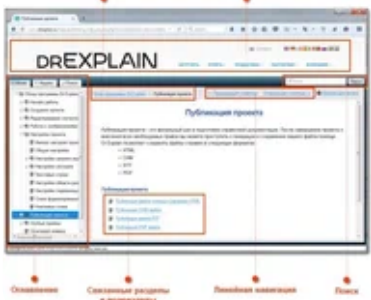


# Операции с текстовыми строками str в Python



drexpain.ru

**Делать документацию в Dr.Explain быстрее, чем в MS Word**  
Видео-обзор всех возможностей за 15 минут + ответим на все вопросы 24/7  
Смотреть

Узнать больше

РЕКЛАМА

[Справочник по языку Python3.](#) / Операции с текстовыми строками str в Python

## Методы и функции строк str

Строка – это последовательность символов, то есть некоторые наборы слов. Слова могут быть как на английском языке, так и на любом другом, поддерживаемом стандартом Unicode, что означает почти на любом языке мира. По умолчанию все строки в Python-3 в Unicode.

[Строки - тип str](#) используются почти в каждой программе на Python. Поэтому уделите чуть больше внимание представленным здесь [методам текстовых строк](#).

Напоминаем: Для строк еще доступны [все общие операции с последовательностями](#);

### [Метод str.capitalize\(\) в Python, первая буква в строке заглавная](#)

Метод str.capitalize() позволяет вернуть копию строки str с первым символом в верхнем регистре, а остальные символы будут в нижнем регистре.

### [Метод str.casefold\(\) в Python, сворачивает регистр строки](#)

Метод str.casefold() вернет регистр свернутой копии строки str. Вернет строку, приведенную к нижнему регистру символов в результате свертывания регистра. Строки в свернутом регистре могут быть использованы для сопоставления строк без регистра.

### [Метод str.center\(\) в Python, выравнивает строку по центру](#)

Метод str.center() позиционирует по центру строку str, дополняя её справа и слева до требуемой длины width указанным символом fillchar. По умолчанию fillchar использует пробел ASCII

### [Метод str.count\(\) в Python, считает совпадения в строке](#)

Метод str.count() возвращает количество вхождений подстроки sub в строку str в диапазоне индексов [start, end]. Необязательные аргументы start и end интерпретируются как обозначения среза строки.

### [Метод str.encode\(\) в Python, преобразует строку в байты](#)

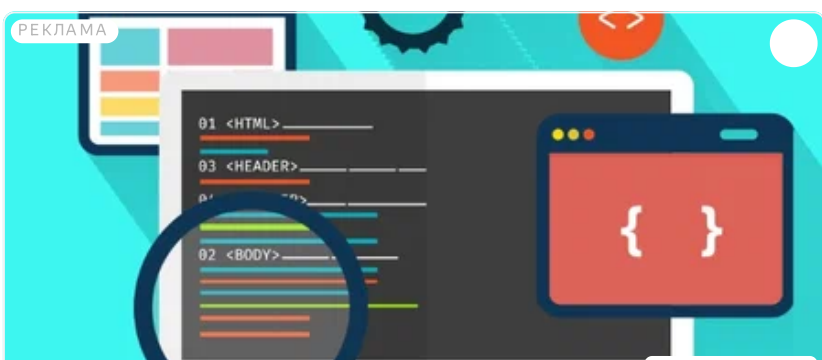
Метод str.encode() вернет закодированную версию строки str как объект байтов. Другими словами кодирует текстовую строку str в строку байтов, используя указанную кодировку encoding.

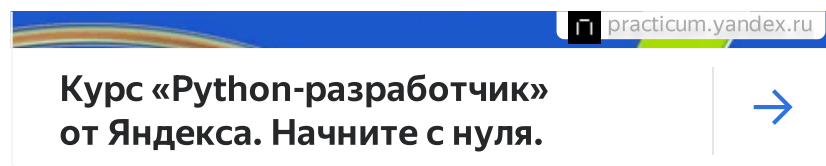
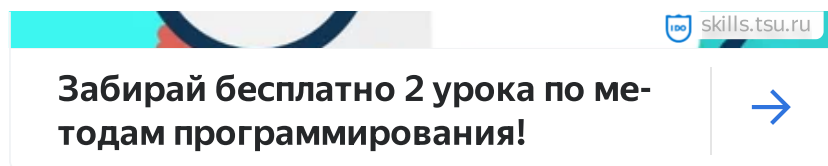
### [Метод str.endswith\(\) в Python, совпадение с концом строки](#)

Метод str.endswith() возвращает True, если строка str заканчивается указанным суффиксом suffix, в противном случае возвращает False. Суффикс suffix также может быть кортежем суффиксов для поиска.

