


ХОЧУ ПОМОЧЬ
ПРОЕКТУ

Модуль unicode в Python, unicode to ASCII

24

bitrix24.ru

РЕКЛАМА



Интеграция СДЭК и Битрикс24

Приложение СДЭК для Битрикс24. Создание и контроль накладных. Выплаты по накладке

Попробовать

Статья 3. / Модуль unicode в Python, unicode to ASCII

Проблема с символами в символах ASCII

Часто приходится работать с текстовыми данными в Unicode, и их нужно представить в ASCII. Например, при интеграции с сервисом, который поддерживает Unicode, или для упрощения ввода нелатинских имен на клавиатуре, или при создании URL-адресов из удобочитаемых строк Unicode, которые должны быть несколько понятными. Проблема возникает при создании URL-адресов из заголовка статьи.

[Модуль unicode](#) не является заменой полной поддержки Unicode для строк. Есть ряд предостережений, связанных с его использованием, особенно когда его вывод непосредственно виден пользователям. Прежде чем использовать модуль unicode в своем проекте, прочтите оставшуюся часть этого материала.

Примечание. Обратите внимание на [модуль transliterate](#), который представляет собой двунаправленный транслитератор текста заточенный под славянские языки, такие как русский, сербский болгарский украинский и т.д.

Установка модуля unicode в виртуальное окружение.

Модуль unicode размещен на PyPI, поэтому установка относительно проста.

```
# создаем виртуальное окружение, если нет
$ python3 -m venv .venv --prompt VirtualEnv
# активируем виртуальное окружение
$ source .venv/bin/activate
# обновляем `pip`
(VirtualEnv):~$ python3 -m pip install -U pip
# ставим модуль `unicode`
(VirtualEnv):~$ python3 -m pip install -U unicode
```

Содержание:

- [Основные моменты, связанные с unicode;](#)
- [Функциональность модуля unicode;](#)
 - [Транслитерация текста из командной строки;](#)
- [Часто задаваемые вопросы.](#)

Основные моменты, связанные с unicode.

Модуль unicode принимает [строки](#) в Юникоде и пытается представить их в символах ASCII (т.е. универсально отображаемых символах между 0x00 и 0x7F), где компромиссы, принимаемые при сопоставлении между двумя наборами символов, выбираются так, чтобы они были близки к тому, что сделал бы человек на клавиатуре с раскладкой США.

Качество результирующего представления ASCII варьируется. Для языков западного происхождения оно должно быть между идеальным и хорошим. С другой стороны, транслитерация (т. е. передача латинскими буквами произношения, выраженного текстом в какой-либо другой системе письма) таких языков, как китайский, японский или корейский, является очень сложной проблемой, и этот модуль даже не пытается ее решить. Он придерживается линии контекстно-свободного посимвольного сопоставления. Таким образом, хорошее эмпирическое правило заключается в том, что чем дальше находится транслитерируемый язык от латинского алфавита, тем хуже будет транслитерация.

Как правило, модуль unicode дает лучшие результаты, чем просто удаление акцентов из символов (что можно сделать в Python с помощью встроенных функций). Он основан на настроенных вручную сопоставлениях символов, которые, например, также содержат приближения ASCII для символов и нелатинских алфавитов.

Вверх

Обратите внимание, что некоторые люди могут счесть определенные транслитерации оскорбительными. Наиболее распространенные примеры включают символы, которые используются в нескольких языках. Пользователь ожидает, что символ будет транслитерирован на его языке, но модуль `unicode` использует транслитерацию для другого языка. Лучше не использовать этот модуль для строк, которые видны непосредственно пользователям приложения.

⋮

Функциональность модуля `unicode`.

