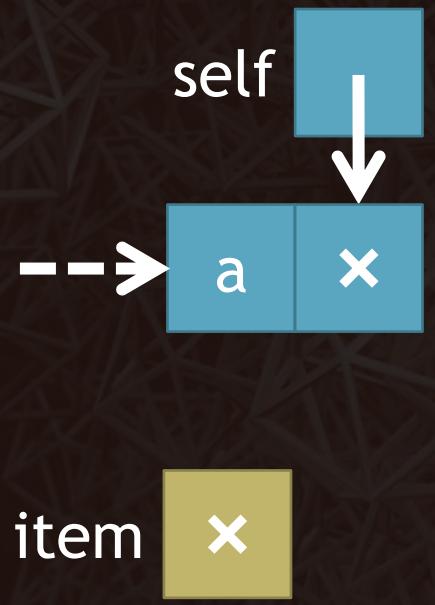
```
def insert(self, value):
        n = ListNode(value)
        n.next=self.next
        self.next=n
```



The singlylinkedlist.py file - delete

This code deletes a node after the node pointed to by self.

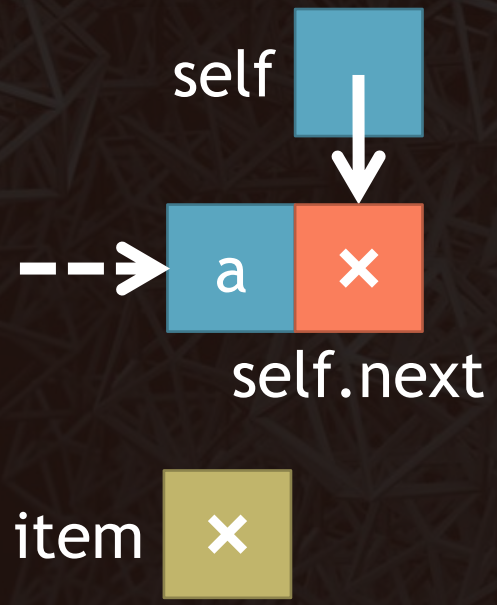
```
def delete(self):
        item=None
        if self.next is not None:
            tmp=self.next
            item=tmp.data
            self.next=tmp.next
        return item
```



The singlylinkedlist.py file - delete

This code deletes a node after the node pointed to by self.

```
def delete(self):
        item=None
        if self.next is not None:
            tmp=self.next
            item=tmp.data
            self.next=tmp.next
        return item
```



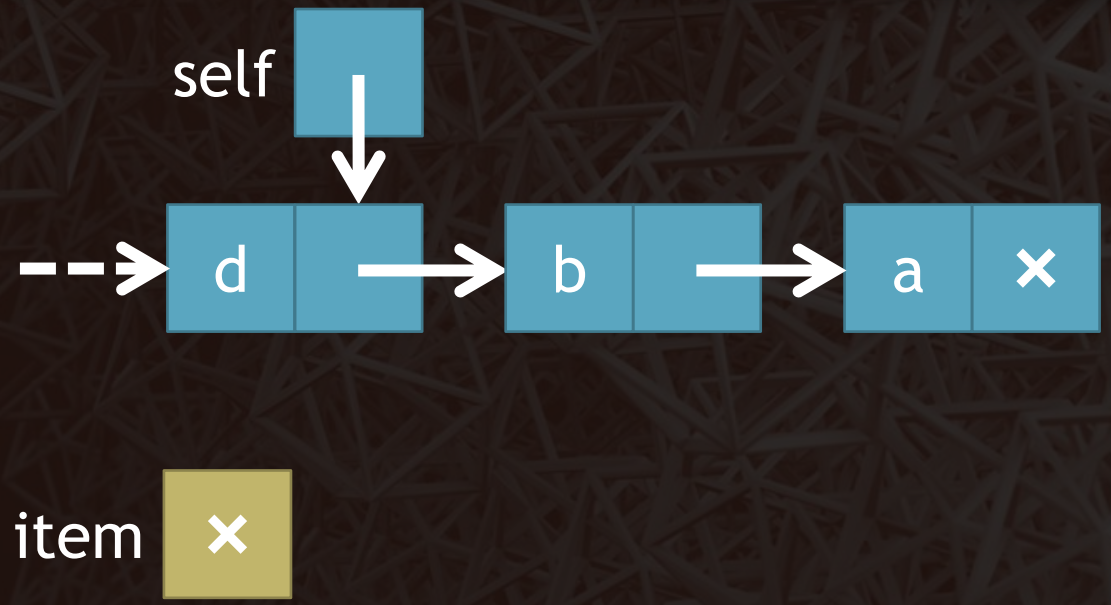
O(1)

Best case

The singlylinkedlist.py file - delete

This code deletes a node after the node pointed to by self.

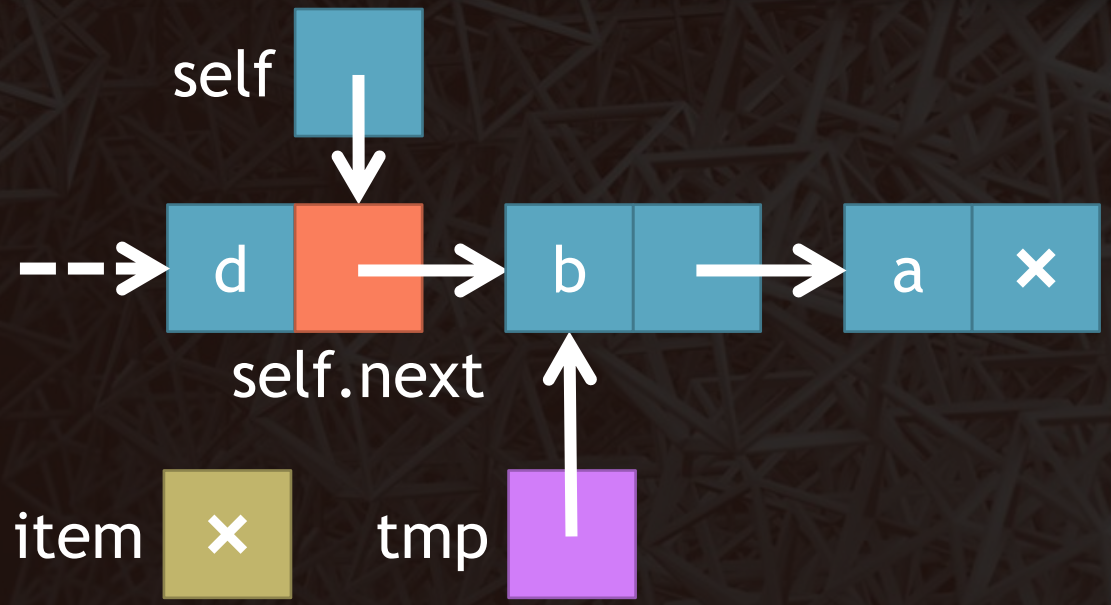
```
def delete(self):
        item=None
        if self.next is not None:
            tmp=self.next
            item=tmp.data
            self.next=tmp.next
        return item
```



The singlylinkedlist.py file - delete

This code deletes a node after the node pointed to by self.

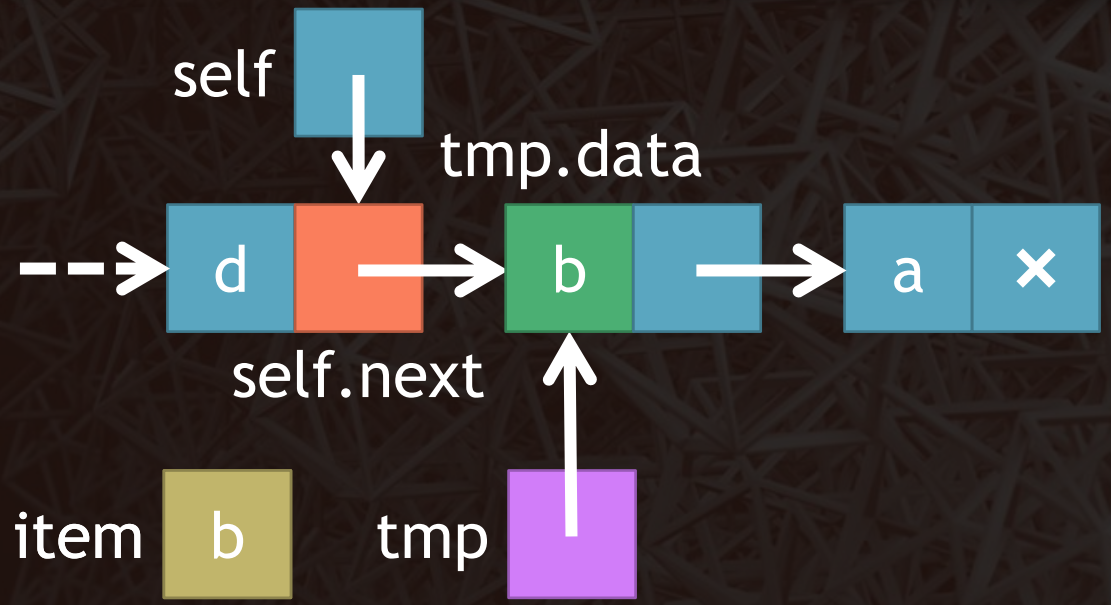
```
def delete(self):
        item=None
        if self.next is not None:
            tmp=self.next
            item=tmp.data
            self.next=tmp.next
        return item
```



The singlylinkedlist.py file - delete

This code deletes a node after the node pointed to by self.

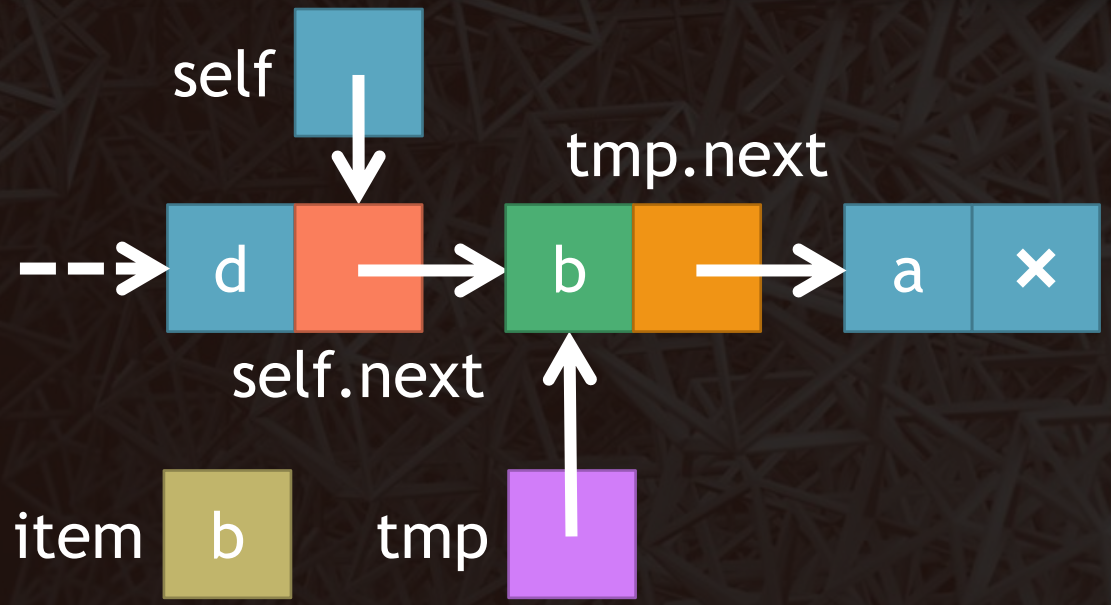
```
def delete(self):
        item=None
        if self.next is not None:
            tmp=self.next
            item=tmp.data
            self.next=tmp.next
        return item
```



The singlylinkedlist.py file - delete

This code deletes a node after the node pointed to by self.

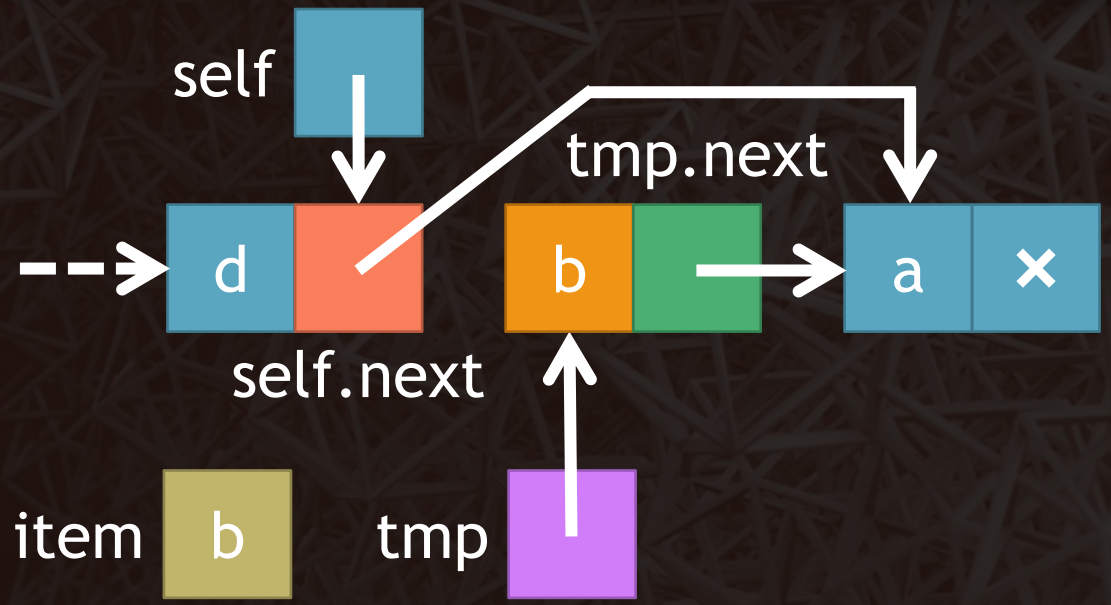
```
def delete(self):
        item=None
        if self.next is not None:
            tmp=self.next
            item=tmp.data
            self.next=tmp.next
        return item
```



The singlylinkedlist.py file - delete

This code deletes a node after the node pointed to by self.

