**Таблица "Функции и методы строк"**

| **Функция или метод** | **Назначение** |
| --- | --- |
| **S = 'str'; S = "str"; S = '''str'''; S = """str"""** | [Литералы строк](https://pythonworld.ru/tipy-dannyx-v-python/stroki-literaly-strok.html) |
| **S = "s\np\ta\nbbb"** | Экранированные последовательности |
| **S = r"C:\temp\new"** | Неформатированные строки (подавляют экранирование) |
| **S = b"byte"** | Строка [байтов](https://pythonworld.ru/tipy-dannyx-v-python/bajty-bytes-i-bytearray.html) |
| **S1 + S2** | Конкатенация (сложение строк) |
| **S1 \* 3** | Повторение строки |
| **S[i]** | Обращение по индексу |
| **S[i:j:step]** | Извлечение среза |
| **len**(S) | Длина строки |
| **S.find**(str, [start],[end]) | Поиск подстроки в строке. Возвращает номер первого вхождения или -1  Метод find(<sub>, <start>, <end>) находит **индекс первого вхождения** подстроки <sub> в исходной строке s. Если строка s не содержит подстроки <sub>, то метод возвращает значение -1. Мы можем использовать данный метод наравне с оператором in для проверки: содержит ли заданная строка некоторую подстроку или нет.  Результатом выполнения следующего кода:  s = 'foo bar foo baz foo qux'  print(s.find('foo'))  print(s.find('bar'))  print(s.find('qu'))  print(s.find('python'))  будет:  0  4  20  -1 |
| **S.rfind**(str, [start],[end]) | Поиск подстроки в строке. Возвращает номер последнего вхождения или -1  Метод rfind(<sub>, <start>, <end>) идентичен методу find(<sub>, <start>, <end>), за тем исключением, что он ищет первое вхождение подстроки <sub> начиная с конца строки s |
| **S.index**(str, [start],[end]) | Поиск подстроки в строке. Возвращает номер первого вхождения или вызывает ValueError  Метод index(<sub>, <start>, <end>) идентичен методу find(<sub>, <start>, <end>), за тем исключением, что он **вызывает ошибку**  ValueError: substring not found во время выполнения программы, если подстрока <sub> не найдена. |
| **S.rindex**(str, [start],[end]) | Поиск подстроки в строке. Возвращает номер последнего вхождения или вызывает ValueError  Метод rindex(<sub>, <start>, <end>) идентичен методу index(<sub>, <start>, <end>), за тем исключением, что он ищет первое вхождение подстроки <sub> начиная с конца строки s. |
| **S.replace**(шаблон, замена[, maxcount]) | Замена шаблона на замену. maxcount ограничивает количество замен  Метод replace(<old>, <new>) возвращает копию s **со всеми** вхождениями подстроки <old>, замененными на <new>.  Результатом выполнения следующего кода:  s = 'foo bar foo baz foo qux'  print(s.replace('foo', 'grault'))  будет:  grault bar grault baz grault qux  Метод replace() может принимать опциональный третий аргумент <count>,  который определяет количество замен.  Результатом выполнения следующего кода:  s = 'foo bar foo baz foo qux'  print(s.replace('foo', 'grault', 2))  будет:  grault bar grault baz foo qux |
| **S.split**(символ) | Разбиение строки по разделителю |
| **S.isdigit**() | Состоит ли строка из цифр  Метод isdigit() определяет, состоит ли исходная строка **только** из цифровых символов. Метод возвращает значение True если исходная строка является непустой и состоит **только** из цифровых символов и False в противном случае.  Результатом выполнения следующего кода:  s1 = '1234567'  s2 = 'abc123'  s3 = ''  print(s1.isdigit())  print(s2.isdigit())  print(s3.isdigit())  будет:  True  False  False |
| **S.isalpha**() | Состоит ли строка из букв  Метод isalpha() определяет, состоит ли исходная строка из буквенных символов. Метод возвращает значение True если исходная строка является непустой и состоит **только** из буквенных символов и False в противном случае.  Результатом выполнения следующего кода:  s1 = 'ABCabc'  s2 = 'abc123'  s3 = ''  print(s1.isalpha())  print(s2.isalpha())  print(s3.isalpha())  будет:  True  False  False |
