6.5. `unicodedata` — Unicode Database

This module provides access to the Unicode Character Database (UCD) which defines character properties for all Unicode characters. The data contained in this database is compiled from the UCD version 9.0.0.

The module uses the same names and symbols as defined by Unicode Standard Annex #44, "Unicode Character Database". It defines the following functions:

`unicodedata.`**`lookup`**(*name*)

Look up character by name. If a character with the given name is found, return the corresponding character. If not found, `KeyError` is raised.

Changed in version 3.3: Support for name aliases [1] and named sequences [2] has been added.

`unicodedata.`**`name`**(*chr*[, *default*])

Returns the name assigned to the character *chr* as a string. If no name is defined, *default* is returned, or, if not given, `ValueError` is raised.

`unicodedata.`**`decimal`**(*chr*[, *default*])

Returns the decimal value assigned to the character *chr* as integer. If no such value is defined, *default* is returned, or, if not given, `ValueError` is raised.

`unicodedata.`**`digit`**(*chr*[, *default*])

Returns the digit value assigned to the character *chr* as integer. If no such value is defined, *default* is returned, or, if not given, `ValueError` is raised.

`unicodedata.`**`numeric`**(*chr*[, *default*])

Returns the numeric value assigned to the character *chr* as float. If no such value is defined, *default* is returned, or, if not given, `ValueError` is raised.

`unicodedata.`**`category`**(*chr*)

Returns the general category assigned to the character *chr* as string.

`unicodedata.`**`bidirectional`**(*chr*)

Returns the bidirectional class assigned to the character *chr* as string. If no such value is defined, an empty string is returned.

`unicodedata.`**`combining`**(*chr*)

Returns the canonical combining class assigned to the character *chr* as integer. Returns `0` if no combining class is defined.

`unicodedata.`**`east_asian_width`**(*chr*)

Returns the east asian width assigned to the character *chr* as string.

`unicodedata.`**`mirrored`**(*chr*)

Returns the mirrored property assigned to the character *chr* as integer. Returns `1` if the character has been identified as a “mirrored” character in bidirectional text, `0` otherwise.

`unicodedata.`**`decomposition`**(*chr*)

Returns the character decomposition mapping assigned to the character *chr* as string. An empty string is returned in case no such mapping is defined.

`unicodedata.`**`normalize`**(*form*, *unistr*)

Return the normal form *form* for the Unicode string *unistr*. Valid values for *form* are ‘NFC’, ‘NFKC’, ‘NFD’, and ‘NFKD’.

The Unicode standard defines various normalization forms of a Unicode string, based on the definition of canonical equivalence and compatibility equivalence. In Unicode, several characters can be expressed in various way. For example, the character U+00C7 (LATIN CAPITAL LETTER C WITH CEDILLA) can also be expressed as the sequence U+0043 (LATIN CAPITAL LETTER C) U+0327 (COMBINING CEDILLA).

For each character, there are two normal forms: normal form C and normal form D. Normal form D (NFD) is also known as canonical decomposition, and translates each character into its decomposed form. Normal form C (NFC) first applies a canonical decomposition, then composes pre-combined characters again.

In addition to these two forms, there are two additional normal forms based on compatibility equivalence. In Unicode, certain characters are supported which normally would be unified with other characters. For example, U+2160 (ROMAN NUMERAL ONE) is really the same thing as U+0049 (LATIN CAPITAL LETTER I). However, it is supported in Unicode for compatibility with existing character sets (e.g. gb2312).

The normal form KD (NFKD) will apply the compatibility decomposition, i.e. replace all compatibility characters with their equivalents. The normal form KC (NFKC) first applies the compatibility decomposition, followed by the canonical composition.

Even if two unicode strings are normalized and look the same to a human reader, if one has combining characters and the other doesn’t, they may not compare equal.

In addition, the module exposes the following constant:

`unicodedata.`**`unidata_version`**

The version of the Unicode database used in this module.

`unicodedata.`**`ucd_3_2_0`**

This is an object that has the same methods as the entire module, but uses the Unicode database version 3.2 instead, for applications that require this specific version of the Unicode database (such as IDNA).

Examples:

```
>>> import unicodedata
>>> unicodedata.lookup('LEFT CURLY BRACKET')
'{'
>>> unicodedata.name('/')
'SOLIDUS'
>>> unicodedata.decimal('9')
9
>>> unicodedata.decimal('a')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: not a decimal
>>> unicodedata.category('A')  # 'L'etter, 'u'ppercase
'Lu'
>>> unicodedata.bidirectional('\u0660') # 'A'rabic, 'N'umber
'AN'
```

Footnotes

[1] http://www.unicode.org/Public/9.0.0/ucd/NameAliases.txt

[2] http://www.unicode.org/Public/9.0.0/ucd/NamedSequences.txt