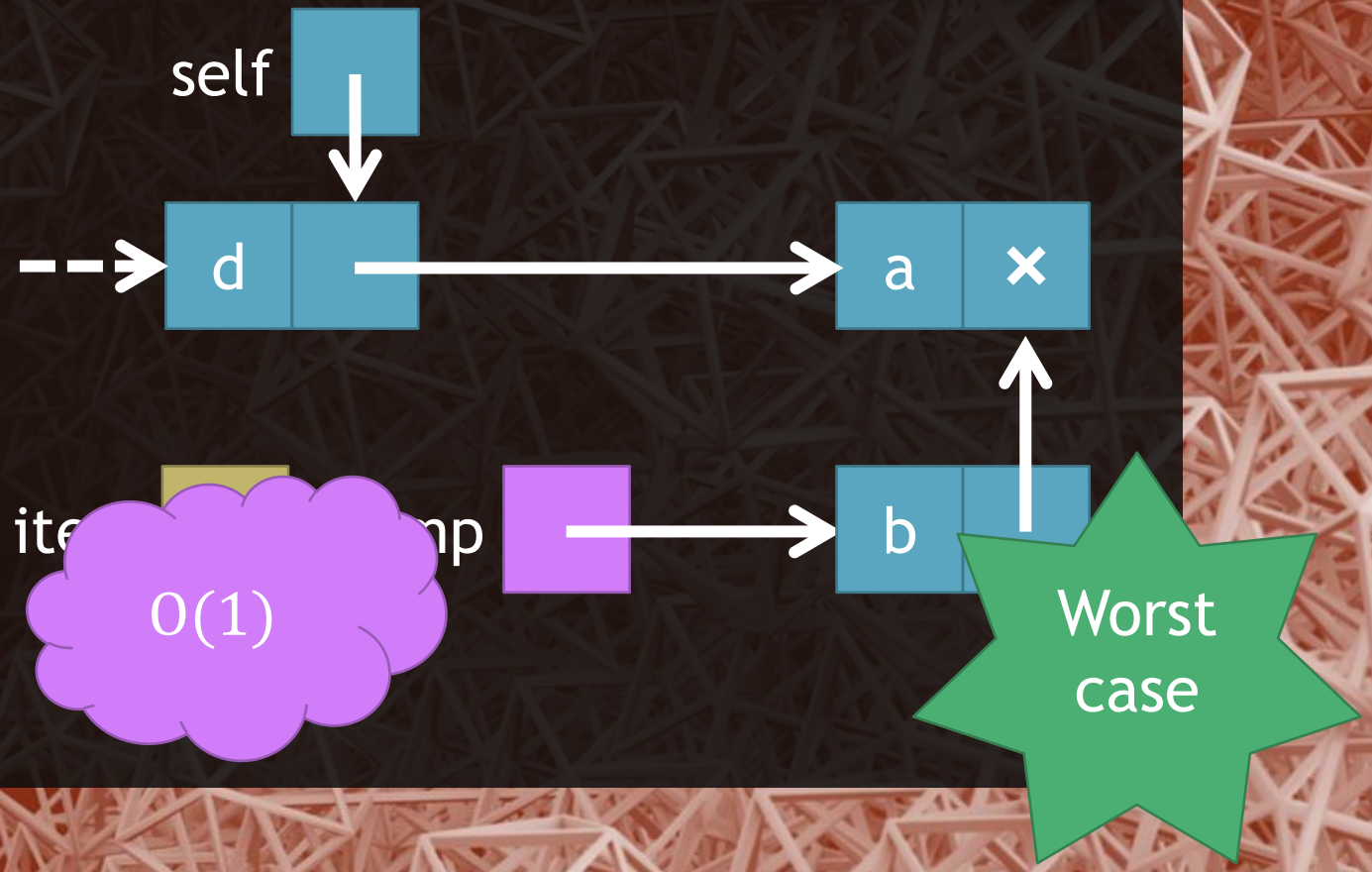
```
def delete(self):
        item=None
        if self.next is not None:
            tmp=self.next
            item=tmp.data
            self.next=tmp.next
        return item
```



The singlylinkedlist.py file - delete

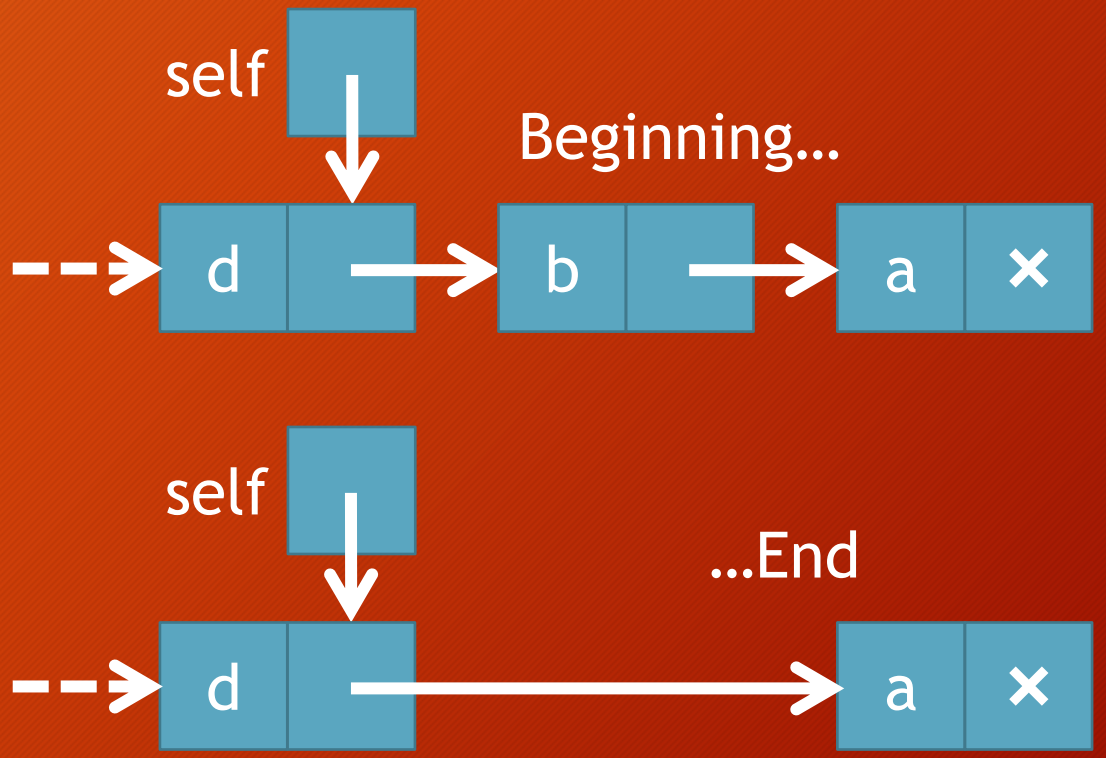
This code deletes a node after the node pointed to by self.

```
def delete(self):
        item=None
        if self.next is not None:
            tmp=self.next
            item=tmp.data
            self.next=tmp.next
        return item
```



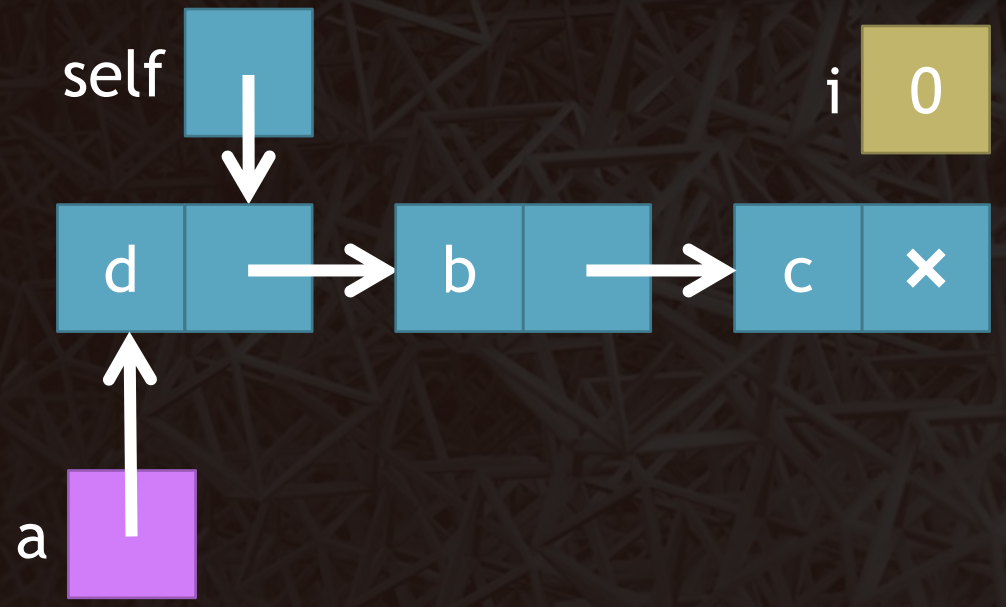
The singlylinkedlist.py file - delete

- This code deletes a node after the node pointed to by self.
- For deletion, we require the pointer of the node **before** the node that must be deleted.



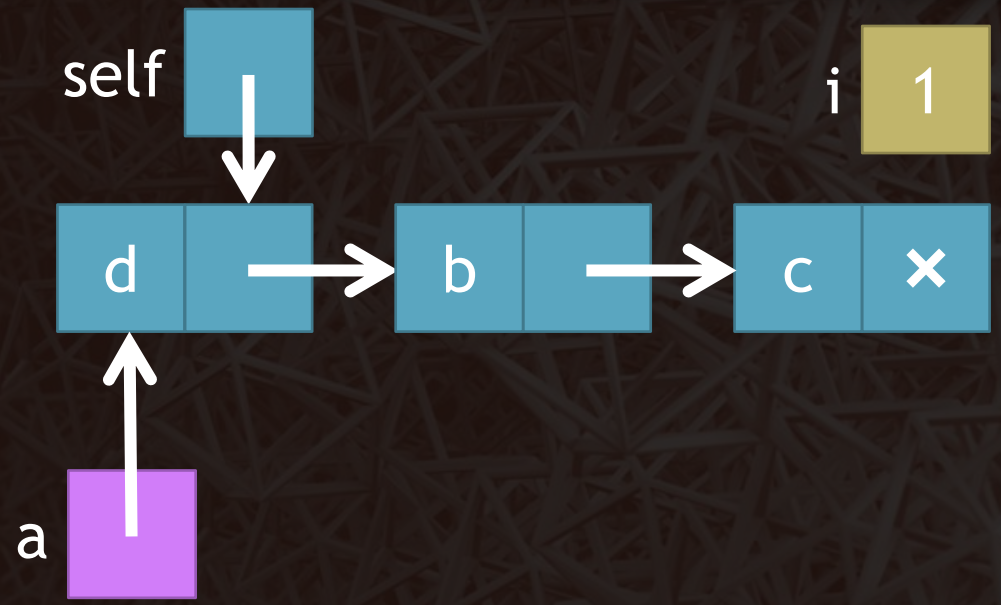
The singlylinkedlist.py file - len

```
def __len__(self):
        a = self
        i = 0
        while a is not None:
            i += 1
            a = a.next
        return i
```



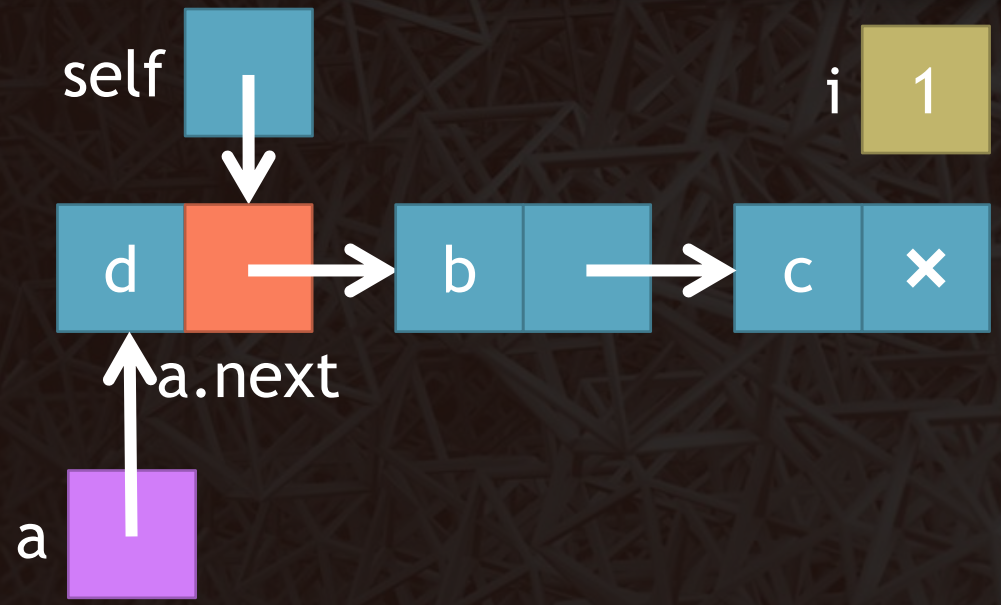
The singlylinkedlist.py file - len

```
def __len__(self):
        a = self
        i = 0
        while a is not None:
            i += 1
            a = a.next
        return i
```



