Python Programming for Linguists

Week 8

Shu-Kai Hsieh

LOPE lab at Graduate Institute of Linguistics,
National Taiwan University

1. Agenda
2. Introduction to Data Structure
3. String
   - Text Processing in Python
4. NLTK and Term Projects

- **[Lecture]**: Review of Iteration; Data Structure (1): String
- **[Praxis]**: In-class exercises with NLTK
- **[Final Projects Pre-proposal discussion and Homework]**
- **[Reading Assignment]**: Think Python: chapter 8-9 (Reading and Scripting)

More on recursion: **break, continue and pass**

- The break Statement in Python terminates the current loop and resumes execution at the next statement.
- The most common use for break is when some external condition is triggered requiring a hasty exit from a loop. The break statement can be used in both while and for loops.

Example

```
for letter in 'Programming':
    if letter == 'a':
        break
    print 'Current letter is:', letter
```

More on recursion: **break, continue and pass**

- The continue statement returns the control to the beginning of the while loop.
- It rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop.
- The continue statement can be used in both while and for loops.

Example

```
for letter in 'Programming':
    if letter == 'a':
        continue
    print 'Current letter is:', letter
```

More on recursion: **break, continue and pass**

- The pass statement is used when a statement is required syntactically but you do not want any command or code to execute.
- The pass statement is a null operation; nothing happens when it executes. The pass is also useful in places where your code will eventually go, but has not been written yet.

Example

```
for letter in 'Programming':
    if letter == 'a':
        pass
        print 'This is pass block'

    print 'Current Letter is:', letter
```

What does it mean by *syntactically required*?

The body of a Python compound statement cannot be empty; it must always contain at least one statement. You can use a pass statement, which performs no action, as a placeholder when a statement is syntactically required but you have nothing to do.

Example

```
if condition1(x):
    process1(x)
elif x>23 or condition2(x) and x<5:
   pass                          # nothing to be done in this case
elif condition3(x):
    process3(x)
else:
    process_default(x)
```

The else Statement Used with Loops

Python supports to have an else statement associated with a loop statements.

- If the else statement is used with a for loop, the else statement is executed when the loop has exhausted iterating the list.
- If the else statement is used with a while loop, the else statement is executed when the condition becomes false.

The else Statement in Loops

One way to think about it is as an if/else construct with respect to the condition:

Example

```
if condition:
    handle_true()
else:
    handle_false()
```

is analogous to the looping construct:

Example

```
while condition:
    handle_true()
else:
   # condition is false now, handle and go on with the rest of the progra
    handle_false()
```

Overview

The overall Python picture is as follows:

- (Python) Programs are composed of **modules** (i.e., a file that contains a collection of related functions and other definitions).
- Modules contain **statements** (i.e., a unit of code that Python interpreter can execute).
- Statements contain **expressions** (i.e., a combination of *values, variables, and operators*).
- Expressions create and process objects (Something a variable can refer to. Also known as data structures).

Very Quick introduction to Objects and Classes

1. We've been generally delaying the discussion of objects and classes till now, a little explanation is needed right now.
2. When you use a variable i and assign a value to it, say integer 5 to it, you can think of it as creating an **object (instance)** *i* of **class (type)** *int*. In fact, you can see `help(int)` to understand this better.
3. A class can also have **methods** i.e. functions defined for use with respect to that class only. You can use these pieces of functionality only when you have an object of that class.

Python's *built-in* **object types** (or data structure types) and some of the syntax used to code their literals - that is, the expressions that generate these objects.

Object type	Example literals/creation
Numbers	12345, 5,66
Strings	'shukai', "python"
Lists	[1,2,3,[2,'we'],7]
Dictionaries	{'food':'great', 'character':'nice'}
Tuples	(1,'great',4,'M')
Files	myfile = open('corpus','r')
Other types	Sets, Booleans

Table: Built-in objects preview. Note that it isn't really complete, because everything we process in Python programs is a kind of objects. Even the File object can be seen as a value that represents an open file.

