Case conversion functions are currently limited to ASCII characters. When used with formal languages, that may be appropriate, but it is often not appropriate for applications.

In order to avoid backwards-compatibility problems and to make sure that the various variants of case conversion (e.g. language-dependent) can be selected, we propose to add an optional parameter to the case conversion functions.

Our current design idea is as follows:

ASCII-only if no parameter:
'Türkiye'.upcase    # => 'TÜRKİYE', note lower-case ü

Parameter triggers (general) Unicode conversion:
'Türkiye'.upcase 'en'  # => 'TÜRKIYE', note upper-case Ü

This means that for languages with special case conversion rules, such as Turkish, this works as follows:
'Türkiye'.upcase 'tr'  # => 'TÜRKİYE', note upper-case İ (with dot!)

In the second example, we used 'en', but most other languages would work, too, because a single case conversion works for most languages. Turkic languages are the biggest exception.

The Unicode standard also defines various cases of "case-folding", which usually is lossy, e.g. mapping German ß to ss and so on. It should be possible to include this functionality in this proposal, e.g. by using symbols or CONSTANTS for the few specific foldings. It may also be possible to define a reversible variant of case conversion in particular for use with swapcase.

In the long term, instead of a direct BCP 47 string, we could create a Locale class that would incorporate language-specific facilities, but this may need more detailed considerations.

The idea of using an additional parameter to indicate language-dependent or other processing variants should be extensible to areas such as number-to-string conversion and date formation. While this proposal is only about case conversion, we should check that there is a good chance to use similar parameter conventions for such extensions.

[This proposal is based on research done together with my student Kimihito Matsui.]
I want default case conversion should be Unicode aware (when encoding is Unicode).
The previous behavior can be done by str.downcase(:ascii).

Non unicode encoding (e.g. Latin-1) can support non ASCII case conversion, but not mandatory.

Matz.

The related issue, [Bug #10550] Resolv::DNS.getaddresses returns no IPs when nameserver returns in differing case than query

The related issue, [Bug #11284] String#upcase and String#downcase don't work for accented characters

Close way overdue, should have happened somewhere around r55281.

Files
CaseConversion.pdf 340 KB 07/23/2014 duerst (Martin Dürst)