Модуль `unicode` содержит функцию, которая принимает строковый объект, возможно, содержащий не-ASCII-символы, и возвращает строку, которую можно безопасно закодировать в ASCII:

```
>>> from unicode import unicode
>>> unicode('Какой-то текст')
# 'Kakoi-to tekst'
>>> unicode('Вячеслав, Юлия, Андрей, София')
'Viacheslav, Iuliia, Andrei, Sofiia'
>>> unicode('kožušček')
# 'kozáscek'
>>> unicode('30 \U0001d5c4\U0001d5c6/\U0001d5c1')
# '30 km/h'
>>> unicode('\u5317\u4EB0')
# 'Bei Jing '
```

Также, для функции `unicode.unicode()` можно указать аргумент `errors`, который будет определять, что делать с символами, которых нет в таблицах транслитерации модуля. Значение по умолчанию - `errors='ignore'`, что означает, что модуль `unicode` будет игнорировать эти символы (заменит их пустой строкой). Значение `errors='strict'` вызовет ошибку `UnicodeError`. Объект исключения будет содержать атрибут `.index`, который можно использовать для поиска символа-нарушителя. Значение `errors='replace'` заменит их на '?' (или другую строку, указанную в аргументе `replace_str`). Значение `errors='preserve'` сохранит исходный символ, отличный от ASCII, в строке.

Обратите внимание, что если используется 'preserve', то строка, возвращаемая `unicode.unicode()`, не будет кодироваться в ASCII!:

```
# unicode не имеет замены для символов частной зоны
>>> unicode('\ue000')
# ''
>>> unicode('\ue000', errors='strict')
# Traceback (most recent call last):
# ...
# unicode.UnicodeError: no replacement found for character '\ue000' in position 0
```

Транслитерация текста из командной строки.

Модуль `unicode` содержит CLI и утилиту, которые позволяют транслитерировать текст из командной строки несколькими способами.

Чтение со стандартного ввода:

```
$ echo 'какой-то текст' | unicode
# kakoi-to tekst
```

Из параметра командной строки:

```
$ unicode -c 'какой-то текст'
# kakoi-to tekst
```

Или из файла:

```
$ unicode test.txt
# kakoi-to tekst
```

Кодировка по умолчанию, используемая утилитой, зависит от языкового стандарта используемой операционной системы. Можно указать другую кодировку с помощью параметра CLI `-e`.

Полный список доступных опций смотрите в выводе команды.

```
$ unicode --help
```

Вверх

Часто задаваемые вопросы.

- Немецкие умлауты транслитерируются неправильно.

Латинские буквы 'ä', 'ö' и 'ü' с диэрезисом транслитерируются Unicode как 'a', 'o', 'u', а не в соответствии с немецкими правилами 'ae', 'oe', 'ue'. Это сделано намеренно и не будет меняться в дальнейшем. Обоснование заключается в том, что эти буквы используются на других языках, кроме немецкого (например, на финском и турецком). Текст на немецком языке, транслитерированный без дополнительной буквы "e", гораздо более удобочитаем, чем текст на других языках, транслитерированный по немецким правилам. Обходной путь заключается в том, чтобы выполнить собственные замены этих символов перед передачей строки в функцию `unicode.unidecode()`.

Пример функции замены:

```
def de_to_ascii(text):
    chs = {'ß': 'ss', 'ä': 'ae', 'ö': 'oe', 'ü': 'ue'}
    for ch, val in chs.items():
        if ch in text:
            text = text.replace(ch, val)
    return text
```

- Японский кандзи транслитерируется как китайский.

Как и в случае с латинскими буквами с акцентами, обсуждаемыми в ответе выше, стандарт Unicode кодирует буквы, а не буквы определенного языка или их значение. С японским и китайским это еще более очевидно, потому что одна и та же буква может иметь очень разные транслитерации в зависимости от языка, в котором она используется. Так как модуль `unidecode` не выполняет транслитерацию для конкретного языка (см. следующий вопрос), он должен выбрать одну парадигму. Для некоторых символов, которые используются как в японском, так и в китайском языках, было принято решение использовать китайскую транслитерацию. Если необходимо транслитерировать японский, китайский или корейский текст, то рассмотрите возможность использования других модулей, которые выполняют транслитерацию для конкретного языка, например [unihandecode](#).

- Unicode поддерживает локализацию? (например, параметр языка или страны, проверка локали системы и т. д.)

Учитывать конкретный язык при транслитерации - сложная задача, которая выходит за рамки этого модуля.

- Unicode автоматически определяет язык транслитерируемого текста?

Определение языка - это совершенно отдельная проблема, выходящая за рамки этой библиотеки.

- Почему Unicode не заменяет символы обратной косой черты \u и \U в строках?

Модуль `unidecode` ничего не знает об экранирующих последовательностях. Интерпретация этих последовательностей и замена их фактическими символами Юникода в строковых литералах является задачей интерпретатора Python.

