


Операции, доступные bytes и bytearray типам



 meb-expo.ru

Выставка «Мебель-2023»

Поставщики • Производители • Выгодные условия • Новинки индустрии
Лидеры рынка

Узнать больше

РЕКЛАМА · 16+

[Справочник по языку Python3.](#) / Операции, доступные bytes и bytearray типам

[Байтовые строки bytes](#) и [объекты bytearray](#) поддерживают [общие операции с последовательностями](#). Они взаимодействуют не только с операндами одного типа, но и с любыми байтовыми объектами, которые поддерживают буферный протокол и могут экспортировать непрерывный C-буфер, например [memoryview](#). Благодаря такой гибкости они могут свободно смешиваться в операциях, не вызывая ошибок. Однако тип возвращаемого результата может зависеть от порядка операндов.

Важно: [Методы в байтовых строках bytes и bytearray объектах](#) не принимают [текстовые строки](#) в качестве своих аргументов, так же как [методы текстовых строк](#) не принимают байты в качестве своих аргументов. Например, вы должны написать:

```
# Для типа 'str' - текстовых строк
a = "abc"
b = a.replace("a", "f")

# Для типа 'bytes' - байтовых строк
a = b"abc"
b = a.replace(b"a", b"f")
```

Некоторые операции с bytes и bytearray предполагают использование ASCII-совместимых двоичных форматов, и, следовательно, их следует избегать при работе с произвольными двоичными данными.

Важно: Использование операций на основе ASCII для управления двоичными данными, которые не хранятся в формате на основе ASCII, может привести к повреждению данных.

Представленные ниже методы для [байтовых строк bytes и bytearray объектов](#) могут использоваться с произвольными двоичными данными:

[Метод count\(\) типов bytes и bytearray в Python](#)

Метод `count()` возвращает число неперекрывающихся вхождений подпоследовательности `sub` в диапазоне `[start, end]`. Необязательные аргументы `start` и `end` интерпретируются как в нотации среза.

[Метод decode\(\) типов bytes и bytearray в Python](#)

Метод `decode()` возвращает строку, декодированную из заданных байтов. Кодировка `encoding` по умолчанию - `'utf-8'`. Обработчик ошибок `errors` может быть задан для другой схемы обработки ошибок.

[Метод endswith\(\) типов bytes и bytearray в Python](#)

Метод `endswith()` возвращает `True`, если двоичные данные заканчиваются указанным суффиксом `suffix`, в противном случае возвращает `False`.

[Метод find\(\) типов bytes и bytearray в Python](#)

Метод `find()` возвращает индекс первого совпадения байтовой подпоследовательности `sub`, такой, что `sub` содержится в срезе `s[start: end]`.

[Метод index\(\) типов bytes и bytearray в Python](#)

Метод типов bytes и bytearray `index()` работает так же как и метод `find()`, за исключением того, что если байтовая подстрока не найдена, поднимается исключение `ValueError`.

[Метод join\(\) типов bytes и bytearray в Python](#)

Метод `join()` возвращает новый `bytes` или `bytearray` объект, который является конкатенацией последовательности бинарных данных в объекте, поддерживающим итерацию `iterable`.

[Метод maketrans\(\) типов bytes и bytearray в Python](#)

Статический метод `maketrans()` создает таблицу преобразования символов для метода `bytes.translate()`, который будет отображать каждый символ в `from` в символ в той же позиции в `to`.

[Метод translate\(\) типов bytes и bytearray в Python](#)

Метод `translate()` возвращает копию объекта `bytes` или `bytearray`, где все байты, встречающиеся в необязательном аргументе `delete`, удаляются, а оставшиеся байты были сопоставлены через заданную таблицу преобразования `table`, которая должна быть

[Метод partition\(\) типов bytes и bytearray в Python](#)

Метод `partition()` делит последовательность при первом появлении `sep` и вернет кортеж из трех значений, которые содержат часть перед разделителем, сам разделитель или его копию в виде `bytearray` и часть после разделителя.

[Метод removesuffix\(\) типов bytes и bytearray в Python](#)

Если исходная байтовая строка заканчивается с байтовой строки `suffix`, то метод `bytes.removeprefix()` возвращает байтовую строку без суффикса `bytes[len(suffix):]`. В противном случае метод вернет копию исходных двоичных данных `bytes`.

