jamo Documentation

Release 0.4-beta

Joshua Dong

Mar 28, 2017

Contents

1 Hangul Decomposition 3

2 Hangul Synthesis 5

3 Large Texts 7

4 Naming Conventions 9

5 Examples 11

Hangul is a modern writing system that originated in 1443 to represent the Korean language. It uses an alphabet of 24 consonants and vowels, each of which are called **jamo** (,).

Let's analyze Korean phonemes by decomposing some Hangul. Using the individual jamo characters, we can construct some Hangul afterwards.

CHAPTER 1

Hangul Decomposition

The python jamo library aims to provide a straightforward interface to Hangul decomposition:

```
>>> from jamo import h2j
>>> h2j("")
''
```

Notice that the characters may have display issues because they are from the U+11xx jamo code block. This is because there are actually two sets of jamo in Unicode. Computers use jamo from the U+31xx code block, known as **Hangul Compatibility Jamo**, here on referenced as *HCJ*. To render HCJ instead of U+11xx jamo:

```
>>> from jamo import h2j, j2hcj
>>> j2hcj(h2j(""))
''
>>> j2hcj(h2j("==jamo"))
'==jamo'
```

Here we convert the Hangul characters to U+11xx jamo characters, then convert them to HCJ for more uniform display.

If you are curious, learn more about the differences between U+11xx and U+31xx jamo at unicode_tutorial. Related, Gernot Katzers has an excellent writeup on Hangul representation in unicode that is well worth a read.

CHAPTER 2

Hangul Synthesis

Hangul synthesis combines a lead, vowel, and optional tail to form a single jamo character:

```
>>> from jamo import j2h
>>> j2h('', '', '')

>>> j2h('', '')

```

A little hack you can use is the splat operator * if your arguments are in string form:

```
>>> j2h(*'')

>>> j2h(*'')

```

CHAPTER 3

Large Texts

When working with large files, we will end up with lots of output. To handle large files, it is recommended to use the provided generator functions:

```
>>> from jamo import hangul_to_jamo
>>> long_story = open(".txt", 'r').read()
>>> hangul_to_jamo(long_story)
<generator object <genexpr> at 0xdeadbeef9001>
```

To produce HCJ output:

```
>>> from jamo import hangul_to_jamo, hangul_to_hcj
>>> long_story = open(".txt", 'r').read()
>>> hangul_to_hcj(hangul_to_jamo(long_story))
<generator object <genexpr> at 0x12cafebabe34>
```

CHAPTER 4

Naming Conventions

The python-jamo module is designed to be simple and lightweight. There are no classes to wrap Hangul strings or jamo characters. Below are two important string generator pairs:

Generator Function	String Function
jamo_to_hcj	j2hcj
hangul_to_jamo	h2j

Note that most functions in the module are named in pairs, where the function with the shorter name is the one best for casual use, and the function with the longer name returns a generator and is probably better for analytic applications.

Module output favors characters whenever possible.

CHAPTER 5

Examples

Basic examples: sample_usage.