String

Python's built-in string services provide all of the text-processing functionality you would expect.

- Python's Core string functionality
- Methods of string objects
- Core template classes, and
- Various string formatting methods.
- Text processing using the standard library

String Basics

- A string is a *sequence of characters*.
- Using *single quotes* (') or *double quotes* ('') to specify strings.
- Using Triple Quotes(''' or """) to specify multi-line strings, thus you can use single quotes and double quotes freely within the triple quotes.

Example

```
'''This is a multi-line string. This is the first line.
This is the second line.
 "What's your name?," I asked. He said "Bond, James Bond."
'''
```

Escape sequences

Suppose, you want to have a string which contains a single quote (’), how will you specify this string?

What’s matter?

- So, you will have to specify that this single quote does not indicate the end of the string.
- This can be done with the help of what is called an **escape sequence**. You specify the single quote as `\'` - notice the backslash. Now, you can specify the string as `'What\'s matter?'`.
- Also, you have to indicate the backslash itself using the escape sequence `\\`

Strings more than one line

What if you wanted to specify a two-line string? In addition to *triple quotes*, you can use an escape sequence for the newline character - `\n` to indicate the start of a new line.

Example

This is the first line.`\n` This is the second line.

Another useful escape sequence to know is the tab - `\t`. There are many more escape sequences but I have mentioned only the most useful ones here.

One thing to note is that in a string, a single backslash at the end of the line indicates that the string is continued in the next line, but no newline is added.

For example,

"This is the first sentence.\ This is the second sentence."

is equivalent to

"This is the first sentence. This is the second sentence."

Raw String

If you need to specify some strings where no special processing such as escape sequences are handled,

then what you need is to specify a raw string by prefixing r or R to the string.

An example is `r"Newlines are indicated by \n"`.

Unicode String

1. Unicode is a standard way of writing international text. If you want to write text in your native language such as Hindi or Arabic, then you need to have a Unicode-enabled text editor.
2. Similarly, Python allows you to handle Unicode text - all you need to do is prefix u or U. For example, `u"This is a Unicode string."`.
3. Remember to use Unicode strings when you are dealing with text files, especially when you know that the file will contain text written in languages other than English.

Understanding Encoding and i18n

Ref:

- NLTK book (3.3 Text Processing with Unicode)
- Using Unicode with Python

  `http://openbookproject.net/py4fun/unicode/unicode.html`)