### [Метод str.expandtabs\(\) в Python, меняет табуляцию на пробел](#)

Метод str.expandtabs() возвращает копию строки str, в которой все символы табуляции \t заменяются одним или несколькими пробелами, в зависимости от текущего столбца и заданного размера табуляции tabsize.





## [Метод `str.find\(\)` в Python, индекс первого совпадения в строке](#)

Метод `str.find()` возвращает наименьший индекс в строке `str`, где подстрока или символ `sub` находится в пределах среза `str[start:end]`. Другими словами, находит и возвращает индекс начала подстроки в строке `str`.

## [Метод `str.format\(\)` в Python, форматирует строку](#)

Метод `str.format()` выполняет операцию форматирования строки `str`. Строка `str` может иметь замещающие поля `{}`. Каждое такое поле содержит либо числовой индекс позиционного аргумента, либо имя ключевого аргумента.

## [Метод `str.format\_map\(\)` в Python](#)

Метод `str.format_map()` работает аналогично `str.format(**mapping)`, за исключением того, что `mapping` используется напрямую и не копируется в словарь `dict`. Это полезно, если, например `mapping`, подкласс `dict`.

## [Метод `str.index\(\)` в Python, индекс первого совпадения подстроки](#)

Метод `str.index()` возвращает индекс первого совпадения подстроки `sub` в строке `str`, где подстрока или символ `sub` находится в пределах среза `str[start:end]`. Метод бросает исключение `ValueError`, если символ или подстрока `sub` не найдены.

## [Метод `str.isalnum\(\)` в Python, строка состоит из цифр и букв](#)

Метод `str.isalnum()` возвращает `True`, если все символы в строке являются буквенно-цифровыми и есть хотя бы один символ, в противном случае `False`.

## [Метод `str.isalpha\(\)` в Python, строка состоит только из букв](#)

Метод строки `str.isalpha()` возвращает `True`, если все символы в строке `str` являются буквенными и есть хотя бы один символ, в противном случае `False`.

## [Метод `str.isascii\(\)` в Python, все символы в строке являются ASCII](#)

Метод `str.isascii()` возвращает `True`, если строка пуста или все символы в строке являются ASCII, `False` в противном случае.

## [Метод `str.isdecimal\(\)` в Python, проверяет строку на десятичное число](#)

Метод `str.isdecimal()` возвращает `True`, если все символы в строке являются десятичными и есть хотя бы один символ, в противном случае `False`.

## [Метод `str.isdigit\(\)` в Python, строка состоит только из цифр](#)

Метод `str.isdigit()` возвращает `True`, если все символы в строке являются цифрами и есть хотя бы один символ, в противном случае `False`.

## [Метод `str.isidentifier\(\)` проверяет строку на идентификатор Python](#)

Метод `str.isidentifier()` возвращает `True`, если строка `str` является допустимым идентификатором в соответствии с определением языка, в противном случае `False`.

## [Метод `str.islower\(\)` в Python, проверяет строку на нижний регистр](#)

Метод `str.islower()` возвращает `True`, если все символы в строке имеют нижний регистр и есть хотя бы один символ, в противном случае `False`.

## [Метод `str.isnumeric\(\)` в Python, проверяет строку на числовые символы](#)

Метод `str.isnumeric()` возвращает `True`, если все символы в строке являются числовыми символами, и есть по крайней мере один числовой символ, `False` в противном случае

## [Метод `str.isprintable\(\)` в Python, проверяет на доступность для печати](#)

Метод `str.isprintable()` возвращает `True`, если все символы в строке доступны для печати или строка пуста, в противном случае `False`.

## [Метод `str.isspace\(\)` в Python, является ли строка пробелом](#)

Метод `str.isspace()` возвращает `True`, если в строке есть только пробелы и если есть хотя бы один пробел, в противном случае `False`.