| **S.isalnum**() | Состоит ли строка из цифр или букв  Метод isalnum() определяет, состоит ли исходная строка из буквенно-цифровых символов. Метод возвращает значение True если исходная строка является непустой и состоит **только** из буквенно-цифровых символов и False в противном случае.  Результатом выполнения следующего кода:  s1 = 'abc123'  s2 = 'abc$\*123'  s3 = ''  print(s1.isalnum())  print(s2.isalnum())  print(s3.isalnum())  будет:  True  False  False |
| **S.islower**() | Состоит ли строка из символов в нижнем регистре  Метод islower() определяет, являются ли **все** буквенные символы исходной строки строчными (имеют нижний регистр). Метод возвращает значение True если все буквенные символы исходной строки являются строчными и False в противном случае. **Все неалфавитные символы игнорируются!**  Результатом выполнения следующего кода:  s1 = 'abc'  s2 = 'abc1$d'  s3 = 'Abc1$D'  print(s1.islower())  print(s2.islower())  print(s3.islower())  будет:  True  True  False |
| **S.isupper**() | Состоит ли строка из символов в верхнем регистре  Метод isupper() определяет, являются ли **все** буквенные символы исходной строки заглавными (имеют верхний регистр). Метод возвращает значение True если все буквенные символы исходной строки являются заглавными и False в противном случае. **Все неалфавитные символы игнорируются!**  Результатом выполнения следующего кода:  s1 = 'ABC'  s2 = 'ABC1$D'  s3 = 'Abc1$D'  print(s1.isupper())  print(s2.isupper())  print(s3.isupper())  будет:  True  True  False |
| **S.isspace**() | Состоит ли строка из неотображаемых символов (пробел, символ перевода страницы ('\f'), "новая строка" ('\n'), "перевод каретки" ('\r'), "горизонтальная табуляция" ('\t') и "вертикальная табуляция" ('\v'))  Метод isspace() определяет, состоит ли исходная строка **только** из пробельных символов. Метод возвращает значение True если строка состоит только из пробельных символов и False в противном случае.  Результатом выполнения следующего кода:  s1 = ' '  s2 = 'abc1$d'  print(s1.isspace())  print(s2.isspace())  будет:  True  False |
| **S.istitle**() | Начинаются ли слова в строке с заглавной буквы |
| **S.upper**() | Преобразование строки к верхнему регистру  Метод upper() возвращает копию строки s, в которой все символы имеют верхний регистр.  Результатом выполнения следующего кода:  s = 'FOO Bar 123 baz qUX'  print(s.upper())  будет:  FOO BAR 123 BAZ QUX |
| **S.lower**() | Преобразование строки к нижнему регистру  Метод lower() возвращает копию строки s, в которой все символы имеют нижний регистр.  Результатом выполнения следующего кода:  s = 'FOO Bar 123 baz qUX'  print(s.lower())  будет:  foo bar 123 baz qux |
| **S.startswith**(str) | Начинается ли строка S с шаблона str  Метод startswith(<suffix>, <start>, <end>) определяет **начинается**ли исходная строка s подстрокой <suffix>. Если исходная строка начинается с подстроки <suffix>,метод возвращает значение True, а если нет, то  False.  Результатом выполнения следующего кода:  s = 'foobar'  print(s.startswith('foo'))  print(s.startswith('baz'))  будет:  True  False |
| **S.endswith**(str) | Заканчивается ли строка S шаблоном str  Метод endswith(<suffix>, <start>, <end>) определяет **оканчивается**ли исходная строка s подстрокой <suffix>. Метод возвращает значение True если исходная строка оканчивается на подстроку <suffix> и False в противном случае.  Результатом выполнения следующего кода:  s = 'foobar'  print(s.endswith('bar'))  print(s.endswith('baz'))  будет:  True  False |
| **S.join**(список) | Сборка строки из списка с разделителем S |
| **ord**(символ) | Символ в его код ASCII |
| **chr**(число) | Код ASCII в символ |
| **S.capitalize**() | Метод capitalize() возвращает копию строки s, в которой первый символ имеет верхний регистр, а все остальные символы имеют нижний регистр.  Результатом выполнения следующего кода:  s = 'foO BaR BAZ quX'  print(s.capitalize())  будет:  Foo bar baz qux  Символы, не являющиеся буквами алфавита, остаются неизменными. |
