And Even More C++
Outline

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

- But first...
  - A review of linked lists
### Sequential vs. Linked

#### Python list

<table>
<thead>
<tr>
<th>Memory address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>&quot;The&quot;</td>
</tr>
<tr>
<td>C1</td>
<td>&quot;cat&quot;</td>
</tr>
<tr>
<td>C2</td>
<td>&quot;sat&quot;</td>
</tr>
<tr>
<td>C3</td>
<td>-</td>
</tr>
<tr>
<td>C4</td>
<td>-</td>
</tr>
<tr>
<td>C5</td>
<td>-</td>
</tr>
<tr>
<td>C6</td>
<td>-</td>
</tr>
<tr>
<td>C7</td>
<td>-</td>
</tr>
<tr>
<td>C8</td>
<td>-</td>
</tr>
<tr>
<td>C9</td>
<td>-</td>
</tr>
</tbody>
</table>

#### Linked list

<table>
<thead>
<tr>
<th>Memory address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>&quot;cat&quot;</td>
</tr>
<tr>
<td>C1</td>
<td>C8</td>
</tr>
<tr>
<td>C2</td>
<td>-</td>
</tr>
<tr>
<td>C3</td>
<td>-</td>
</tr>
<tr>
<td>C4</td>
<td>&quot;The&quot;</td>
</tr>
<tr>
<td>C5</td>
<td>Co</td>
</tr>
<tr>
<td>C6</td>
<td>-</td>
</tr>
<tr>
<td>C7</td>
<td>-</td>
</tr>
<tr>
<td>C8</td>
<td>&quot;sat&quot;</td>
</tr>
<tr>
<td>C9</td>
<td>null</td>
</tr>
</tbody>
</table>

The diagram illustrates the memory addresses and values for both Python list and linked list representations of the phrase "The cat sat."
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

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

```c
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

```c
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;
```

<table>
<thead>
<tr>
<th>Memory address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>&quot;cat&quot;</td>
</tr>
<tr>
<td>C1</td>
<td>C8</td>
</tr>
<tr>
<td>C2</td>
<td>-</td>
</tr>
<tr>
<td>C3</td>
<td>-</td>
</tr>
<tr>
<td>C4</td>
<td>&quot;The&quot;</td>
</tr>
<tr>
<td>C5</td>
<td>Co</td>
</tr>
<tr>
<td>C6</td>
<td>-</td>
</tr>
<tr>
<td>C7</td>
<td>-</td>
</tr>
<tr>
<td>C8</td>
<td>&quot;sat&quot;</td>
</tr>
<tr>
<td>C9</td>
<td>null</td>
</tr>
</tbody>
</table>

The linked list structure:

- **first**: "The" \(\rightarrow\) "cat" \(\rightarrow\) "sat" \(\rightarrow\) null
- **second**: "cat"
- **third**: "sat"
```cpp
#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 variable points to start of list
  - Print all the strings in the list

```cpp
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
Summary

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

- But first...
  - A review of linked lists