### [Метод str.istitle\(\) в Python, проверяет наличие заглавных букв в словах](#)

Метод `str.istitle()` возвращает `True`, если каждое слово в строке `str` начинается с заглавной буквы и в ней есть хотя бы один символ в верхнем регистре. Возвращает `False` в противном случае.

### [Метод str.isupper\(\) в Python, проверяет строку на верхний регистр](#)

Метод `str.isupper()` возвращает `True`, если все символы в строке `str` прописные и есть хотя бы один символ в верхнем регистре, в противном случае `False`.

### [Метод str.join\(\) в Python, объединяет список строк](#)

Метод `str.join()` возвращает строку, которая является конкатенацией элементов объекта со строками `iterable`. Разделителем между элементами является строка `str`.

### [Метод str.ljust\(\) в Python, ровняет строку по левому краю](#)

Метод `str.ljust()` вернет новую строку с текстом `str`, выровненным по левому краю и шириной `width`. Заполнение строки `str` выполняется с использованием указанного символа `fillchar`. По умолчанию используется пробел ASCII.

### [Метод str.lower\(\) в Python, строку в нижний регистр](#)

Метод `str.lower()` вернет копию строки `str`, в которой все символы будут преобразованы в нижний регистр.

### [Метод str.lstrip\(\) в Python, обрезает символы в начале строки](#)

Метод `str.lstrip()` вернет копию строки `str` с удаленными символами начала строки `chars`. Аргумент `chars` - это строка, указывающая набор удаляемых символов.

### [Метод str.maketrans\(\) в Python, таблица символов для str.translate\(\)](#)

Статический метод `str.maketrans()` таблицу преобразования символов, используемую для метода `str.translate()`.

### [Метод str.partition\(\) в Python, делит строку по первому совпадению](#)

Метод `str.partition()` разбивает строку при первом появлении разделителя `sep` и вернет кортеж, содержащий часть строки `str` перед разделителем, сам разделитель `sep` и часть строки `str` после разделителя.

### [Метод str.removeprefix\(\) в Python, удаляет префикс строки](#)

Если строка `str` начинается со строки префикса `prefix`, то метод `str.removeprefix()` возвращает копию строки без префикса `string[len(prefix):]`.

### [Метод str.removesuffix\(\) в Python, удаляет суффикс строки](#)

Если исходная строка `str` заканчивается строкой суффикса `suffix`, то метод `str.removesuffix()` возвращает копию строки без суффикса `string[:-len(suffix)]`.

### [Метод str.replace\(\) в Python, меняет подстроку/символ в строке](#)

Метод `str.replace()` вернет копию строки, в которой все вхождения подстроки `old` заменены на подстроку `new`.

### [Метод str.rfind\(\) в Python, индекс последнего совпадения подстроки](#)

Метод `str.rfind()` возвращает индекс последнего совпадения подстроки `sub` в строке `str`, где подстрока или символ `sub` находится в пределах среза `str[start:end]`.

### [Метод str.rindex\(\) в Python, индекс последнего совпадения в строке](#)

Метод `str.rindex()` возвращает индекс последнего совпадения подстроки `sub` в строке `str`, где подстрока или символ `sub` находится в пределах среза `str[start:end]`. Метод бросает исключение `ValueError`, если символ или подстрока `sub` не найдены

### [Метод str.rjust\(\) в Python, ровняет строку по правому краю](#)

Метод `str.rjust()` вернет новую строку с текстом `str`, выровненным по правому краю и шириной `width`.

### [Метод str.rpartition\(\) в Python, делит строку по последнему совпадению](#)

Метод `str.rpartition()` разбивает строку при последнем появлении разделителя `sep` и вернет кортеж, содержащий часть строки `str` перед разделителем, сам разделитель `sep` и часть строки `str` после разделителя.

### [Метод str.rsplit\(\) в Python, делит строку справа](#)

Метод `str.rsplit()` возвращает список подстрок (слов) из строки `str`, используя разделитель `sep` в качестве разделителя строки `str`. Метод `str.rsplit()` деление строки начинает с права.

