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.

a	q	c	f	k	f	g	h
0	1	2	3	4	5	6	7

Lists!

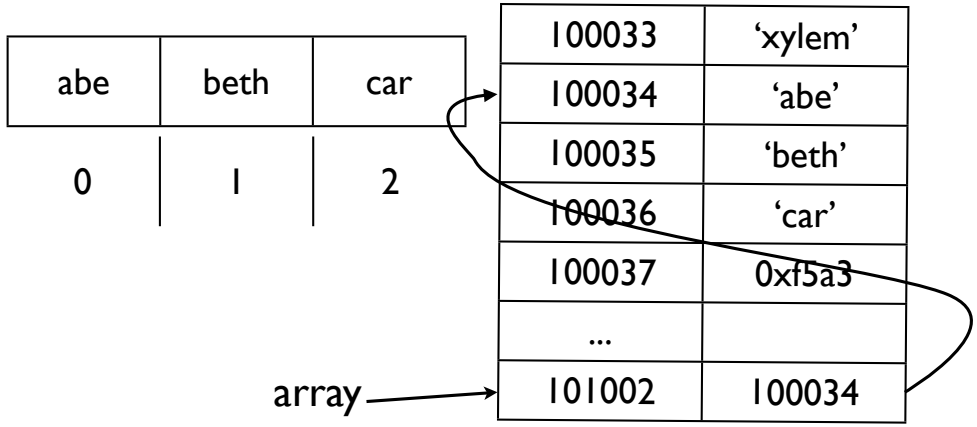
- First and most basic "data structure".
- Describes an ordering of data.

abe	beth	car	45	9.2	bug	i	q
0	1	2	3	4	5	6	7

List demo.

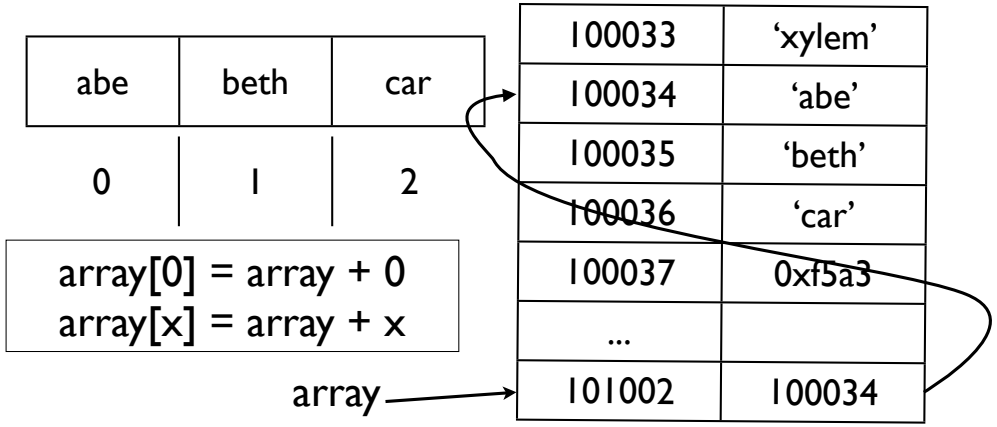
List indexing

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



List indexing

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



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

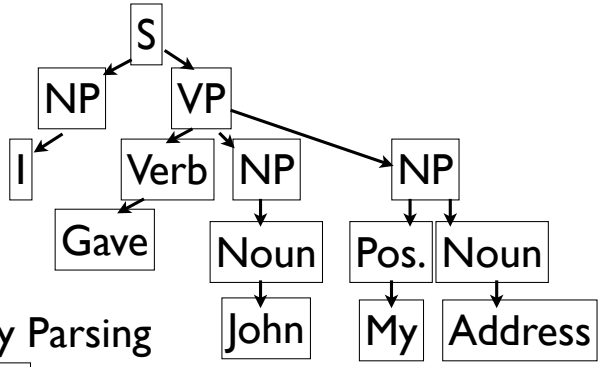
- Lists
  - Iterating over lists
  - for loops
  - while loops
  - range vs xrange
- Comments
- Writing programs.

Parsing

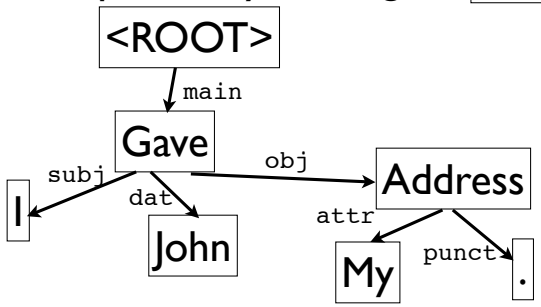
- Generate a Parse Tree from:
  - The surface form (words) of the text
  - Part of Speech Tokens

Parsing Styles

- Parse Trees



- Dependency Parsing



Context Free Grammars for Parsing

- **S → VP**
- **S →NP VP**
- **NP → Det Nom**
- **Nom → Noun**
- **Nom → Adj Nom**
- **VP → Verb Nom Nom**

- **Det** → "A", "The"
- **Noun** → "I", "John", "Address"
- **Verb** → "Gave"
- **Adj** → "My", "Blue"
- **Adv** → "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.