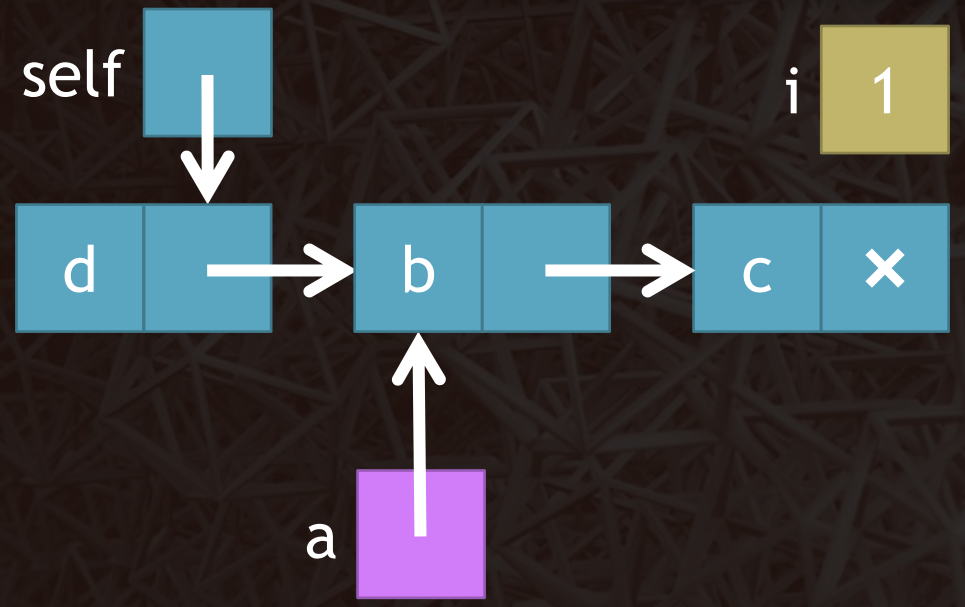
The singlylinkedlist.py file - len

```
def __len__(self):
        a = self
        i = 0
        while a is not None:
            i += 1
            a = a.next
        return i
```



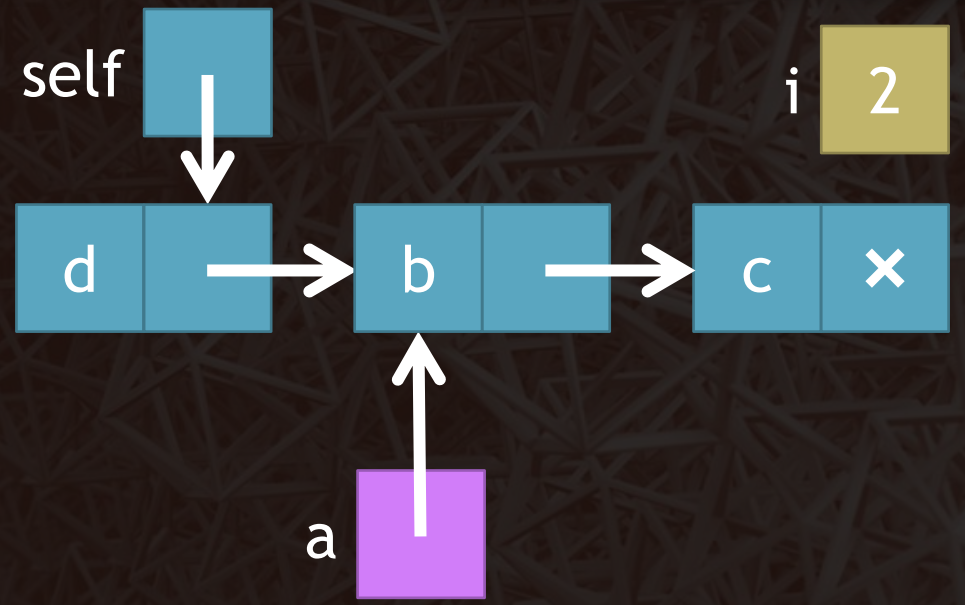
The singlylinkedlist.py file - len

```
def __len__(self):
        a = self
        i = 0
        while a is not None:
            i += 1
            a = a.next
        return i
```



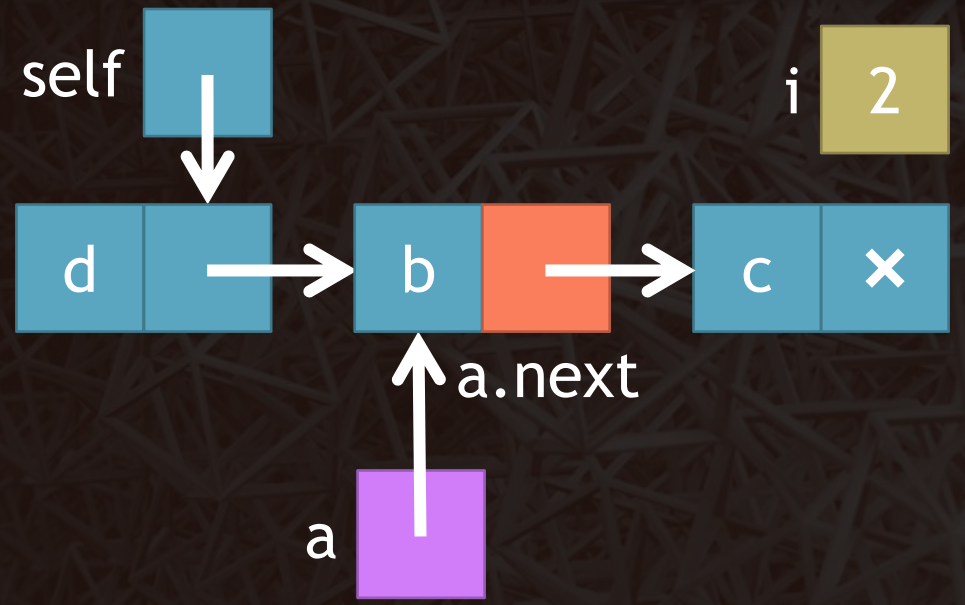
The singlylinkedlist.py file - len

```
def __len__(self):
        a = self
        i = 0
        while a is not None:
            i += 1
            a = a.next
        return i
```



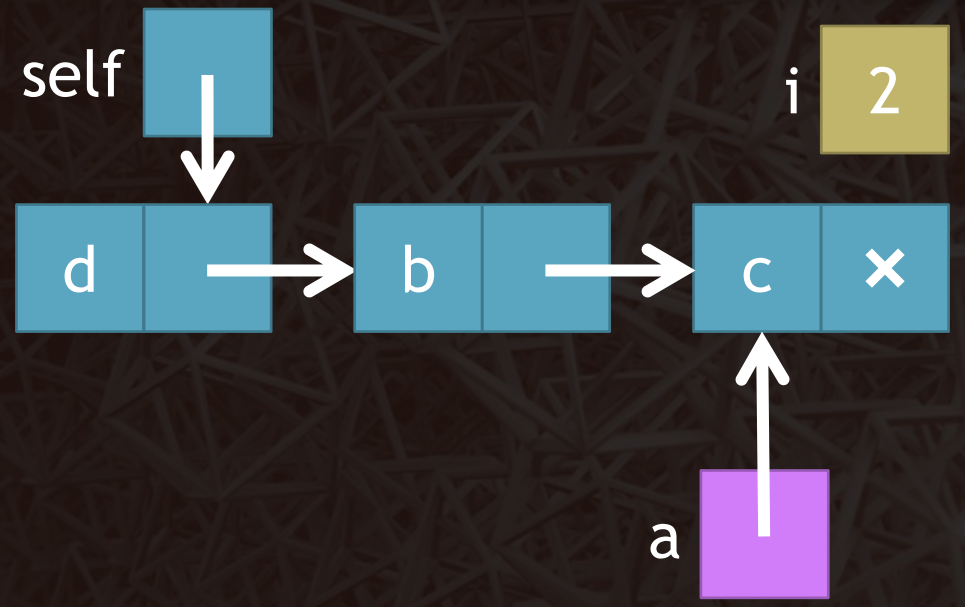
The singlylinkedlist.py file - len

```
def __len__(self):
        a = self
        i = 0
        while a is not None:
            i += 1
            a = a.next
        return i
```



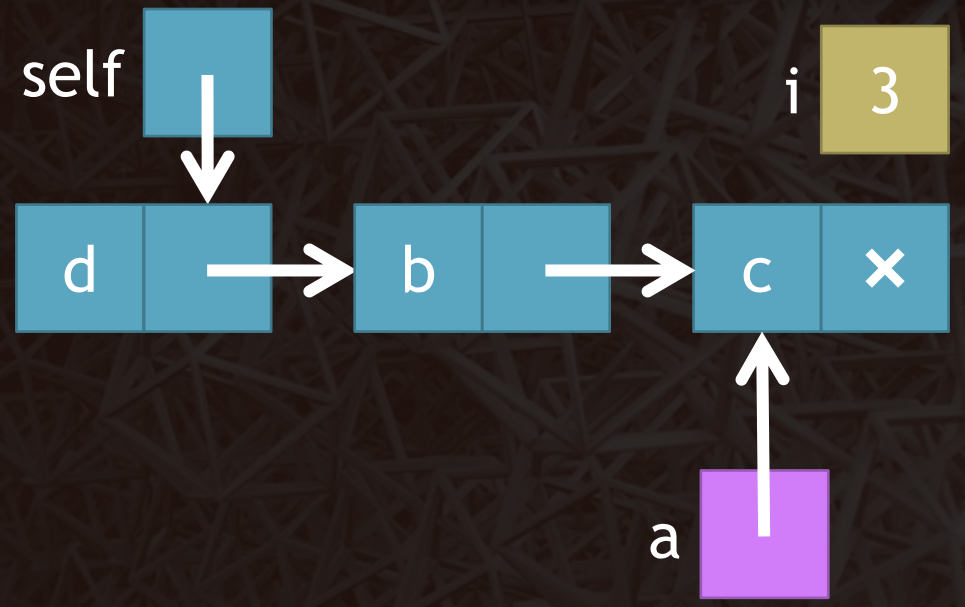
The singlylinkedlist.py file - len

```
def __len__(self):
        a = self
        i = 0
        while a is not None:
            i += 1
            a = a.next
        return i
```



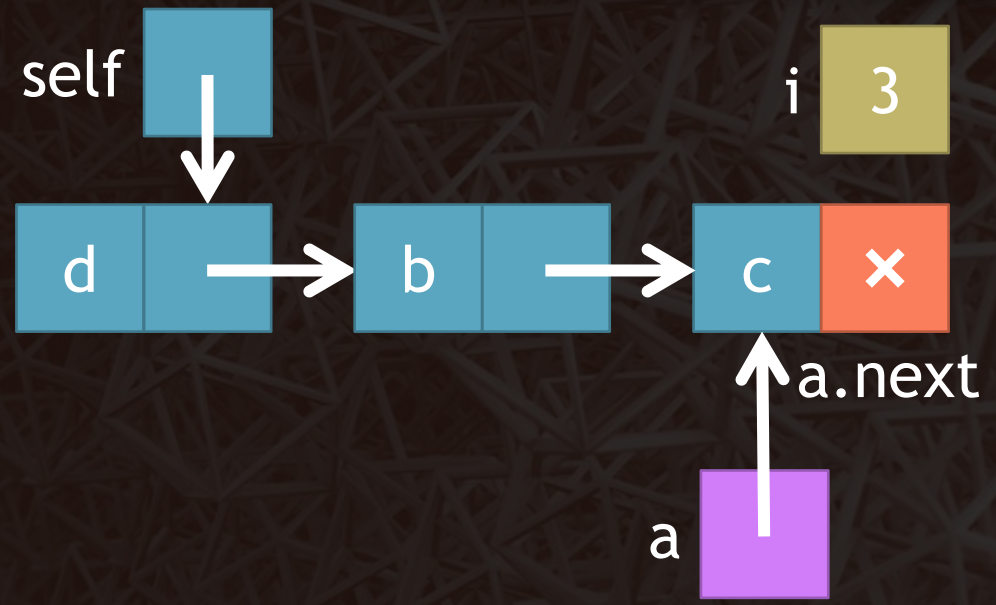
The singlylinkedlist.py file - len

```
def __len__(self):
        a = self
        i = 0
        while a is not None:
            i += 1
            a = a.next
        return i
```



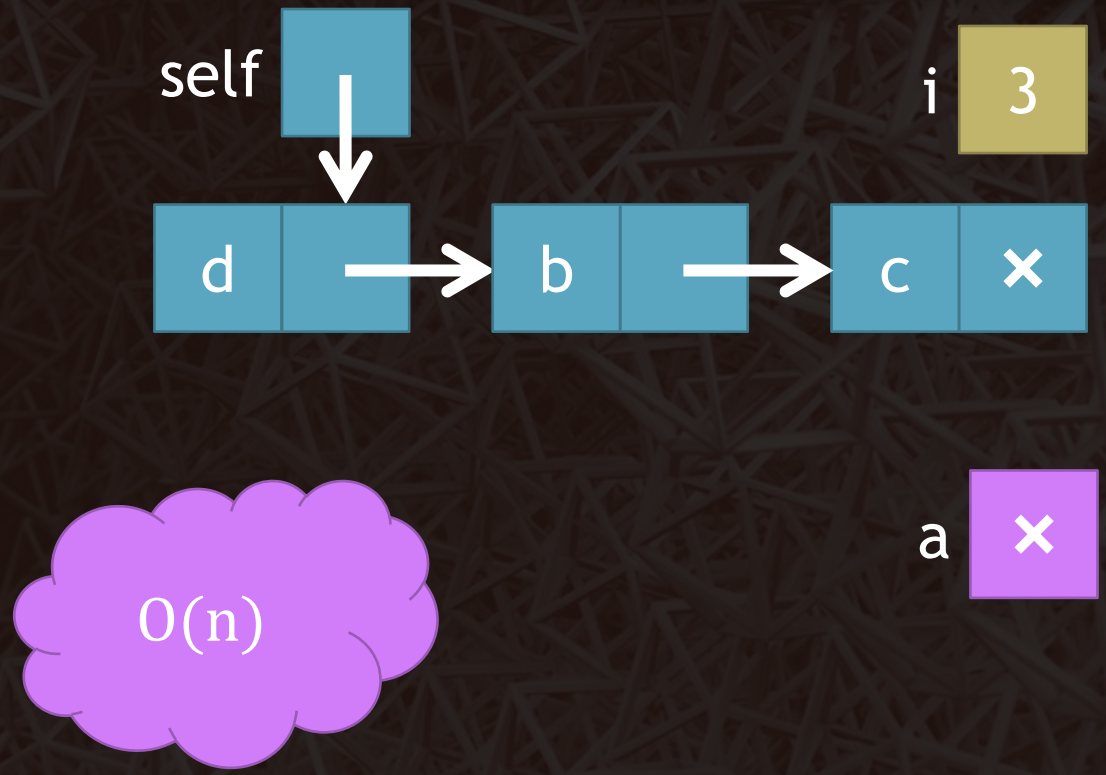
The singlylinkedlist.py file - len

```
def __len__(self):
        a = self
        i = 0
        while a is not None:
            i += 1
            a = a.next
        return i
```



