


# Строковые и байтовые литералы



eon-energydrink.com

РЕКЛАМА

Больше информации на сайте  
рекламодателя

Узнать больше

[Справочник по языку Python3.](#) / Строковые и байтовые литералы

[Как строковые, так и байтовые литералы](#) могут быть заключены в соответствующие одинарные кавычки (') или двойные кавычки ("). Они также могут быть заключены в соответствующие группы из трех одинарных или двойных кавычек ('''') или (""""), они обычно называются строки в тройных кавычках.

Символ обратной косой черты (\) используется для экранирования символов, которые имеют особое значение, такие как например новая строка \n.

Байтовые литералы всегда начинаются с префикса 'b' или 'B', они производят экземпляр [bytes тип](#) вместо [str типа](#). Они могут содержать только символы ASCII. Байты с числовым значением 128 или более должны быть выражены с помощью экранирования \.

Как строковые, так и байтовые литералы могут дополнительно иметь префикс в виде буквы 'r' или 'R'. Такие строки называются необработанными (сырыми) строками и обрабатывают обратную косую черту как символ литерала. В результате, в сырых строках, экранирование символов Юникода '\u' и '\u' специально ни как не обрабатываются и учитывая, что необработанные литералы Unicode в Python-2.x ведут себя иначе, чем в Python-3.x, синтаксис 'ur' не поддерживается.

Новое в Python-3.3: 'rb' префикс сырых байтов литералов был добавлен как синоним 'br'.

Новое в Python-3.3: Поддержка устаревшего литерала Юникода (u'value') была вновь введена для упрощения обслуживания двух кодовых баз Python-2.x и Python-3.x.

Строковый литерал с префиксом ['f' или 'F' является форматированным строковым литералом](#). Префикс 'f' может быть объединен с префиксом 'r', но не может быть объединен с префиксом 'b' или 'u'.

В тройных кавычках допускаются и сохраняются ескаре-последовательность новой строки \n и кавычки, за исключением того, что три неэкранированные кавычки в строке завершают литерал. "Кавычка" - это символ, используемый для открытия литерала, т. е. либо ' или ".

```
>>> """Открывать" строку 4 кавычки могут"""
# "'Открывать" строку 4 кавычки могут'
>>> """А вот закрывать строку подряд "4 кавычки не могут"""
# File "<stdin>", line 1
#      """А вот закрывать строку подряд "4 кавычки не могут"""
#
# SyntaxError: EOL while scanning string literal
>>> """В этом случае кавычки нужно экранировать \"обратным слешем\""""
# 'В этом случае кавычки нужно экранировать "обратным слешем"'
```

Если префикс 'r' или 'R' отсутствует, [escape-последовательности](#) в строковых и байтовых литералах интерпретируются в соответствии с правилами, аналогичными тем, которые используются стандартом языка C.

В сырой строке кавычки можно экранировать с помощью обратной косой черты, но в результате обратная косая черта останется. Например, строковый литерал `r"\"`, состоящий из двух символов - обратной косой черты и двойной кавычки будет интерпретироваться в `'\"'`. В сырых строках **необработанный литерал не может заканчиваться одним обратным слешем**, так как обратный слеш будет экранировать следующий символ кавычки. Также обратите внимание, что `escape-последовательность` новой строки `\n`, интерпретируется как `\\n`, а не как новая строка.

Пример особенностей сырых строк:

```
# Необработанные (сырые) строки
>>> r"пробуем экранировать \" кавычку"
# 'пробуем экранировать \" кавычку'
>>> print(r"пробуем экранировать \" кавычку")
# пробуем экранировать \" кавычку

>>> r"escape-последовательность новой \nстроки не работает"
'escape-последовательность новой \\nстроки не работает'
>>> print(r"escape-последовательность новой \nстроки не работает")
# escape-последовательность новой \nстроки не работает


# Простые строки
>>> "пробуем экранировать \" кавычку"
# 'пробуем экранировать " кавычку'
>>> print("пробуем экранировать \" кавычку")
# пробуем экранировать " кавычку


>>> "escape-последовательность новой \nстроки работает как положено"
'escape-последовательность новой \nстроки работает как положено'
>>> print("escape-последовательность новой \nстроки работает как положено")
# escape-последовательность новой
# строки работает как положено
```

Содержание раздела:
<ul style="list-style-type: none"><li><a href="#">КРАТКИЙ ОБЗОР МАТЕРИАЛА.</a></li><li><a href="#">escape-последовательности</a></li></ul>

ХОЧУ ПОМОЧЬ ПРОЕКТУ

РЕКЛАМА



 ddos-guard.net

Мощная и надежная защита от DDoS-атак для вашего сайта!

от 8 000 ₺

Надежная защита от всех известных видов DDoS-атак. Легкий переезд. Гибкие тарифы.

Защищенный хостинг

Защищенные VDS

Выделенные серверы

