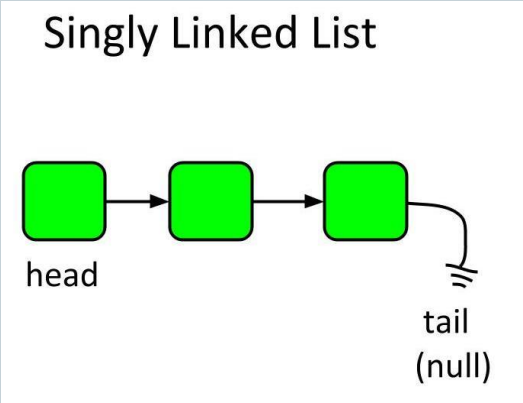
And Even More C++

Outline

- Coming Up:
  - C++ Classes
  - Special Members
  - Friendship

- But first…
  - A review of linked lists



Sequential vs. Linked

Memory address	Value
C0	"The"
C1	"cat"
C2	"sat"
C3	-
C4	-
C5	-
C6	-
C7	-
C8	-
C9	-

Python list

Memory address	Value
C0	"cat"
C1	C8
C2	-
C3	-
C4	"The"
C5	C0
C6	-
C7	-
C8	"sat"
C9	null

linked list

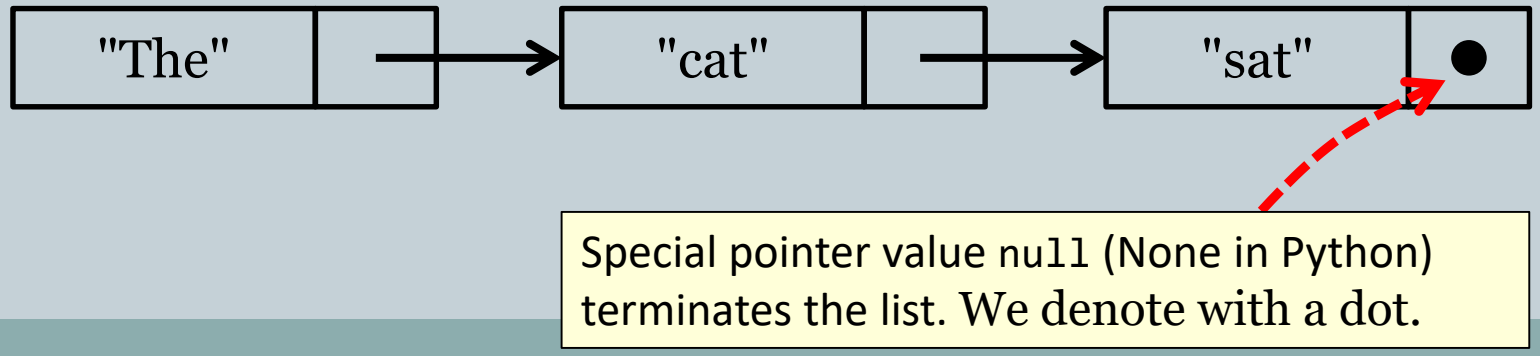
Linked List

- Linked list
  - Simplest linked data structure
  - Node is a recursive data structure
  - Each node contains:
    - An item (some data)
    - A pointer to next node in the list

```
class Node:

def __init__(self, s):
    self.item = s
    self.next = None
```

```
struct node
{
    …
    // data items
    …
    node * next;
};
```

Three Node objects hooked together to form a linked list



Special pointer value null (None in Python) terminates the list. We denote with a dot.

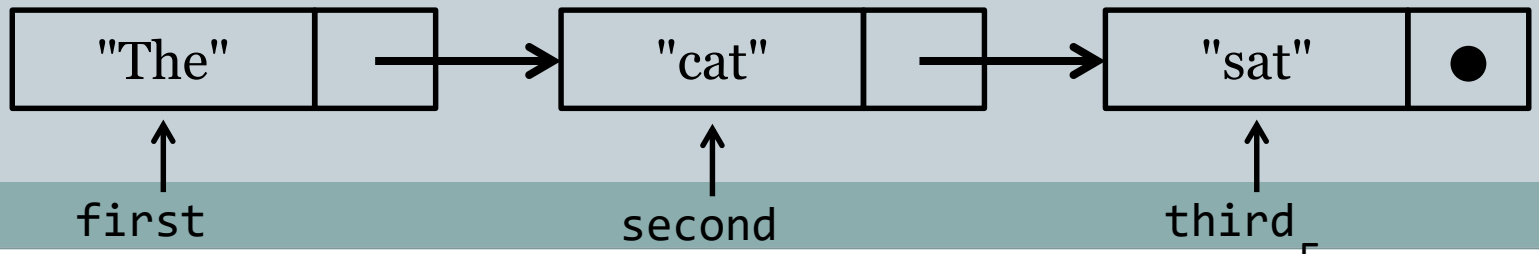
Building a linked list

```
first  = Node()
first.item  = "The"

second = Node()
se
th
th
fi
se
```

```
node * first  = new node;
first->item  = "The";

node * second = new node;
second->item = "cat";

node * third  = new node;
third->item  = "sat";

first->next  = second;
second->next = third;
```

second →
first →
third →

Memory address	Value
C0	"cat"
C1	C8
C2	-
C3	-
C4	"The"
C5	C0
C6	-
C7	-
C8	"sat"
C9	null