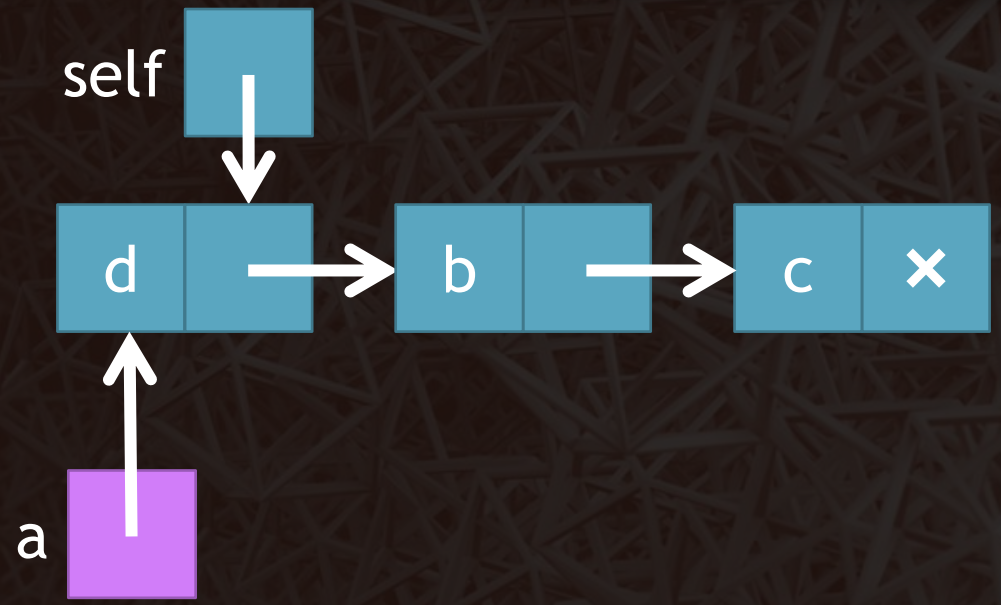
The singlylinkedlist.py file - len

```
def __len__(self):
        a = self
        i = 0
        while a is not None:
            i += 1
            a = a.next
        return i
```



The singlylinkedlist.py file - traverse

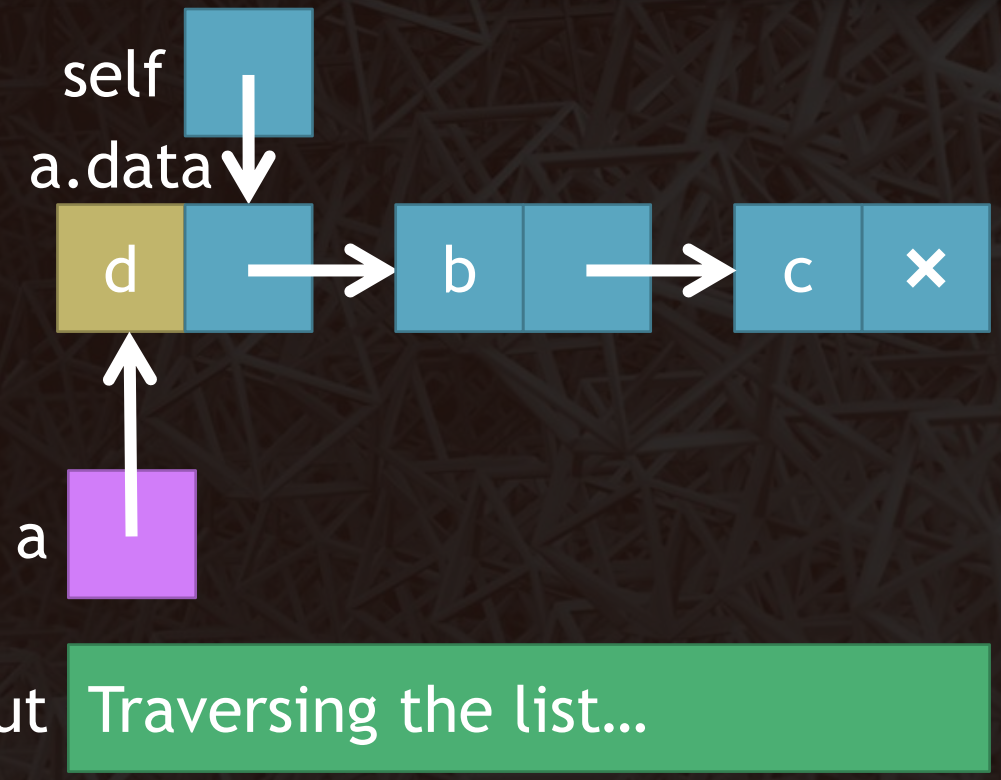
```
def traverse(self):
        a=self
        print("\nTraversing the list...")
        while a is not None:
            print(a.data,end=" ")
            a=a.next
```



output Traversing the list...

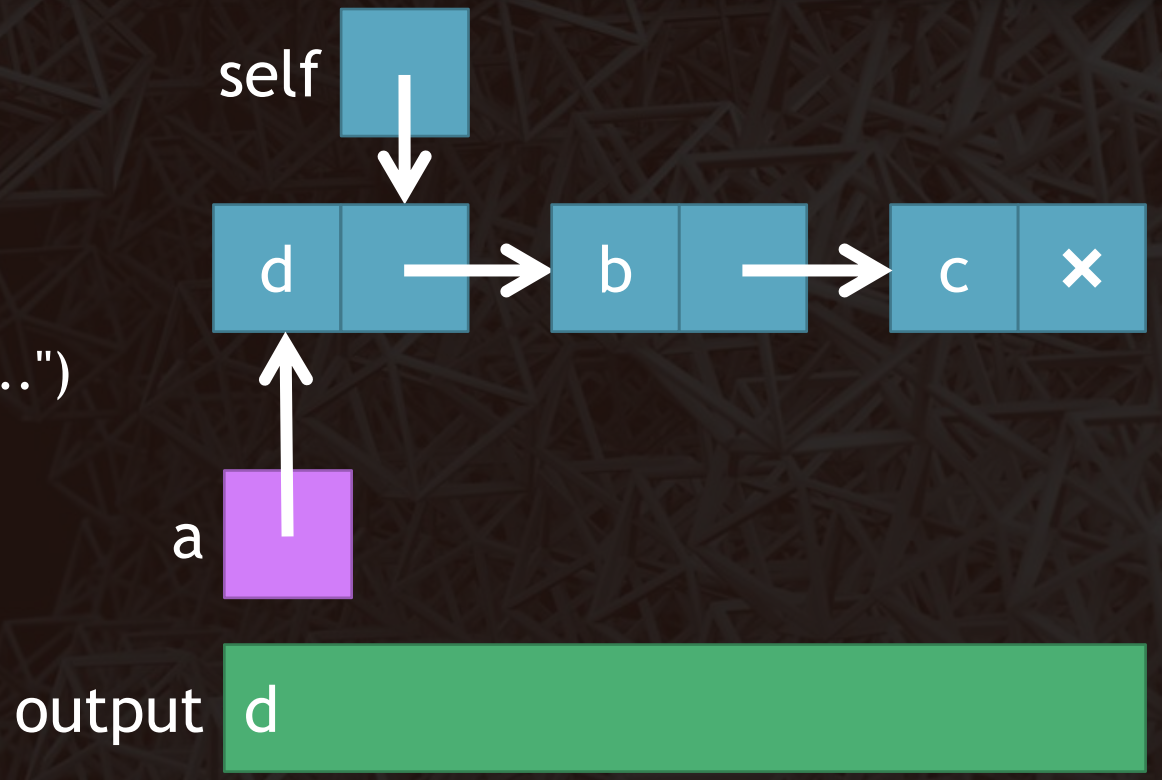
The singlylinkedlist.py file - traverse

```
def traverse(self):
        a=self
        print("\nTraversing the list...")
        while a is not None:
            print(a.data,end=" ")
            a=a.next
```



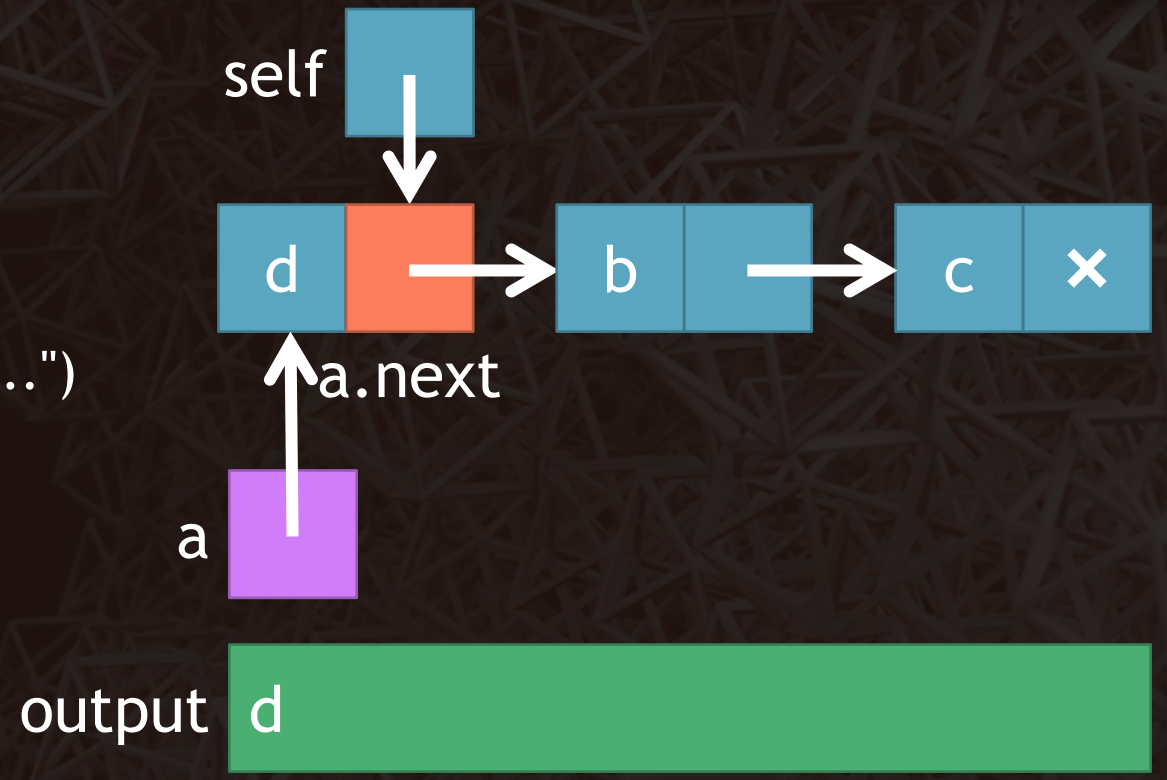
The singlylinkedlist.py file - traverse

```
def traverse(self):
        a=self
        print("\nTraversing the list...")
        while a is not None:
            print(a.data,end=" ")
            a=a.next
```



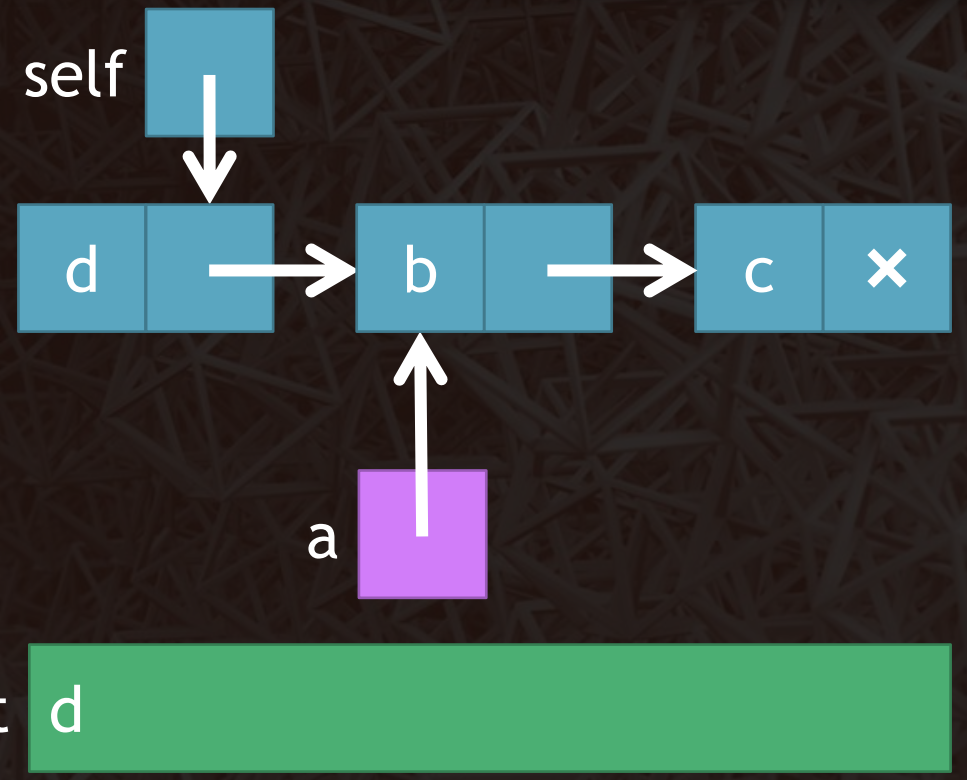
The singlylinkedlist.py file - traverse

```
def traverse(self):
        a=self
        print("\nTraversing the list...")
        while a is not None:
            print(a.data,end=" ")
            a=a.next
```



