Lists and Tuples

Methods in Computational Linguistics I
Lecture 4
Last Time

- Text Corpora in NLTK
Today

• Lists and Tuples
• More work with Storing Scripts and Classes
Lists!

- First and most basic “data structure”.
- Describes an ordering of data.

```
  1  2  7  10 12  4  -2  1
  0  1  2  3  4  5  6  7
```
Lists!

- First and most basic “data structure”.
- Describes an ordering of data.

<table>
<thead>
<tr>
<th>a</th>
<th>q</th>
<th>c</th>
<th>f</th>
<th>k</th>
<th>f</th>
<th>g</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>
Lists!

- First and most basic “data structure”.
- Describes an ordering of data.

<table>
<thead>
<tr>
<th>abe</th>
<th>beth</th>
<th>car</th>
<th>45</th>
<th>9.2</th>
<th>bug</th>
<th>i</th>
<th>q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

List demo.
List indexing

- Why is the first element of a list indexed by 0?

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>abe</td>
<td>beth</td>
<td>car</td>
</tr>
</tbody>
</table>

| 100033 | 'xylem' |
| 100034 | 'abe'   |
| 100035 | 'beth'  |
| 100036 | 'car'   |
| 100037 | 0xf5a3  |
| ...    |         |
| 101002 | 100034  |
List indexing

- Why is the first element of a list indexed by 0?

<table>
<thead>
<tr>
<th></th>
<th>beth</th>
<th>car</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

array[0] = array + 0
array[x] = array + x

<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>100033</td>
<td>'xylem'</td>
</tr>
<tr>
<td>100034</td>
<td>'abe'</td>
</tr>
<tr>
<td>100035</td>
<td>'beth'</td>
</tr>
<tr>
<td>100036</td>
<td>'car'</td>
</tr>
<tr>
<td>100037</td>
<td>0xf5a3</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>101002</td>
<td>100034</td>
</tr>
</tbody>
</table>
Functions

- Functions are a way to **reuse** code.
- Make your code easier to read and understand
- Lead to fewer mistakes!
List and Tuple demo

• demos with lists, tuples and loops.
Rule of Thumb

- Programming should almost never require copy-and-paste
- If you find yourself copying code, and making a slight or no modification...
- Use a Function or Variable
Structuring a program

• Write programs to:
  • Determine if a list is sorted or not.
    [is_sorted.py]
  • Does a string start with “who”, “what”, “where”, “when”, “why”, or “how”?
    [starts_with_wh.py]
  • Does a string end with a question mark?
    [ends_with_qmark.py]
Recap

- Lists
  - Iterating over lists
  - for loops
  - while loops
  - range vs xrange
- Comments
- Writing programs.
Parsing

- Generate a Parse Tree from:
  - The surface form (words) of the text
  - Part of Speech Tokens
Parsing Styles
• Parse Trees

• Dependency Parsing

I Gave John My Address.
Context Free Grammars for Parsing

- $S \rightarrow VP$
- $S \rightarrow NP \ VP$
- $NP \rightarrow Det \ Nom$
- $Nom \rightarrow Noun$
- $Nom \rightarrow Adj \ Nom$
- $VP \rightarrow Verb \ Nom \ Nom$
- $Det \rightarrow “A”, “The”$
- $Noun \rightarrow “I”, “John”, “Address”$
- $Verb \rightarrow “Gave”$
- $Adj \rightarrow “My”, “Blue”$
- $Adv \rightarrow “Quickly”$
Using these rules

• Construct a parse that fits the desired text.
Limitations

- The grammar must be built by hand.
- Can’t handle ungrammatical sentences.
- Can’t resolve ambiguity.
Probabilistic Parsing

- Assign each transition a probability
- Find the parse with the greatest “likelihood”
Next Time

• Input and Output with Matt.