[Метод removeprefix\(\) типов bytes и bytearray в Python](#)

Если исходная байтовая строка начинаются с байтовой строки `prefix`, то метод `bytes.removeprefix()` возвращает байтовую строку без префикса `bytes[len(prefix):]`. В противном случае метод вернет копию исходных двоичных данных `bytes`.

[Метод replace\(\) типов bytes и bytearray в Python](#)

Метод `replace()` вернет копию байтовой последовательности, в которой все вхождения подпоследовательности `old` заменены на новые `new`. Если указан необязательный аргумент `count`, заменяются только первые `count` вхождения.

[Метод rfind\(\) типов bytes и bytearray в Python](#)

Метод `rfind()` возвращает самый высокий индекс в байтовой строке `bytes` или `bytearray`, в которой находится подпоследовательность байтов `sub`, такой, что `sub` содержится в пределах `s[start: end]`.

[Метод rindex\(\) типов bytes и bytearray в Python](#)

Метод `rindex()` работает и возвращает то же самое что и метод `rfind()`, за исключением того, что при неудачном поиске поднимает исключение `ValueError`.

[Метод rpartition\(\) типов bytes и bytearray в Python](#)

Метод `rpartition()` делит последовательность **при последнем вхождении `sep`** и вернет кортеж из трех значений, которые содержат часть перед разделителем, сам разделитель или его копию в виде `bytearray` и часть после разделителя.

[Метод startswith\(\) типов bytes и bytearray в Python](#)

Метод `startswith()` возвращает `True`, если двоичные данные начинаются указанным префиксом `prefix`, в противном случае возвращает `False`.

[Метод center\(\) типов bytes и bytearray в Python](#)

Метод `center()` возвращает копию строки соответствующего типа, размещенную по центру указанной длины `width`. Заполнение выполняется с использованием указанного байта заполнения `fillbyte`, по умолчанию используется пространство ASCII.

[Метод ljust\(\) типов bytes и bytearray в Python](#)

Метод `ljust()` возвращает копию строки соответствующего типа, выравненную по левому краю указанной длины `width`. Заполнение выполняется с использованием указанного байта заполнения `fillbyte`, по умолчанию используется пространство ASCII.

[Метод rjust\(\) типов bytes и bytearray в Python](#)

Метод `rjust()` возвращает копию строки соответствующего типа, выравненную по правому краю указанной длины `width`. Заполнение выполняется с использованием указанного байта заполнения `fillbyte`, по умолчанию используется пространство ASCII.

[Метод lstrip\(\) типов bytes и bytearray в Python](#)

Метод `lstrip()` возвращает копию байтовой строки с удаленными заданными ведущими байтами. Аргумент `chars` представляет собой двоичную последовательность, задающую набор байтовых значений, которые должны быть удалены.

[Метод rstrip\(\) типов bytes и bytearray в Python](#)

Метод `rstrip()` возвращает копию байтовой строки соответствующего типа с удаленными заданными `chars` конечными байтами. Аргумент `chars` представляет собой двоичную последовательность, задающую набор байтовых значений, которые должны быть удалены.

[Метод strip\(\) типов bytes и bytearray в Python](#)

Метод `strip()` возвращает копию байтовой строки соответствующего типа с удаленными начальными и конечными байтами. Аргумент `chars` представляет собой двоичную последовательность, задающую набор байтовых значений, которые должны быть удалены.

[Метод split\(\) типов bytes и bytearray в Python](#)

Метод `split()` делит байтовую строку на список строк того же типа, используя `sep` в качестве разделителя байтовой строки. Аргументом `sep` может быть любой байт-подобный объект.

[Метод rsplit\(\) типов bytes и bytearray в Python](#)

Метод `rsplit()` делит байтовую строку на список подстрок того же типа, используя `sep` в качестве байтового разделителя. За исключением того, что начинает деление байтовой строки справа, метод `rsplit()` ведет себя как метод `split()`

[Метод capitalize\(\) типов bytes и bytearray в Python](#)

Метод `capitalize()` возвращает копию байтовой строки, каждый байт которой интерпретируется как символ ASCII, причем первый байт будет заглавный, а остальные строчные. Байтовые значения, отличные от ASCII, передаются без изменений.