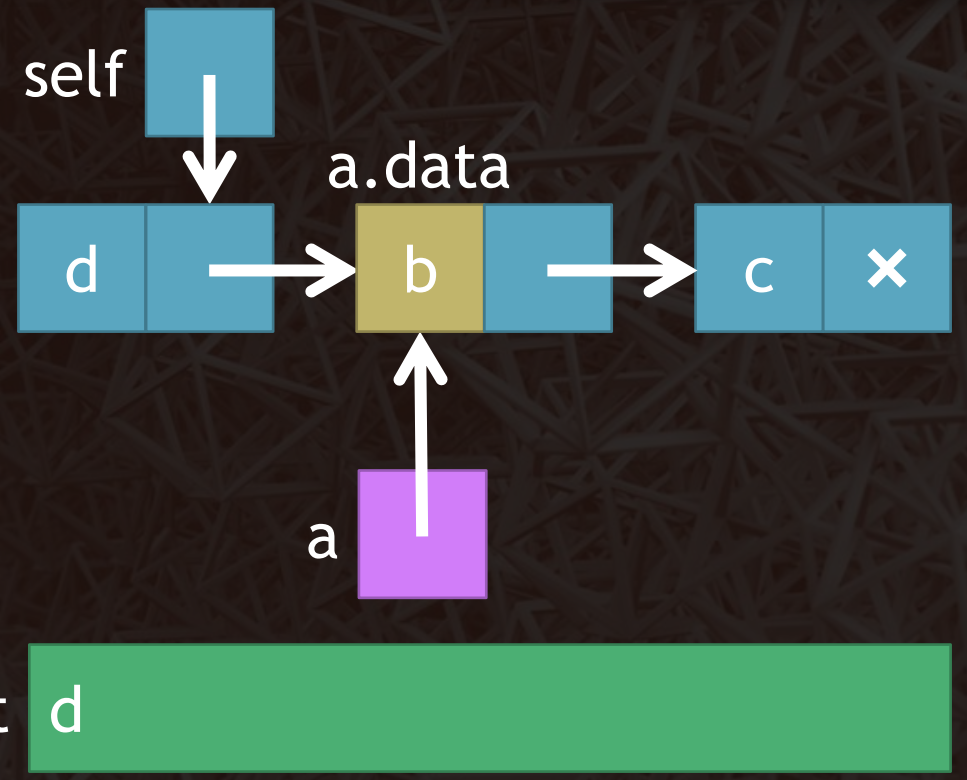
The singlylinkedlist.py file - traverse

```
def traverse(self):
        a=self
        print("\nTraversing the list...")
        while a is not None:
            print(a.data,end=" ")
            a=a.next
```



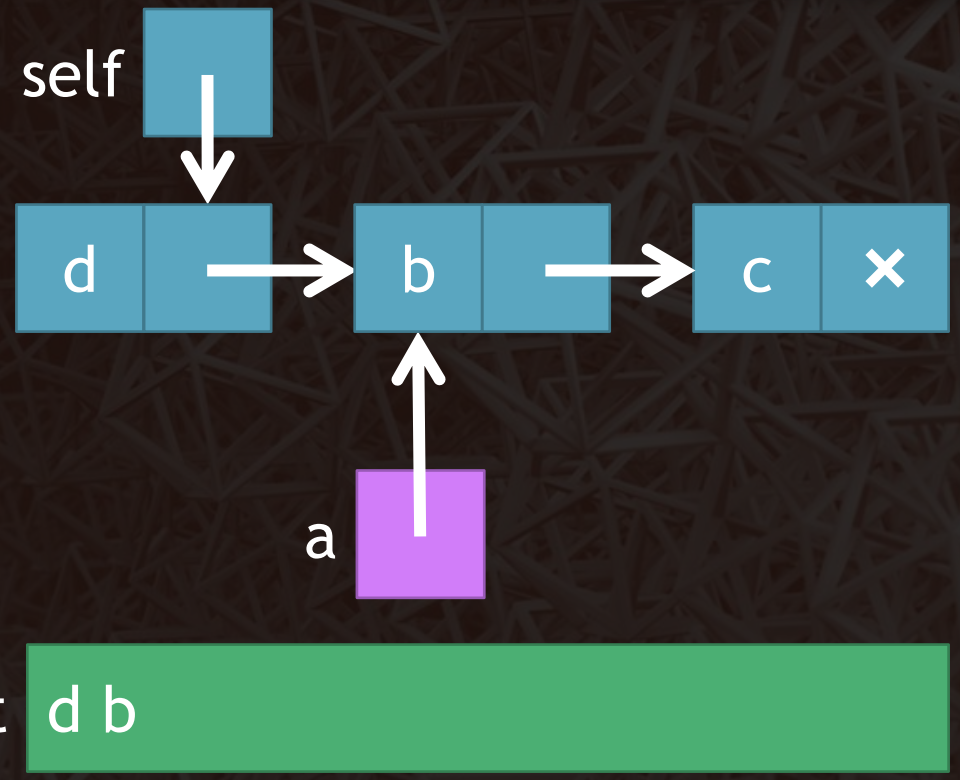
The singlylinkedlist.py file - traverse

```
def traverse(self):
        a=self
        print(“\nTraversing the list...")
        while a is not None:
            print(a.data,end=" ")
            a=a.next
```



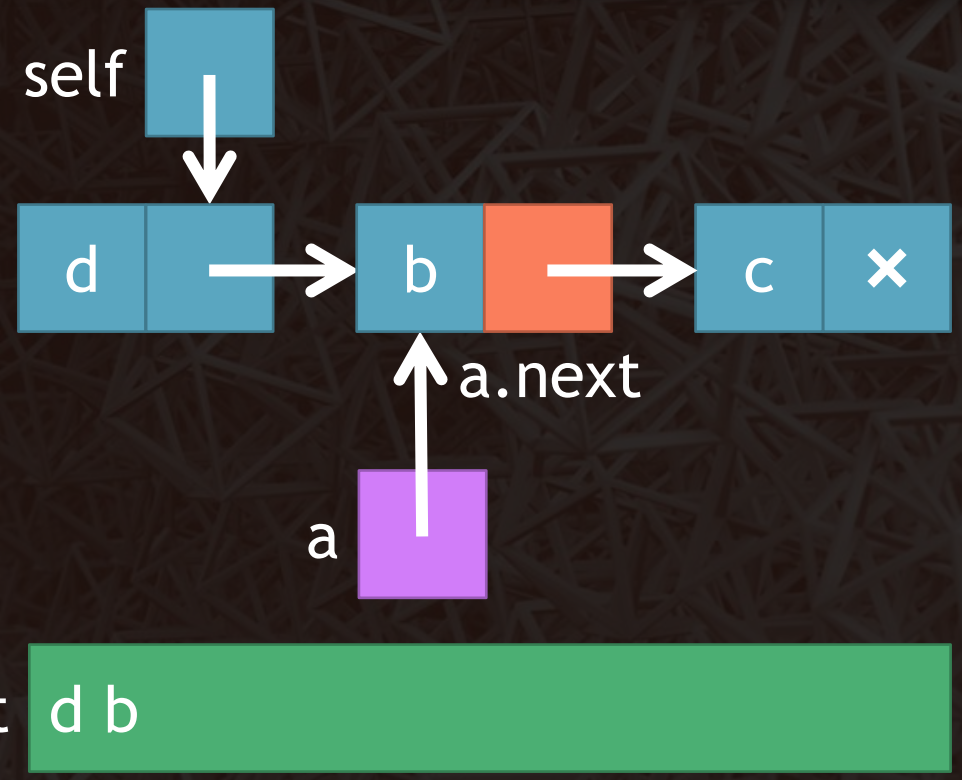
The singlylinkedlist.py file - traverse

```
def traverse(self):
        a=self
        print(“\nTraversing the list...")
        while a is not None:
            print(a.data,end=" ")
            a=a.next
```



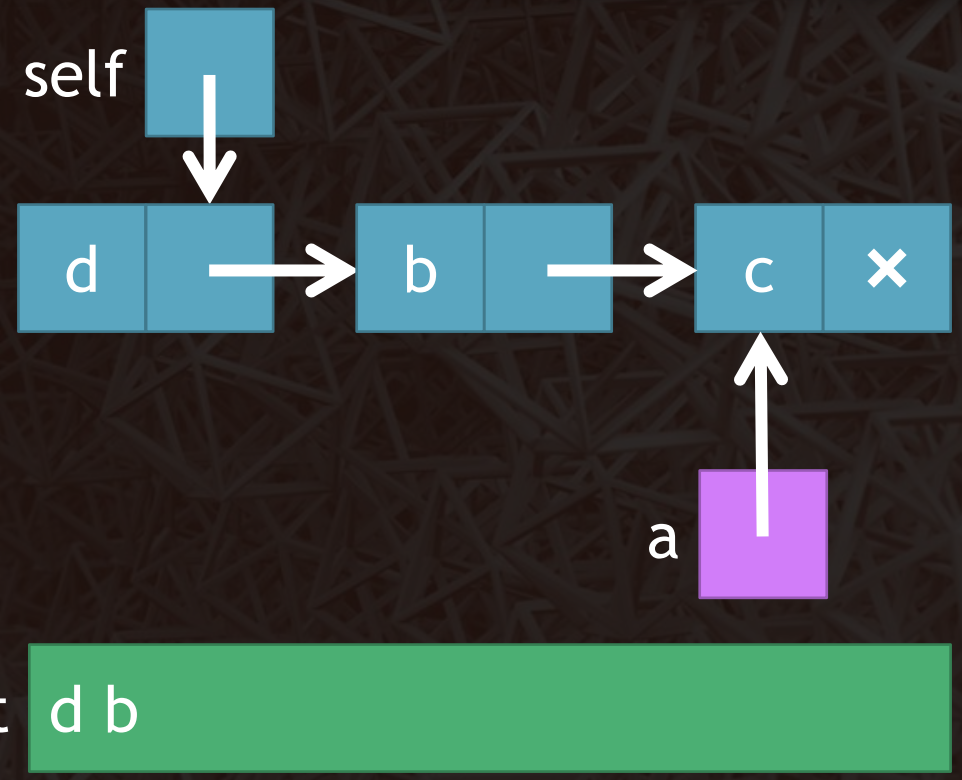
The singlylinkedlist.py file - traverse

```
def traverse(self):
        a=self
        print("\nTraversing the list...")
        while a is not None:
            print(a.data,end=" ")
            a=a.next
```



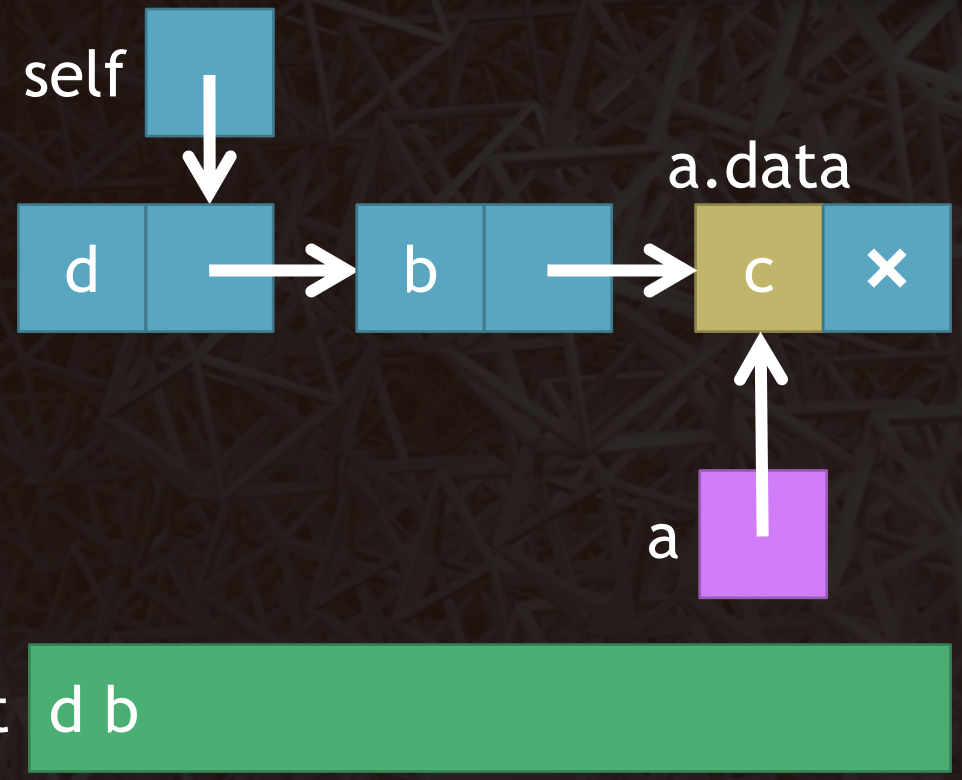
The singlylinkedlist.py file - traverse

```
def traverse(self):
        a=self
        print("\nTraversing the list...")
        while a is not None:
            print(a.data,end=" ")
            a=a.next
```



