Chapiter_1 Preface

Emphasis

To learn about **advanced algorithms** implemented in NLTK, examine the Python source code and consult other materials cited in this book.

Why Python?

- Shallow learning curve, its syntax and semantics are transparent, and good string-handling functionality.
- As an interpreted language, Python facilitates interactive exploration.
- As an object-oriented language, Python permits data and methods to be encapsulated and re-used easily.
- As a dynamic language, Python permits attributes to be added to objects on the fly, and permits variables to be typed dynamically, facilitating rapid development.
- An extensive standard library, including components for graphical programming, numerical processing, and web connectivity.

V3.0 VS V2.0

Python 3 includes some significant changes(see details here or convert Python 2 code to Python 3 via 2to3.py):

- print statement is now a function, so "**print (...)**";
- many functions now return **iterators** instead of lists (to save memory usage);
- **integer division** returns a floating point number;
- all text is now **Unicode**
- strings are formatted using the **format method**

Software Requirements

- **Python** version 3.2 or later (NLTK 3.0 also works with Python 2.6 and 2.7.)
- **NLTK** version 3.0
- **NLTK-Data** (contains the linguistic corpora)
- **NumPy** (support for multidimensional arrays and linear algebra)
- **Matplotlib** (2D plotting library for data visualization)
- **Stanford NLP Tools** (useful for large scale language processing)
- **NetworkX** (for storing and manipulating network structures consisting of nodes and edges. For visualizing semantic networks, also install the Graphviz library)
- **Prover9** (automated theorem prover for first-order and equational logic, used to support *inference* in language processing)