[Метод expandtabs\(\) типов bytes и bytearray в Python](#)

Метод `expandtabs()` возвращает копию байтовой строки, в которой все символы табуляции ASCII заменяются одним или несколькими пробелами ASCII, в зависимости от текущего столбца и заданного размера вкладки.

[Метод isalnum\(\) типов bytes и bytearray в Python](#)

Метод `isalnum()` вернет `True` если все байты в байтовой строке являются алфавитными символами ASCII или десятичными цифрами ASCII и байтовая строка не пуста, противном случае `False`.

[Метод isalpha\(\) типов bytes и bytearray в Python](#)

Метод `isalpha()` вернет `True` если все байты в последовательности являются буквенными символами ASCII и байтовая строка не пуста, противном случае `False`.

[Метод isdigit\(\) типов bytes и bytearray в Python](#)

Метод `isdigit()` вернет `True` если все байты в байтовой строке являются только десятичными цифрами ASCII и байтовая строка не пуста, в противном случае вернет `False`.

[Метод isascii\(\) типов bytes и bytearray в Python](#)

Метод `isascii()` вернет `True`, если байтовая строка пуста или все байты в последовательности ASCII, в противном случае вернет `False`.

[Метод islower\(\) типов bytes и bytearray в Python](#)

Метод `islower()` возвращает `True`, если в байтовой строке есть хотя бы один строчный символ ASCII и нет прописных символов ASCII, в противном случае вернет `False`.

[Метод isspace\(\) типов bytes и bytearray в Python](#)

Метод `isspace()` возвращает `True`, если все байты в байтовой строке являются пробельными символами ASCII и последовательность не пуста, в противном случае вернет `False`.

[Метод istitle\(\) типов bytes и bytearray в Python](#)

Метод `istitle()` вернет `True`, если последовательности символов ASCII в байтовой строке начинаются с символов в верхнем регистре и байтовая строка не пуста, в противном случае метод вернет `False`.

[Метод isupper\(\) типов bytes и bytearray в Python](#)

Метод `isupper()` вернет `True` если в байтовой строке есть хотя бы один алфавитный символ в верхнем регистре ASCII и нет символов ASCII в нижнем регистре, в противном случае вернет `False`.

[Метод lower\(\) типов bytes и bytearray в Python](#)

Метод `lower()` вернет копию байтовой строки, в которой все символы ASCII в верхнем регистре преобразованы в соответствующие им строчные буквы (нижний регистр).

[Метод splitlines\(\) типов bytes и bytearray в Python](#)

Метод `splitlines()` возвращает список байтовых строк строк из двоичной последовательности. Деление происходит по управляющим символам перевода строки ASCII, при этом учитываются все возможные символы разрыва строки.

[Метод swapcase\(\) типов bytes и bytearray в Python](#)

Метод `swapcase()` вернет копию байтовой строки со всеми символами ASCII в нижнем регистре, преобразованными в соответствующие им символы в верхний регистр, и наоборот.

[Метод title\(\) типов bytes и bytearray в Python](#)

Метод `title()` возвращает копию байтовой строки, в которой все слова начинаются с символа ASCII в верхнем регистре, а остальные символы с нижнего регистра.

[Метод upper\(\) типов bytes и bytearray в Python](#)

Метод `upper()` вернет копию байтовой строки, в которой все символы ASCII в нижнем регистре преобразованы в соответствующие им прописные буквы (верхний регистр).

[Метод zfill\(\) типов bytes и bytearray в Python](#)

Метод `zfill()` вернет копию байтовой строки и заполнит цифрами ASCII `'b'0''` пространство `'width - len(s)'`, для создания последовательности символов заданной ширины `'width'`.

[Функция printf байтовых строк в Python](#)

Байтовые объекты `'bytes/bytearray'` имеют одну уникальную встроенную операцию: оператор `'%'`, оператор называется по модулю. Это оператор также известен как оператор форматирования байтов или интерполяции.

ХОЧУ ПОМОЧЬ
ПРОЕКТУ

Ещё!

Распродажа

до 80%

Успейте с 18 по 30 сентября





[DOCS-Python.ru™](#), 2023 г. (Внимание! При копировании материала ссылка на источник обязательна) [@docs_python_ru](#)