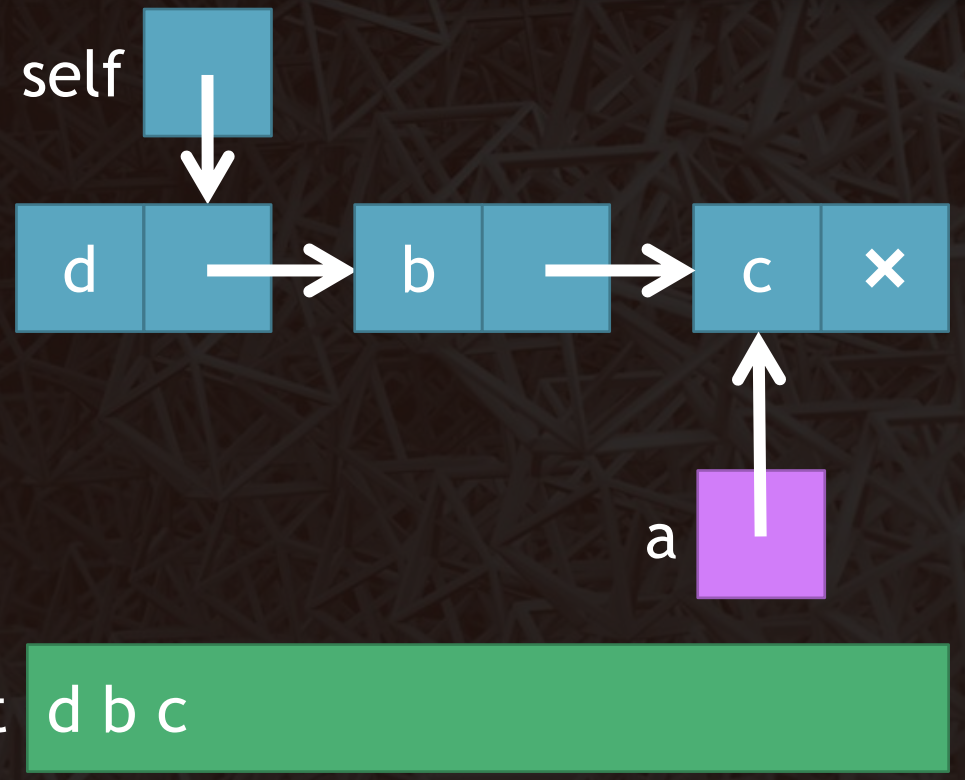
The singlylinkedlist.py file - traverse

```
def traverse(self):
        a=self
        print("\nTraversing the list...")
        while a is not None:
            print(a.data,end=" ")
            a=a.next
```



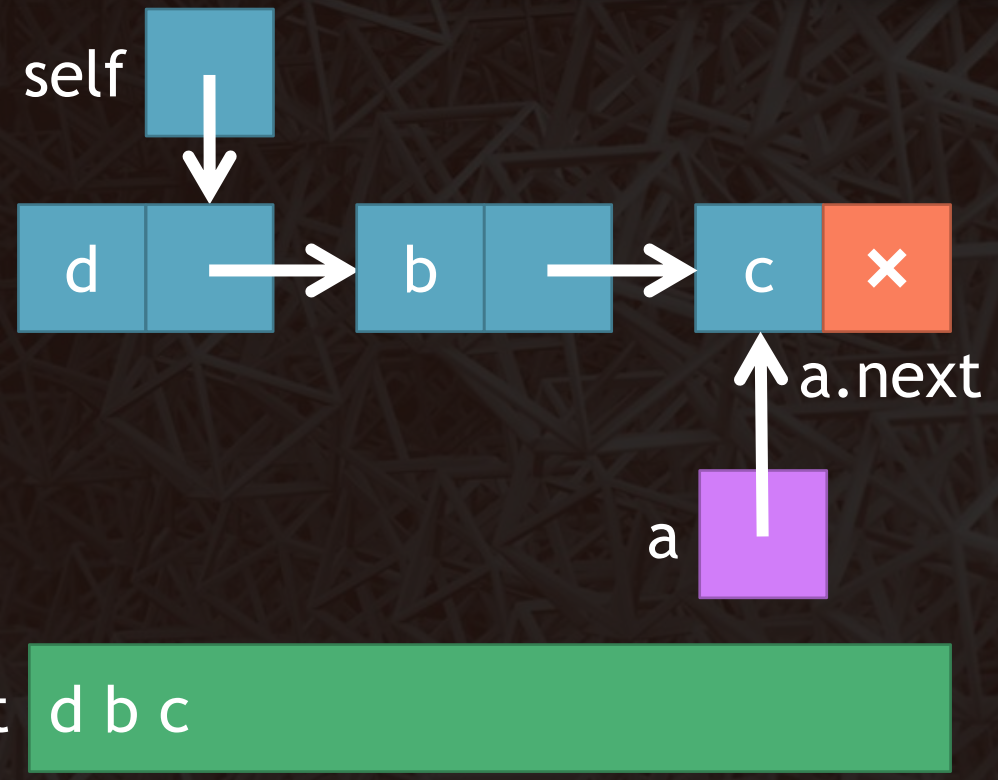
The singlylinkedlist.py file - traverse

```
def traverse(self):
        a=self
        print("\nTraversing the list...")
        while a is not None:
            print(a.data,end=" ")
            a=a.next
```



The singlylinkedlist.py file - traverse

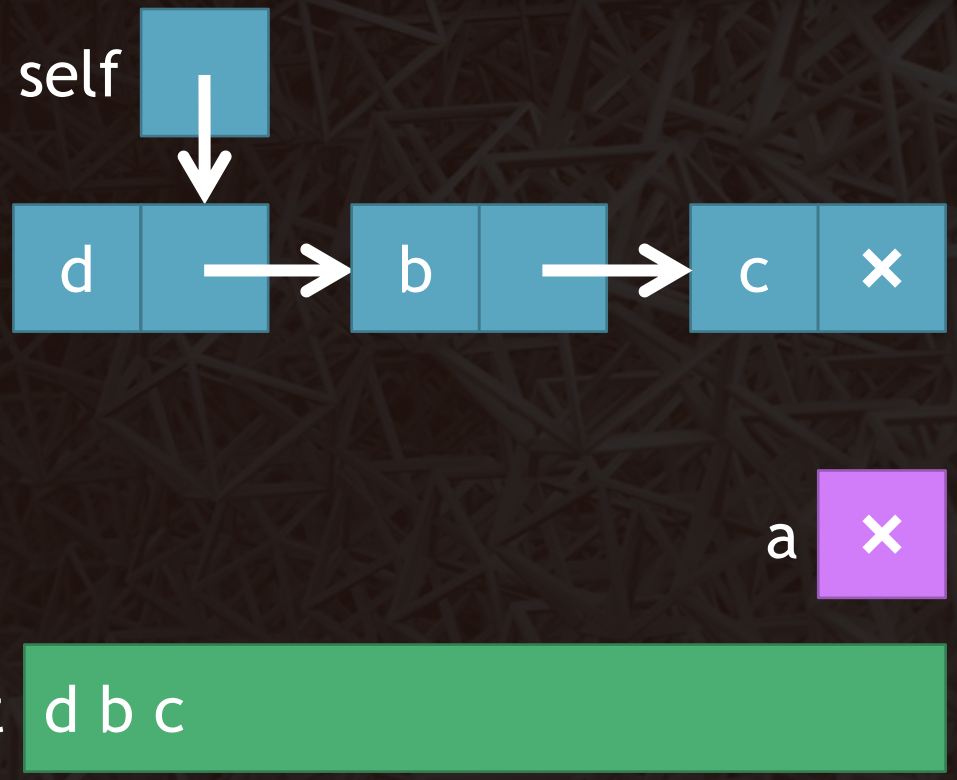
```
def traverse(self):
        a=self
        print("\nTraversing the list...")
        while a is not None:
            print(a.data,end=" ")
            a=a.next
```



The singlylinkedlist.py file - traverse

O(n)

```
def traverse(self):
        a=self
        print("\nTraversing the list...")
        while a is not None:
            print(a.data,end=" ")
            a=a.next
```

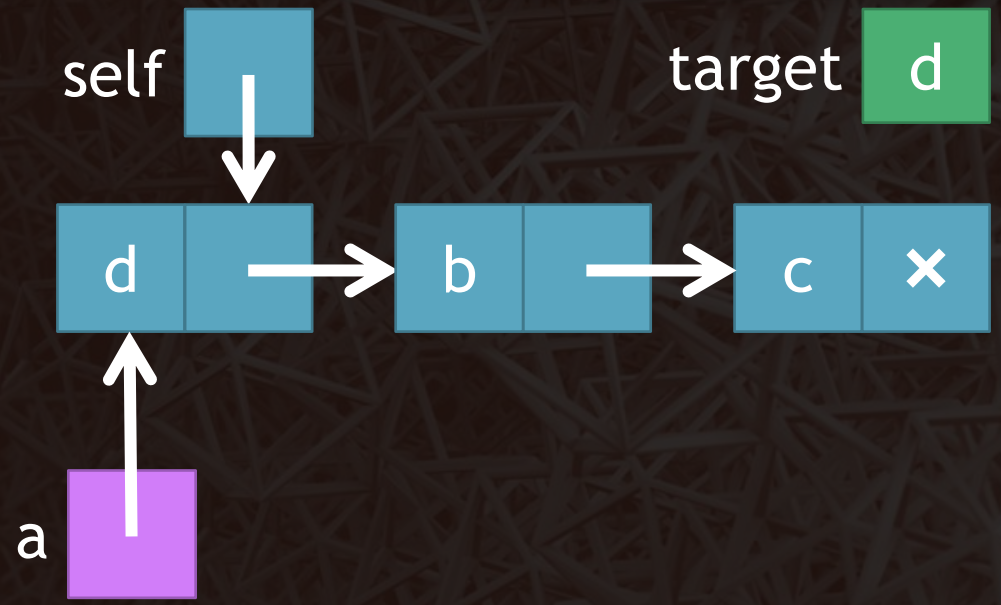


Searching in a singly linked list

- We implement a function to search for a data item in the linked list as a class function of the class ListNode.
- This function returns a Python list of three elements.
- The first member of the returned Python list is True if the target of the search is present in the linked list, and False otherwise.
- The second member of the returned Python list is a pointer to the node just before the node containing the target of the search.
- The third member of the returned Python list is a pointer to the node containing the target of the search.
- If the target is found in the first node of the linked list, the second member of the returned Python list is None.
- We return the pointer to the node before the node containing the target, to help a user who is looking for the target to finally delete it.

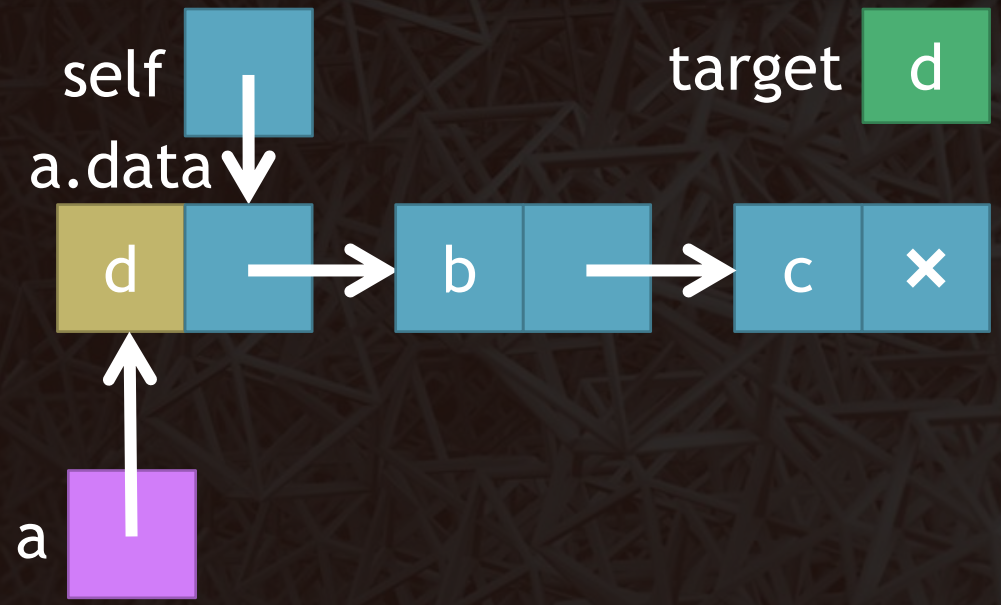
The singlylinkedlist.py file - search

```
def search(self,target):
    a=self
    if a.data==target:
        return [True, None, a]
    b=a.next
    while b is not None and \
                b.data!=target:
        a=a.next
        b=b.next
    return [b is not None, a, b]
```



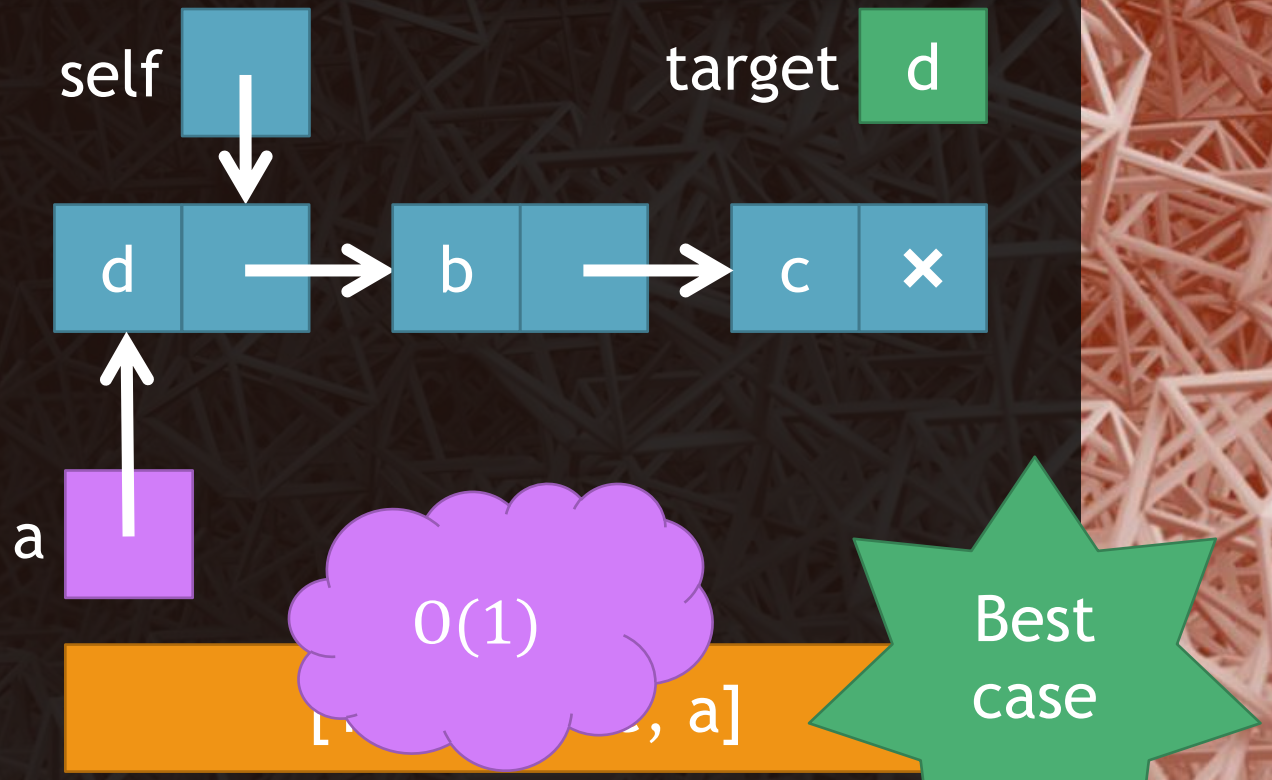
The singlylinkedlist.py file - search

```
def search(self,target):
    a=self
    if a.data==target:
        return [True, None, a]
    b=a.next
    while b is not None and \
                b.data!=target:
        a=a.next
        b=b.next
    return [b is not None, a, b]
```