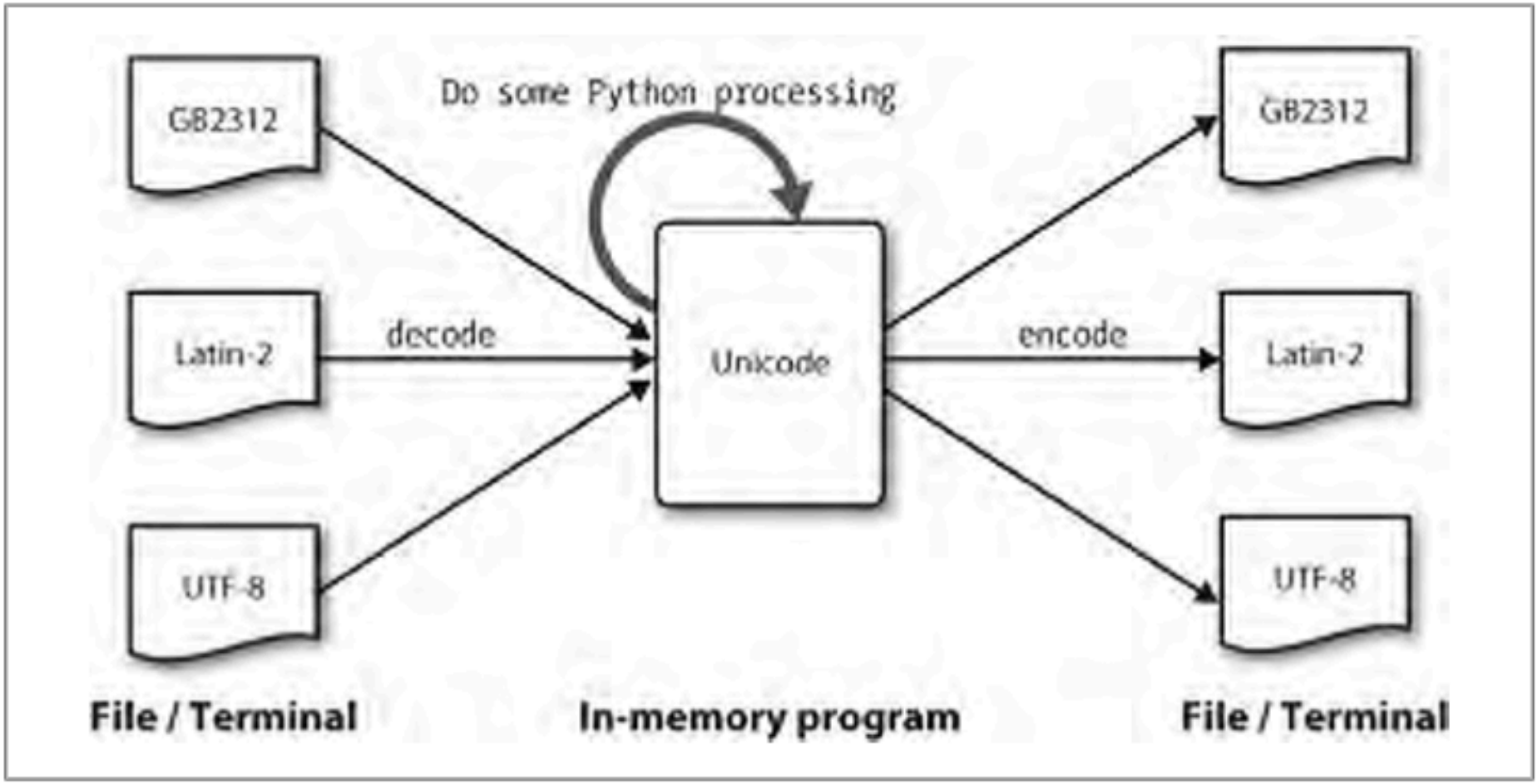

Figure 3-3. Unicode decoding and encoding.

Reading encoded data: Chinese in utf-8

Example

```
import codecs
f = codecs.open('plurkcorpus.txt', encoding='utf-8')
raw = f.read()
print raw
len(raw)
len(set(raw))
```

Reading encoded data: Polish in latin-2

Example

```
import codecs
f = codecs.open('polish-lat2.txt', encoding='latin2')
raw = f.read()
tokens = nltk.word_tokenize(raw)
text = nltk.Text(tokens)
len(text)
```

String Concatenation

Accessing Individual Characters

- Use the same slice notation we used for list.
- The slice `[m,n]` contains the characters from position *m* through *n-1*.

Iterating over the characters in strings

1. We can write loop to iterate over the character strings.
2. `Print` statement ends with a trailing comma tells Python not to print a newline at the end.

Example

```
sent = 'colorless green ideas sleep furiously'
for char in sent:
    print char,
```

Useful string methods

Table 1-4. Some word comparison operators

Function	Meaning
`s.startswith(t)`	Test if s starts with t
`s.endswith(t)`	Test if s ends with t
`t in s`	Test if t is contained inside s
`s.islower()`	Test if all cased characters in s are lowercase
`s.isupper()`	Test if all cased characters in s are uppercase
`s.isalpha()`	Test if all characters in s are alphabetic
`s.isalnum()`	Test if all characters in s are alphanumeric
`s.isdigit()`	Test if all characters in s are digits
`s.istitle()`	Test if s is titlecased (all words in s have initial capitals)