### [Метод str.rstrip\(\) в Python, обрезает символы на конце строки](#)

Метод `str.rstrip()` вернет копию строки `str` с удаленными символами конца строки `chars`. Другими словами, обрежет сзади строку `str` на заданные символы `chars`. Аргумент `chars` - это строка, указывающая набор удаляемых символов.

### [Метод str.split\(\) в Python, делит строку по подстроке](#)

Метод `str.split()` возвращает список слов в строке, используя `sep` в качестве разделителя строки. Если задан `maxsplit`, то выполняется не более `maxsplit` разбиений, таким образом, список будет иметь не более `maxsplit+1` элементов.

### [Метод str.splitlines\(\) в Python, делит текст по символу '\n'](#)

Метод `str.splitlines()` возвращает список строк, текста `str`, разделенного по универсальным разрывам строк. Разрывы (разделители) строк не включаются в результирующий список, если не задано значение `keepends=True`.

### [Метод str.startswith\(\) в Python, совпадение с началом строки](#)

Метод `str.startswith()` возвращает `True`, если строка `str` начинается указанным префиксом `prefix`, в противном случае возвращает `False`. Ограничивать поиск начала строки можно необязательными индексами `start` и `end`.

### [Метод str.strip\(\) в Python, обрежет строку с обоих концов](#)

Метод `str.strip()` вернет копию строки `str` с удаленными начальными и конечными символами `chars`. Другими словами, обрежет строку `str` с обоих концов. Аргумент `chars` - это строка, указывающая набор удаляемых символов.

### [Метод str.swapcase\(\) в Python, сменит регистр символов в строке](#)

Метод `str.swapcase()` возвращает копию строки с прописными символами, преобразованными в строчные и наоборот. Другими словами метод меняет регистр символов в строке `str`.

### [Метод str.title\(\) в Python, каждое слово с заглавной буквы](#)

Метод `str.title()` возвращает копию строки, в которой у каждого слова, первый символ имеет верхний регистр, а остальные символы слова переводятся в нижний регистр.

### [Метод str.translate\(\) в Python, транслирование строки](#)

Метод `str.translate()` возвращает копию строки, в которой каждый символ был сопоставлен и преобразован согласно карте перевода символов `table`.

### [Метод str.upper\(\) в Python, переведет строку в верхний регистр](#)

Метод `str.upper()` вернет копию строки `str` с символами, преобразованными в верхний регистр. Используемый алгоритм преобразования в верхний регистр описан в разделе 3.13 стандарта Unicode.

### [Метод str.zfill\(\) в Python, дополнит строку нулями](#)

Метод `str.zfill()` вернет копию строки, у которой начало строки будет заполнено цифрой ASCII 0, до указанной длины `width`. Начальный префикс знака '+' / '-' обрабатывается путем вставки отступа после символа знака, а не до него.

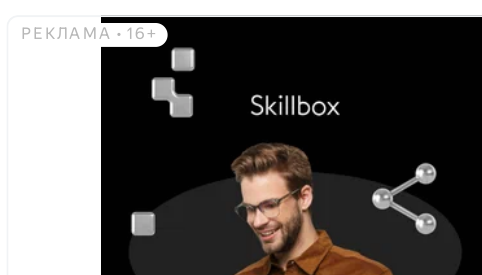
### [Форматирование строк в стиле printf в Python](#)

Строки имеют одну уникальную встроенную операцию: оператор `%`. При заданном формате `'string' % values`, спецификации преобразований в `string` заменяются на ноль или более элементов значений. Эффект аналогичен использованию `sprintf()` в языке C.


### [F-строки. Форматированные строки в Python](#)

Форматированный строковый литерал или `f-string` - это строковый литерал с префиксом `'f'` или `'F'`. Эти строки могут содержать поля замены, которые являются выражениями, разделенными фигурными скобками `{}`.

ХОЧУ ПОМОЧЬ  
ПРОЕКТУ







S skillbox.ru

**Курс «Data Scientist» - 3 специализации на выбор!**

Научитесь работать с Big Data, писать эффективный код на Python и SQL-запросы.

Освойте английский с нуля

Трудоустройство

Содержание курса

Формат обучения

Узнать больше

[DOCS-Python.ru](#)<sup>™</sup>, 2023 г.

(**Внимание!** При копировании материала ссылка на источник обязательна)

[@docs\\_python\\_ru](#)