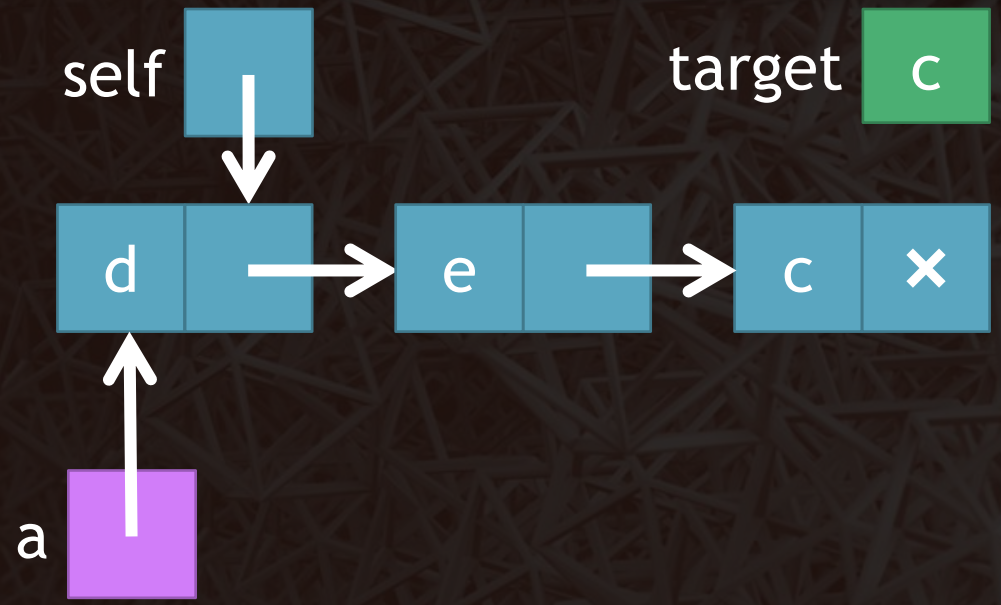
The singlylinkedlist.py file - search

```
def search(self,target):
    a=self
    if a.data==target:
        return [True, None, a]
    b=a.next
    while b is not None and \
                b.data!=target:
        a=a.next
        b=b.next
    return [b is not None, a, b]
```



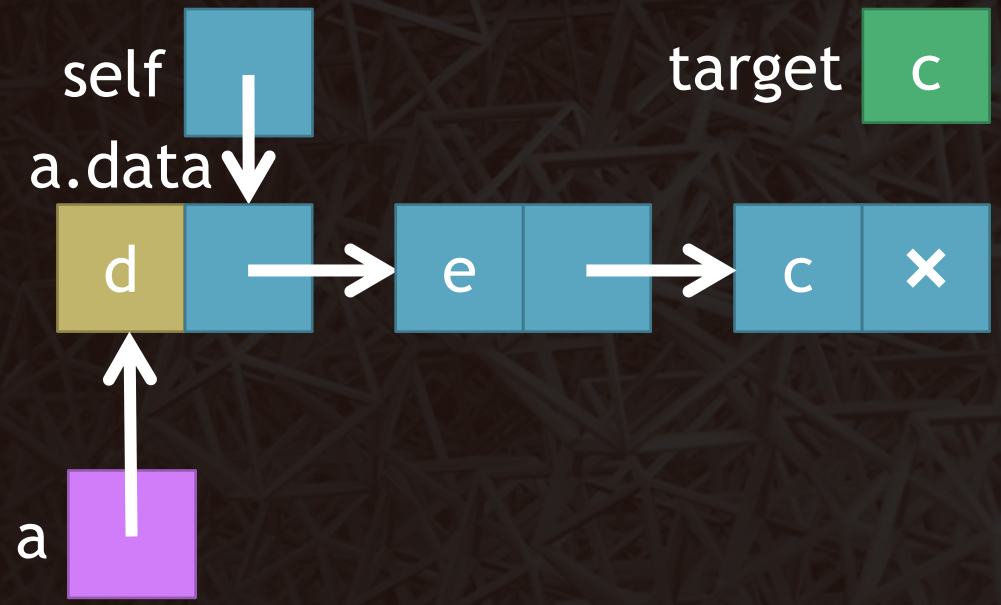
The singlylinkedlist.py file - search

```
def search(self,target):
    a=self
    if a.data==target:
        return [True, None, a]
    b=a.next
    while b is not None and \
                b.data!=target:
        a=a.next
        b=b.next
    return [b is not None, a, b]
```



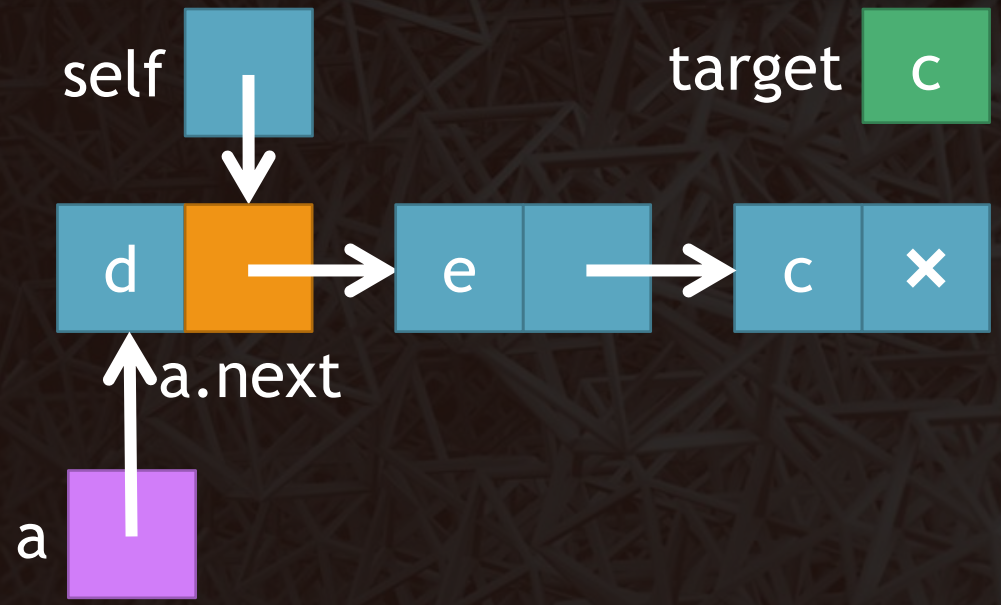
The singlylinkedlist.py file - search

```
def search(self,target):
    a=self
    if a.data==target:
        return [True, None, a]
    b=a.next
    while b is not None and \
                b.data!=target:
        a=a.next
        b=b.next
    return [b is not None, a, b]
```



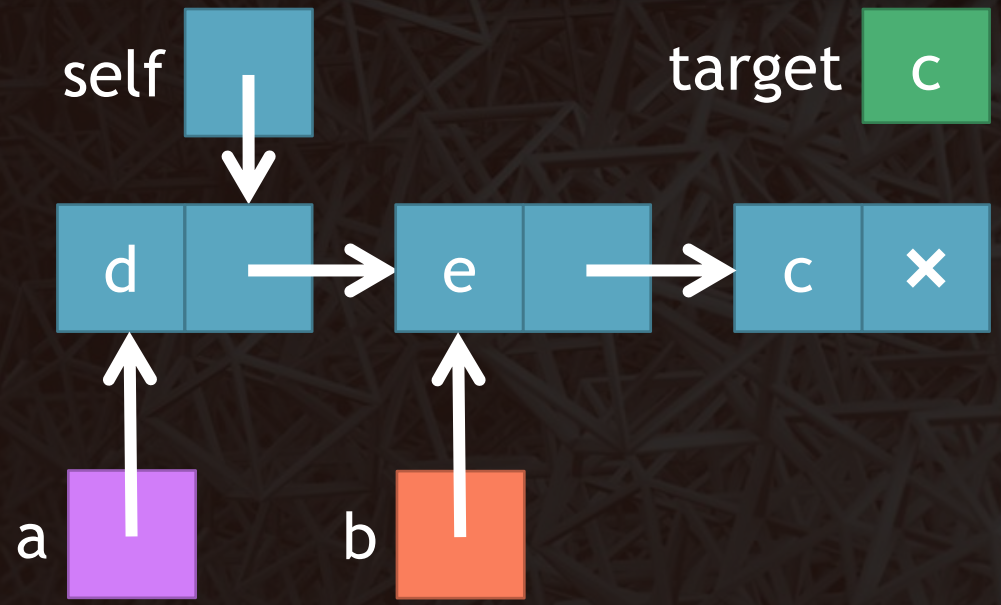
The singlylinkedlist.py file - search

```
def search(self,target):
    a=self
    if a.data==target:
        return [True, None, a]
    b=a.next
    while b is not None and \
                b.data!=target:
        a=a.next
        b=b.next
    return [b is not None, a, b]
```



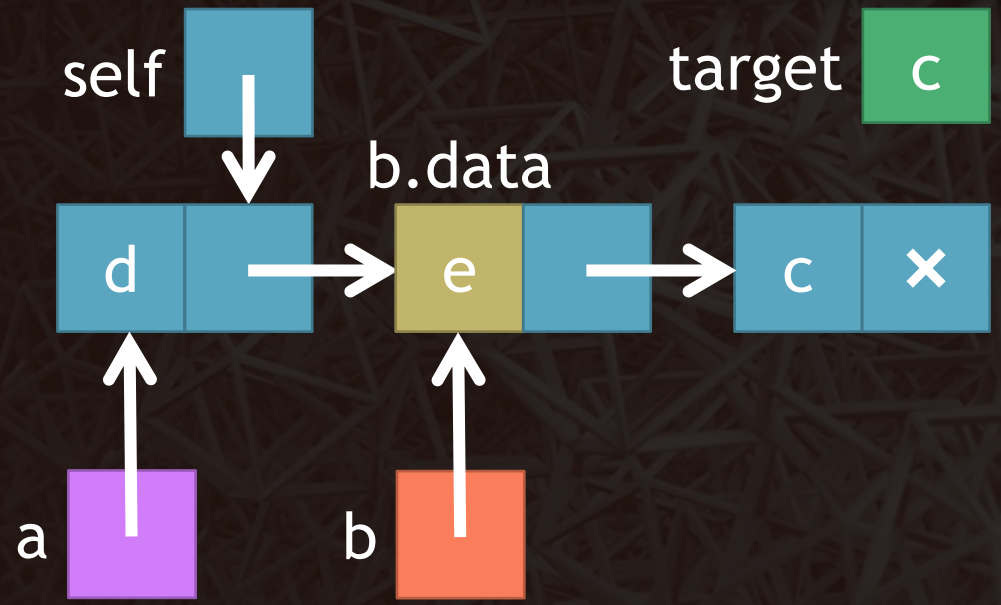
The singlylinkedlist.py file - search

```
def search(self,target):
    a=self
    if a.data==target:
        return [True, None, a]
    b=a.next
    while b is not None and \
                b.data!=target:
        a=a.next
        b=b.next
    return [b is not None, a, b]
```



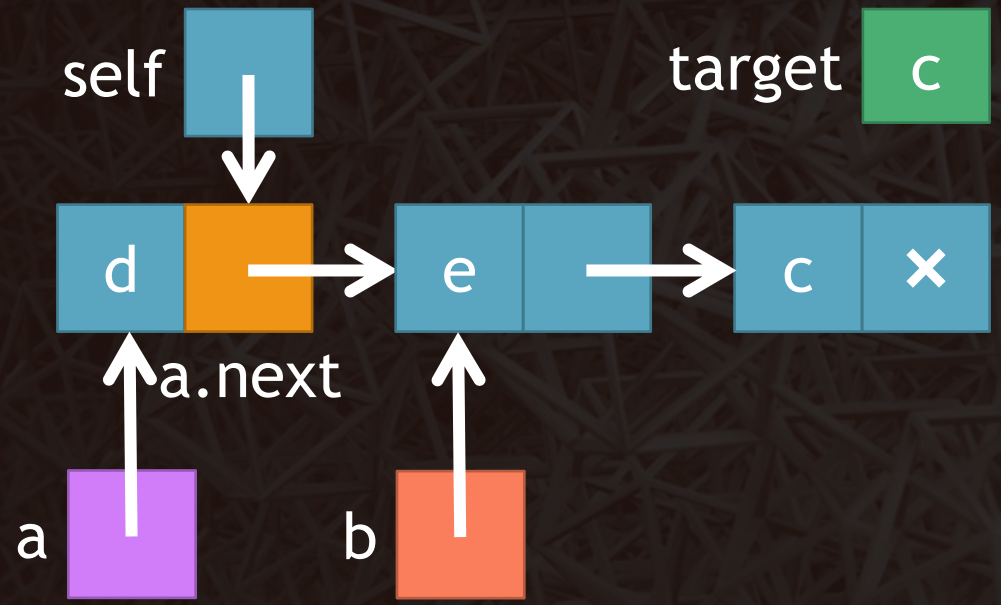
The singlylinkedlist.py file - search

```
def search(self,target):
    a=self
    if a.data==target:
        return [True, None, a]
    b=a.next
    while b is not None and \
                b.data!=target:
        a=a.next
        b=b.next
    return [b is not None, a, b]
```