Useful string methods

Method	Functionality
`s.find(t)`	Index of first instance of string `t` inside `s` (`-1` if not found)
`s.rfind(t)`	Index of last instance of string `t` inside `s` (`-1` if not found)
`s.index(t)`	Like `s.find(t)`, except it raises `ValueError` if not found
`s.rindex(t)`	Like `s.rfind(t)`, except it raises `ValueError` if not found
`s.join(text)`	Combine the words of the text into a string using `s` as the glue
`s.split(t)`	Split `s` into a list wherever a `t` is found (whitespace by default)
`s.splitlines()`	Split `s` into a list of strings, one per line
`s.lower()`	A lowercased version of the string `s`
`s.upper()`	An uppercased version of the string `s`
`s.titlecase()`	A titlecased version of the string `s`
`s.strip()`	A copy of `s` without leading or trailing whitespace
`s.replace(t, u)`	Replace instances of `t` with `u` inside `s`

Example

Counting the most frequently occurring letters in a text and plot the distribution.

Example

```
from nltk.corpus import gutenberg
raw = gutenberg.raw('melville-moby_dick.txt')
fdist = nltk.FreqDist(ch.lower() for ch in raw if ch.isalpha())
fdist.keys()
fdist.plot()
```

Strings are immutable

» This means that once you have created a string, you cannot change it. (not necessarily a limitation!).

Because strings are immutable, the string methods are used only to obtain their return value and don't modify the string object they're attached to in any way. → But what if you want to *work with* strings?

The `split()` and `join()` string methods

- `split()` and `join()` methods are invaluable for those who works with strings.
- `split()` returns a list of substrings in the string, and `join()` takes a list of strings and puts them together to form a single string with the original string between each element.
- Typically, `split()` uses whitespace as the delimiter to the strings it's splitting, but you can change that via an optional argument.

The split() and join() string methods

```
>>>" ".join(["join", "puts", "spaces", "between", "elements"])
'join puts spaces between elements'

>>> "::".join(["Separated", "with", "colons"])
'Separated::with::colons'

>>> "".join(["Separated", "by", "nothing"])
'Separatedbynothing'
```

The split() and join() string methods

```
>>> x = "You\t\t can have tabs\t\n \t and newlines \n\n " \
"mixed in"

>>> x.split()
['You', 'can', 'have', 'tabs', 'and', 'newlines', 'mixed', 'in']

>>> x = "Mississippi"

>>> x.split("ss")
['Mi', 'i', 'ippi']
```

Text Processing

Categorizing types of text data

- Structured text: Providing information through markup. `XML`, `HTML`
- Semi-structured text: no explicit markup, but the structure and formats are required (for the parser). `csv`, `email message text`, `JavaScript Object Notation (JSON) web data`, etc.
- Unstructured text. `letter`, `book copy`.

JSON

JSON `http://www.pythonforbeginners.com/python-on-the-web/what-is-json/` (JavaScript Object Notation) is a compact, text based format for computers to exchange data. The official Internet media type for JSON is application/json, and the JSON filename extension is `.json`

- The standard Python distribution includes a powerful set of built-in libraries designed to manage textual content.

- **Babelfish** is an online language translation API provided by Yahoo.
- NLTK comes with a simple interface for using it.(Be sure you are connected to the internet first)

functions provided

```
babelfish.translate(), babelfish.babelize()
babelfish.available_languages
```

Example

```
from nltk.misc import babelfish

# see all the languages for translation
babelfish.available_languages

# specify source and target languages
babelfish.translate('cookbook', 'english', 'french')

# translates back and forth between the source and
# target language
# until there are no more changes.
# it works well only when English is one of them.
for text in babelfish.babelize('cookbook', 'english', 'spanish'
    print text
```

Mid-term Exam

Review the following:

1. HTTLCS Chap 1-8
2. NLTK book Chap 1-3
3. All the exercises and homework

open book, allowance to access internet; 100 min