



Модуль unicodedata в Python, база данных Unicode



vag-academy.ru

РЕКЛАМА • 16+

Отличайся от других эзотериков! Уникальный метод PLPT



Узнать больше

[Стандартная библиотека Python3.](#) / Модуль unicodedata в Python, база данных Unicode

База данных Unicode

Модуль unicodedata обеспечивает доступ к базе данных символов Unicode (UCD), которая определяет свойства символов для всех символов Unicode.

[Модуль unicodedata](#) использует те же имена и символы, которые определены в Стандартном приложении Unicode № 44 "База данных символов Unicode" .

Изменено в Python 3.11. База данных Unicode обновлена до версии 14.0.0.

Содержание:

- [Синтаксис модуля](#),
- Функция [unicodedata.lookup\(\)](#),
- Функция [unicodedata.name\(\)](#),
- Функция [unicodedata.decimal\(\)](#),
- Функция [unicodedata.digit\(\)](#),
- Функция [unicodedata.numeric\(\)](#),
- Функция [unicodedata.category\(\)](#),
- Функция [unicodedata.bidirectional\(\)](#),
- Функция [unicodedata.combining\(\)](#),
- Функция [unicodedata.east_asian_width\(\)](#),
- Функция [unicodedata.mirrored\(\)](#),
- Функция [unicodedata.decomposition\(\)](#),
- Функция [unicodedata.normalize\(\)](#),
- Функция [unicodedata.is_normalized\(\)](#),
- Константа [unicodedata.unidata_version](#),
- Класс [unicodedata.ucd_3_2_0](#).

Синтаксис:

```
import unicodedata
unicodedata.function_name()

import unicodedata as ucd
ucd.function_name()

from unicodedata import function_name
function_name()
```

Модуль определяет следующие функции:

unicodedata.lookup(name):

Функция lookup() производит поиск символа Unicode по имени. Если найден символ с указанным name именем, то lookup() вернет соответствующий символ. Если не найден, то поднимается [исключение KeyError](#).

```
>>> from unicodedata import lookup
>>> lookup('CYRILLIC SMALL LETTER A')
# 'а'
```

unicodedata.name(chr[, default]):

Функция `name()` возвращает имя, присвоенное символу `chr` в виде [строки](#). Если имя не определено, возвращается значение, заданное по умолчанию `default` или, если не указано, вызывается [исключение ValueError](#).

```
>>> from unicodedata import name
>>> name('А')
# 'CYRILLIC CAPITAL LETTER A'
>>> name('1')
# 'DIGIT ONE'
```

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

Функция `decimal()` возвращает десятичное значение, назначенное символу `chr`, как [целое число](#). Если такое значение не определено, возвращается значение, заданное по умолчанию или, если не задано, вызывается [исключение ValueError](#).

```
>>> from unicodedata import decimal
>>> decimal('7')
# 7
>>> decimal('a', False)
# False
```

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

Функция `digit()` возвращает цифровое значение, назначенное символу `chr`, как [целое число](#). Если такое значение не определено, возвращается значение, заданное по умолчанию или, если не задано, вызывается [исключение ValueError](#).

```
>>> from unicodedata import digit
>>> digit('9')
#
>>> digit('f', False)
# False
```

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

Функция `numeric()` возвращает числовое значение, назначенное символу `chr`, как [float](#). Если такое значение не определено, возвращается значение, заданное по умолчанию или, если не задано, вызывается [исключение ValueError](#).

```
>>> from unicodedata import numeric
>>> numeric('5')
# 5.0
>>> numeric('z', False)
# False
```

`unicodedata.category(chr):`

Функция `category()` возвращает общую категорию, назначенную символу `chr` в виде [строки](#).

```
>>> from unicodedata import category
>>> category('3')
# 'Nd'
>>> category('z')
# 'Ll'
```

`unicodedata.bidirectional(chr):`

Функция `bidirectional()` возвращает двунаправленный класс, назначенный символу `chr` в виде [строки](#). Если такое значение не определено, возвращается пустая строка.

```
>>> from unicodedata import bidirectional
>>> bidirectional('1')
# 'EN'
>>> bidirectional('+')
# 'ES'
```

`unicodedata.combining(chr):`

Функция `combining()` возвращает класс канонического объединения, назначенный символу `chr`, как [целое число](#). Возвращает 0, если класс объединения не определен.

```
>>> from unicodedata import combining
>>> combining('9')
# 0
```

```
>>> combining('=')  
# 0
```

unicodedata.east_asian_width(chr):