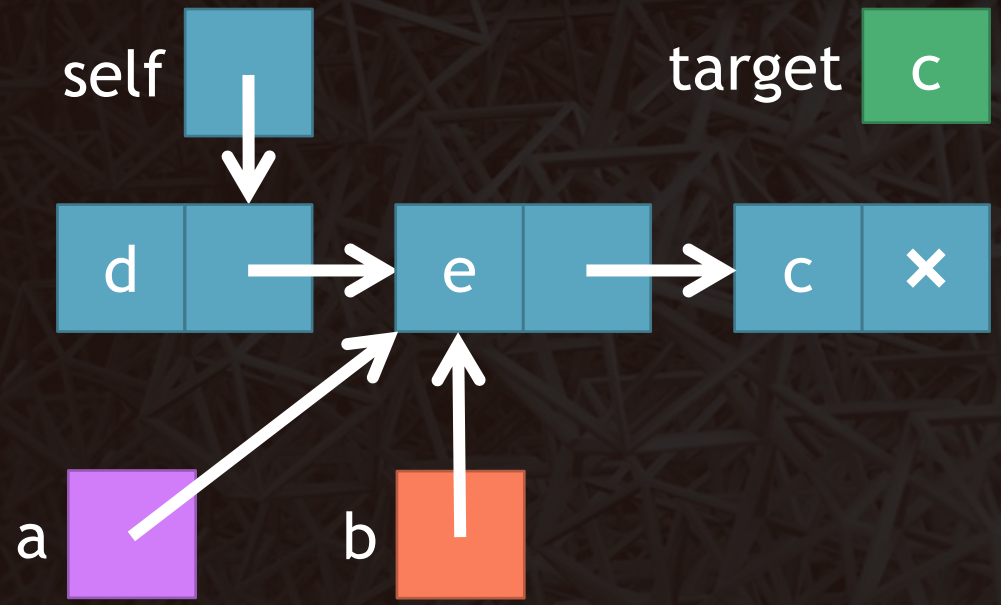
The singlylinkedlist.py file - search

```
def search(self,target):
    a=self
    if a.data==target:
        return [True, None, a]
    b=a.next
    while b is not None and \
                b.data!=target:
        a=a.next
        b=b.next
    return [b is not None, a, b]
```



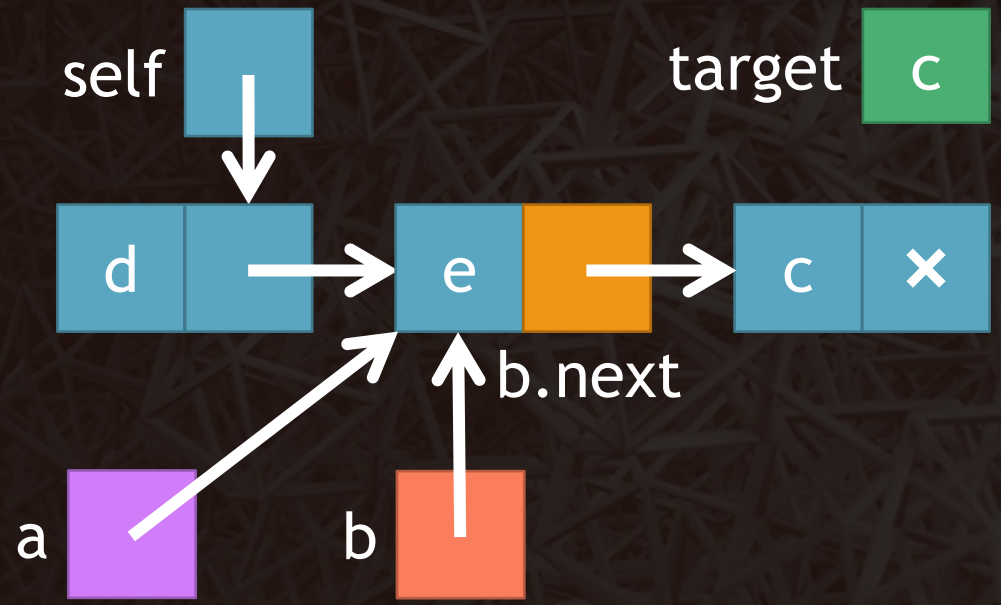
The singlylinkedlist.py file - search

```
def search(self,target):
    a=self
    if a.data==target:
        return [True, None, a]
    b=a.next
    while b is not None and \
                b.data!=target:
        a=a.next
        b=b.next
    return [b is not None, a, b]
```



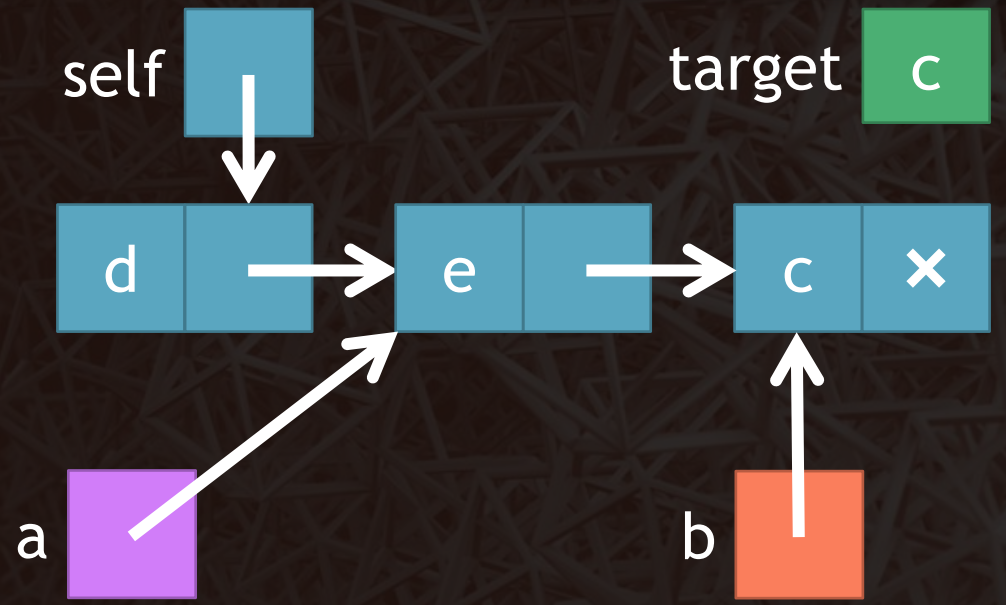
The singlylinkedlist.py file - search

```
def search(self,target):
    a=self
    if a.data==target:
        return [True, None, a]
    b=a.next
    while b is not None and \
                b.data!=target:
        a=a.next
        b=b.next
    return [b is not None, a, b]
```



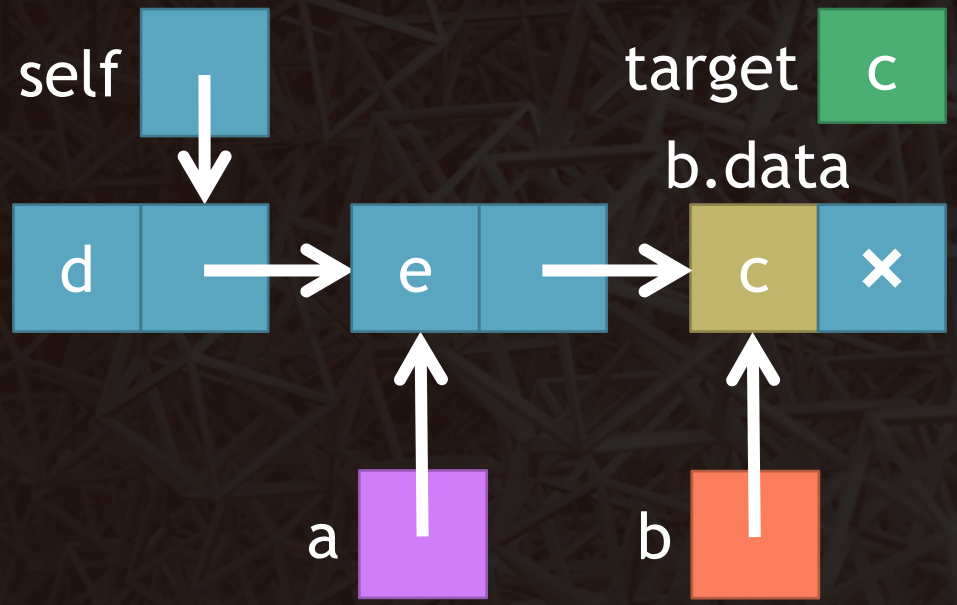
The singlylinkedlist.py file - search

```
def search(self,target):
    a=self
    if a.data==target:
        return [True, None, a]
    b=a.next
    while b is not None and \
                b.data!=target:
        a=a.next
        b=b.next
    return [b is not None, a, b]
```



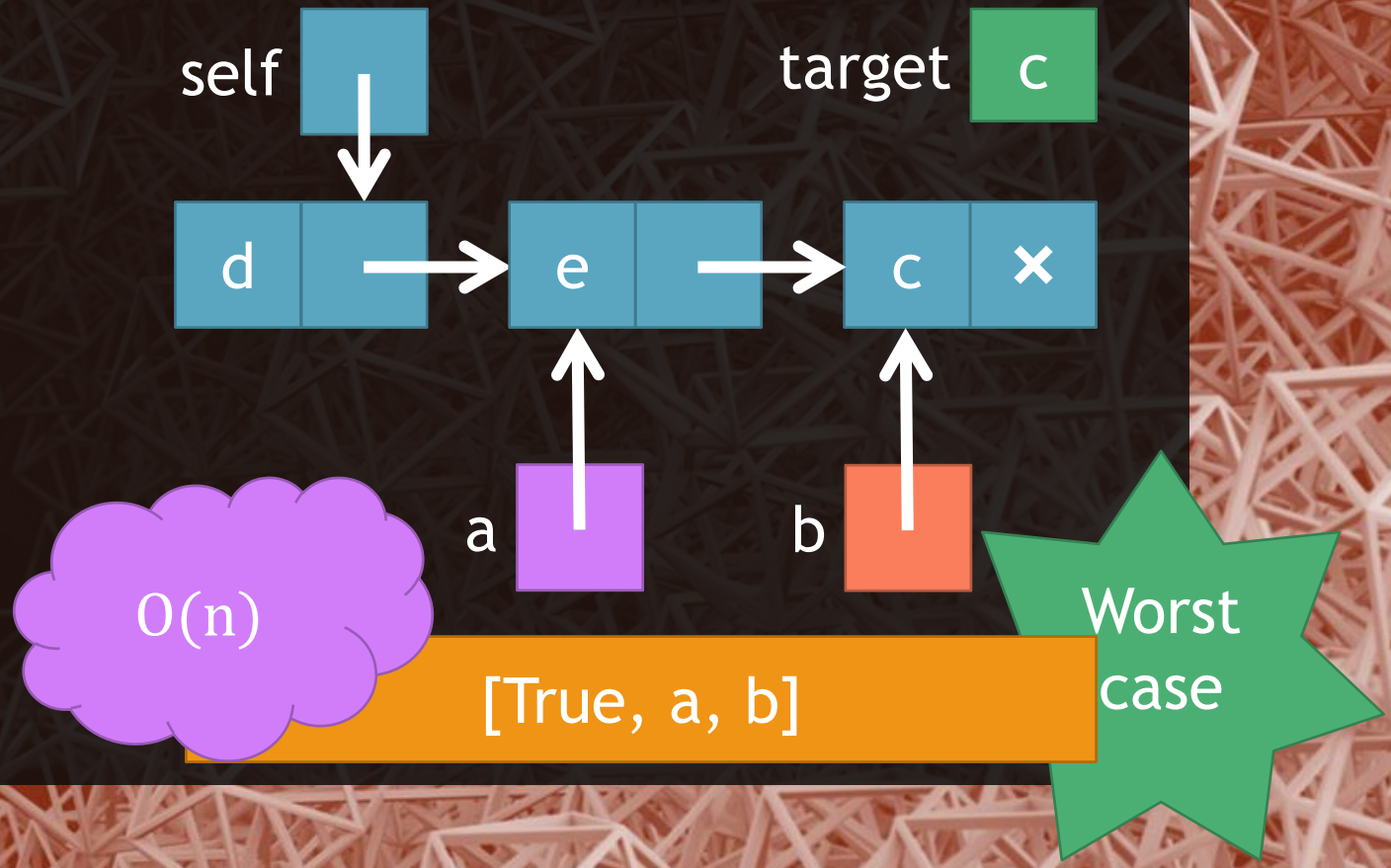
The singlylinkedlist.py file - search

```
def search(self,target):
    a=self
    if a.data==target:
        return [True, None, a]
    b=a.next
    while b is not None and \
                b.data!=target:
        a=a.next
        b=b.next
    return [b is not None, a, b]
```



The singlylinkedlist.py file - search

```
def search(self,target):
    a=self
    if a.data==target:
        return [True, None, a]
    b=a.next
    while b is not None and \
                b.data!=target:
        a=a.next
        b=b.next
    return [b is not None, a, b]
```



Homework

A singly linked list contains the following elements in sequence: 1,5,7,3,8. Run the search function for target = 6. Draw figures for all steps. Can this be considered the function's best or worst case?

So what did we learn today?

We were introduced to singly linked lists.

We implemented a class to create and manage a singly linked list.

We saw pictorial depictions of code fragments.

We found time complexities of all codes.

Things to do

Read the book!

Note your questions and put them up in the relevant online session.

Email suggestions on content or quality of this lecture at uroojain@neduet.edu.pk