| **S.center**(width, [fill]) | Возвращает отцентрованную строку, по краям которой стоит символ fill (пробел по умолчанию) |
| **S.count**(str, [start],[end]) | Возвращает количество непересекающихся вхождений подстроки в диапазоне [начало, конец] (0 и длина строки по умолчанию)  Метод count(<sub>, <start>, <end>) считает количество **непересекающихся** вхождений подстроки <sub> в исходную строку s.  Результатом выполнения следующего кода:  s = 'foo goo moo'  print(s.count('oo'))  print(s.count('oo', 0, 8)) # подсчет с 0 по 7 символ  будет:  3  2 |
| **S.expandtabs**([tabsize]) | Возвращает копию строки, в которой все символы табуляции заменяются одним или несколькими пробелами, в зависимости от текущего столбца. Если TabSize не указан, размер табуляции полагается равным 8 пробелам |
| **S.lstrip**([chars]) | Удаление пробельных символов в начале строки  Метод lstrip() возвращает копию строки s у которой удалены все пробелы стоящие **в начале** строки.  Результатом выполнения следующего кода:  s = ' foo bar foo baz foo qux '  print(s.lstrip())  будет:  foo bar foo baz foo qux⎵ ⎵ ⎵ ⎵ ⎵ ⎵ |
| **S.rstrip**([chars]) | Удаление пробельных символов в конце строки  Метод rstrip() возвращает копию строки s у которой удалены все пробелы стоящие **в конце** строки.  Результатом выполнения следующего кода:  s = ' foo bar foo baz foo qux '  print(s.rstrip())  будет:  ⎵ ⎵ ⎵ ⎵ ⎵ ⎵foo bar foo baz foo qux |
| **S.strip**([chars]) | Удаление пробельных символов в начале и в конце строки  Метод strip() возвращает копию строки s у которой удалены все пробелы стоящие **в начале и конце** строки.  Результатом выполнения следующего кода:  s = ' foo bar foo baz foo qux '  print(s.strip())  будет:  foo bar foo baz foo qux  Методы strip(), lstrip(), rstrip() могут принимать на вход опциональный аргумент<chars>. Необязательный аргумент <chars>– это строка, которая определяет набор символов для удаления. |
| **S.partition**(шаблон) | Возвращает кортеж, содержащий часть перед первым шаблоном, сам шаблон, и часть после шаблона. Если шаблон не найден, возвращается кортеж, содержащий саму строку, а затем две пустых строки |
| **S.rpartition**(sep) | Возвращает кортеж, содержащий часть перед последним шаблоном, сам шаблон, и часть после шаблона. Если шаблон не найден, возвращается кортеж, содержащий две пустых строки, а затем саму строку |
| **S.swapcase**() | Переводит символы нижнего регистра в верхний, а верхнего – в нижний  Метод swapcase() возвращает копию строки s, в которой все символы, имеющие верхний регистр, преобразуются в символы нижнего регистра и наоборот.   Результатом выполнения следующего кода:  s = 'FOO Bar 123 baz qUX'  print(s.swapcase())  будет:  foo bAR 123 BAZ Qux |
| **S.title**() | Первую букву каждого слова переводит в верхний регистр, а все остальные в нижний  Метод title() возвращает копию строки s, в которой первый символ каждого слова переводится в верхний регистр.  Результатом выполнения следующего кода:  s = 'the sun also rises'  print(s.title())  будет:  The Sun Also Rises  Этот метод использует довольно простой алгоритм: он не пытается различить важные и неважные слова и не обрабатывает аббревиатуры и апострофы. Результатом выполнения следующего кода:  s = "what's happened to ted's IBM stock?"  print(s.title())  будет:  What'S Happened To Ted'S Ibm Stock? |
| **S.zfill**(width) | Делает длину строки не меньшей width, по необходимости заполняя первые символы нулями |
| **S.ljust**(width, fillchar=" ") | Делает длину строки не меньшей width, по необходимости заполняя последние символы символом fillchar |
| **S.rjust**(width, fillchar=" ") | Делает длину строки не меньшей width, по необходимости заполняя первые символы символом fillchar |
| **S.format**(\*args, \*\*kwargs) | [Форматирование строки](https://pythonworld.ru/osnovy/formatirovanie-strok-metod-format.html) |