Функция `east_asian_width()` возвращает восточно-азиатскую ширину, назначенную символу `chr` в виде [строки](#).

```
>>> from unicodedata import east_asian_width  
>>> east_asian_width('V')  
# 'Na'  
>>> east_asian_width('Ю')  
# 'A'
```

unicodedata.mirrored(chr):

Функция `mirrored()` возвращает зеркальное свойство, назначенное символу `chr`, как [целое число](#). Возвращает 1, если символ был идентифицирован как "зеркальный" в двунаправленном тексте, 0 в противном случае.

```
>>> from unicodedata import mirrored  
>>> mirrored('=')  
# 0
```

unicodedata.decomposition(chr):

Функция `decomposition()` возвращает отображение декомпозиции символа, назначенное символу `chr` в виде [строки](#). Возвращается пустая строка, если такое отображение не определено.

```
>>> from unicodedata import decomposition  
>>> decomposition('=')  
# ''  
>>> decomposition('ϕ')  
# ''
```

unicodedata.normalize(form, unistr):

Функция `normalize()` возвращает нормальную форму `form` для строки Unicode `unistr`. Допустимые значения для формы: NFC, NFKC, NFD и NFKD.

Стандарт Unicode определяет различные формы нормализации строки Unicode на основе определения канонической эквивалентности и эквивалентности совместимости. В Unicode несколько символов могут быть выражены по-разному. Например, символ U + 00C7 (LATIN CAPITAL LETTER C WITH CEDILLA) также может быть выражен как последовательность U + 0043 (LATIN CAPITAL LETTER C) U + 0327 (COMBINING CEDILLA).

Для каждого символа есть две нормальные формы: нормальная форма C и нормальная форма D. Нормальная форма D (NFD) также известна как каноническое разложение и переводит каждый символ в разложенную форму. Нормальная форма C (NFC) сначала применяет каноническую декомпозицию, затем снова создает предварительно объединенные символы.

В дополнение к этим двум формам существуют две дополнительные нормальные формы, основанные на эквивалентности совместимости. В Unicode поддерживаются определенные символы, которые обычно объединяются с другими символами. Например, U + 2160 (ROMAN NUMERAL ONE) действительно то же самое, что U + 0049 (LATIN CAPITAL LETTER I). Тем не менее, он поддерживается в Unicode для совместимости с существующими наборами символов, например gb2312.

Нормальная форма KD (NFKD) будет применять декомпозицию совместимости, то есть заменять все символы совместимости их эквивалентами. Нормальная форма KC (NFKC) сначала применяет декомпозицию совместимости, а затем каноническую композицию.

Даже если две строки Unicode нормализованы и выглядят одинаково для человека, если одна имеет комбинированные символы, а другая нет, они могут не совпадать.

```
>>> from unicodedata import normalize  
>>> normalize('NFC', 'Й')  
# 'Й'  
>>> normalize('NFD', 'Й')  
# 'Й'
```

unicodedata.is_normalized(form, unistr):

Функция `is_normalized()` проверяет, возвращает ли Unicode строка `unistr` в нормальной форме `form`. Допустимые значения для формы: NFC, NFKC, NFD и NFKD.

```
>>> from unicodedata import is_normalized  
>>> is_normalized('NFD', 'Й')
```

```
# False
>>> is_normalized('NFD', 'ϕ')
# True
```

unicodedata.unidata_version:

Константа unidata_version возвращает Версия базы данных Unicode, используемая в этом модуле.

```
>>> from unicodedata import unidata_version
>>> unidata_version
# '9.0.0'
```

unicodedata.ucd_3_2_0:

Класс ucd_3_2_0 - это объект, который имеет те же методы, что и весь модуль, но вместо этого использует базу данных Unicode версии 3.2 для приложений, которым требуется эта конкретная версия базы данных Unicode (например, IDNA).

```
>>> from unicodedata import ucd_3_2_0
>>> ucd_3_2_0.name('э', False)
# 'CYRILLIC SMALL LETTER E'
>>> ucd_3_2_0.name('е', False)
# 'CYRILLIC SMALL LETTER IE'
```

Примеры использования:

```
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

# 'L'etter, 'u'ppercase
unicodedata.category('A')
# 'Lu'

# 'A'rabic, 'N'umber
unicodedata.bidirectional('\u0660')
# 'AN'
```


ХОЧУ ПОМОЧЬ
ПРОЕКТУ

РЕКЛАМА • 16+ Я



Python

Яндекс Практикум

practicum.yandex.ru

Профессия: Python-разработчик. Курс от Яндекса.

5,0 ★ Рейтинг организации ⓘ

Поможем освоить новую профессию с нуля за 9 месяцев. Начните учиться бесплатно!

Курсы программирования

Трудоустройство

Программа

Узнать больше