Some Demo Code

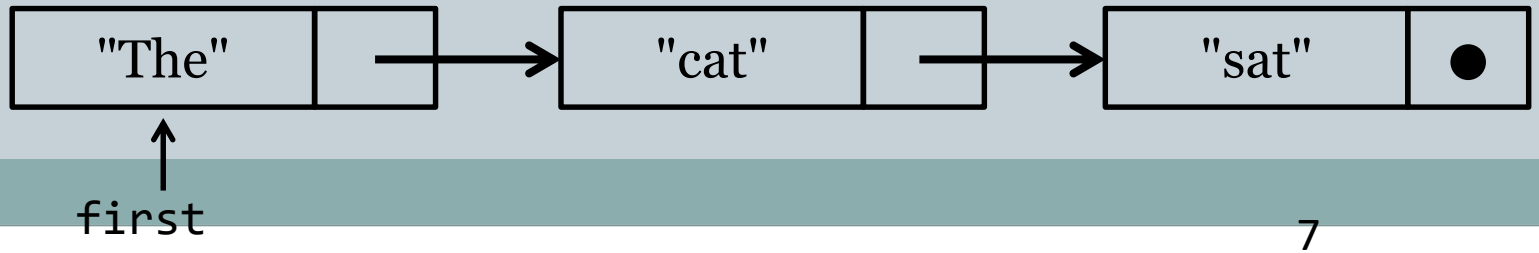
```
#include <stdlib.h>
#include <iostream>
#include <string>
using namespace std;

struct node
{
    string item;
    node * next;
};

int main()
{
    node * first  = new node;
    first->item  = "The";

    node * second = new node;
    second->item = "cat";

    node * third  = new node;
    third->item  = "sat";

    first->next  = second;
    second->next = third;

    cout << first->item << endl;
    cout << second->item << endl;
    cout << third->item << endl;

    return 0;
}
```

Traversing a List

- Iterate over all elements in a linked list
  - Assume list is null terminated
  - Assume `first` instance variable points to start of list
  - Print all the strings in the list

```
current = first
while current != None:
    print(current.item)
    current = current.next
```

```
node * current = first;
  while (current != NULL)
  {
          cout << current->item << endl;
          current = current->next;
  }
```



Playing with a Linked List

- What things might we want to do with a list?
  - Construct a node
  - Add a node to the end
  - Insert a node at a certain position
  - Remove a node from a position
  - Print out the list of nodes

Data Structures

- A data structure is a group of data elements grouped together under one name
  - Not quite the same thing as a data type in Java
- Use struct to define a structure in C++

```
struct type_name {
member_type1 member_name1;
member_type2 member_name2;
member_type3 member_name3;
.
.
} object_names;
```

```
struct product {
  int weight;
  double price;
} ;

product apple;
product banana, melon;
```

```
struct product {
  int weight;
  double price;
} apple, banana, melon;
```

Pointers to Structures

- The arrow operator -> is used to access structures that have member elements

```
struct movies_t {
  string title;
  int year;
};

movies_t amovie;
movies_t * pmovie;
```

```
pmovie = &amovie;
```

`pmovie->title` is equivalent to: `(*pmovie).title`

`*pmovie.title` is equivalent to: `*(pmovie.title)`

Expression	What is evaluated	Equivalent
`a.b`	Member `b` of object `a`	
`a->b`	Member `b` of object pointed to by `a`	`(*a).b`
`*a.b`	Value pointed to by member `b` of object `a`	`*(a.b)`

new and new[]

- new is followed by a data type specifier and if there are multiple elements needed, brackets are used, to specify an array

```
pointer = new type
pointer = new type [number_of_elements]
```

- For example:

```
int * foo;
foo = new int [5];
```

- In this example, a pointer to an integer is created, and then a block of memory is allocated to store 5 of them

Playing with a Linked List

- What things might we want to do with a list?
  - Print out the list of nodes

Playing with a Linked List

- What things might we want to do with a list?
  - Add a node to the end

Playing with a Linked List

- What things might we want to do with a list?
  - Insert a node at a certain position

Playing with a Linked List

- What things might we want to do with a list?
  - Remove a node from a position

delete and delete[]

- C++ does not handle garbage collection for you
  - You need to determine when a particular data item is no longer needed and then remove it
  - Use delete and delete[] to do this

```
delete pointer;
delete[] pointer;
```

  - The “thing” deleted should be either something that was created with new or new[] before, or it should be a null pointer (in which case nothing happens)

Summary

- Coming Up:
  - C++ Classes
  - Special Members
  - Friendship

- But first…
  - A review of linked lists