



Строки

Докладчик: Евграфов Михаил

Строки

str

конвертация логического типа

```
>>> str(True)
'True'
```

конвертация целых чисел

```
>>> str(42)
'42'
```

конвертация float

```
>>> str(3.14)
'3.14'
```

конвертация списка

```
>>> str([1, 2, 3])
'[1, 2, 3]'
```

конвертация кортежа

```
>>> str((4, 5, 6))
'(4, 5, 6)'
```

конвертация строки

```
>>> str("abc")
'abc'
```

пустая строка

```
>>> str()
''
```

Строковые литералы

одинарные кавычки

'this is string literal'

двойные кавычки

"this is string literal too"

включение кавычек другого типа

"I'm iron man"

включение кавычек того же типа

'I\'m iron man'

Объединение литералов

```
# объединение с помощью \  
>>> string = "this is one " \  
             "single string"  
>>> string  
'this is one single string'
```

```
# python-way объединение  
>>> string = (  
             "this is one "  
             "single string"  
             )  
>>> string  
'this is one single string'
```

```
# объединение на одной строке  
>>> "abc" "def" "ghi" "j"  
'abcdefghij'
```

Специальный символ \n

```
>>> string = (  
    "string 1\n"  
    "string 2\n"  
    "string 3"  
)
```

```
>>> print(string)
```

```
string 1
```

```
string 2
```

```
string 3
```

Специальные символы \t и \r

использование \t

```
>>> string = "word\ttabbed word"
```

```
>>> print(string)
```

```
word      tabbed word
```

использование \r

```
>>> string = "world\roverriden world"
```

```
>>> print(string)
```

```
overriden world
```

Мультистроки

двойные кавычки

```
>>> print(
    """\
line1
line2\
"""
)
line1
line2
```

одинарные кавычки

```
>>> print(
    '''\
line1
line2\
'''
)
line1
line2
```


Пример использования мультистрок

```
def do_staff(param1: int, param2: str = "") -> None:  
    """
```

```
    Do some really usefull staff.
```

```
    Args:
```

```
        param1: first parameter - integer.
```

```
        param2: second parameter - string.
```

```
        Default - empty string.
```

```
    Returns:
```

```
        None.
```

```
    """
```

```
    ...
```

Определение длины

длина непустых строки

```
>>> len("abcdefg")  
7
```

пробелы тоже считаются

```
>>> len("abc def")  
7
```

длина пустой строки всегда 0

```
>>> len("")  
0
```

Конкатенация

```
>>> string1 = "abc"
```

```
>>> string2 = "def"
```

```
# конкатенация
```

```
>>> string1 + string2  
'abcdef'
```

```
# конкатенация в составном присваивании
```

```
>>> string1 += string2
```

```
>>> string1  
'abcdef'
```


Повторение

```
>>> string1 = "abc"
```

```
>>> string2 = "def"
```

```
# повторение
```

```
>>> string2 * 3  
'defdefdef'
```

```
# повторение в составном присваивании
```

```
>>> string2 *= 3
```

```
>>> string2  
'defdefdef'
```

Оператор in

```
>>> "bcd" in "abcdef"  
True
```

```
>>> "ace" in "abcdef"  
False
```

```
>>> "" in "abc"  
True
```

Индексирование

```
string = 'what a wonderful world'
```

<code>string[1]</code>	<code># == h</code>
<code>string[-1]</code>	<code># == d</code>
<code>string[::-1]</code>	<code># == dlrow lufrednow a tahw</code>
<code>string[2:8]</code>	<code># == at a w</code>

Логические операции

```
>>> "abc" == "def"
```

```
False
```

```
>>> "abc" != "def"
```

```
True
```

```
>>> "abc" < "def"
```

```
True
```

```
>>> "abc" <= "def"
```

```
True
```

```
>>> "abc" > "def"
```

```
False
```

```
>>> "abc" >= "def"
```

```
False
```

Работа с кодами

```
# использование ord()
```

```
>>> ord("a")  
97
```

```
# использование chr()
```

```
>>> chr(97)  
'a'
```

```
# неверное использование ord()
```

```
>>> ord("hello")  
TypeError: ...
```

Методы манипуляции с регистром

использование upper

```
>>> 'Gvido Van Rossum'.upper()  
'GVIDEO VAN ROSSUM'
```

использование lower

```
>>> 'Gvido Van Rossum'.lower()  
'gvido van rossum'
```

использование title

```
>>> 'gvido van rossum'.title()  
'Gvido Van Rossum'  
>>> 'GVIDEO VAN ROSSUM'.title()  
'Gvido Van Rossum'
```


Пример использования

```
words = [  
    'CIA', 'border', 'Alabama', 'apple',  
    'Appel', 'zero', 'two', 'Paris',  
]  
words.sort()  
  
print(words)  
# [  
#     'Alabama', 'Appel', 'CIA', 'Paris',  
#     'apple', 'border', 'two', 'zero'  
# ]
```

Пример использования

```
words = [  
    'CIA', 'border', 'Alabama', 'apple',  
    'Appel', 'zero', 'two', 'Paris',  
]  
words.sort(key=str.upper)  
  
print(words)  
# [  
#     'Alabama', 'Appel', 'apple', 'border',  
#     'CIA', 'Paris', 'two', 'zero'  
# ]
```

Проверка символов: `isalpha`

пример использования `isalpha`

```
>>> "AbCdEfG".isalpha()
```

```
True
```

```
>>> "12345".isalpha()
```

```
False
```

```
>>> "".isalpha()
```

```
False
```


Проверка символов: `isdigit`

пример использования `isdigit`

```
>>> "AbCdEfG".isdigit()
```

```
False
```

```
>>> "12345".isdigit()
```

```
True
```

```
>>> "".isdigit()
```

```
False
```

Проверка символов: `isspace`

пример использования `isspace`

```
>>> "AbCdEfG".isspace()
```

```
False
```

```
>>> "\t\n".isspace()
```

```
True
```

```
>>> "".isspace()
```

```
False
```

Проверка соответствия регистру

```
>>> "Abcdef".isupper()
```

```
False
```

```
>>> "ABCDEF".isupper()
```

```
True
```

```
>>> "Abcdef".islower()
```

```
False
```

```
>>> "abcdef".islower()
```

```
True
```

Методы очистки строк

```
>>> "aaaabbbaabbaaa".replace("a",  
"" )  
'bbbbbb'
```

```
>>> "aaaabbbaabbaaa".strip("a")  
'bbbaabb'
```

```
>>> "aaaabbbaabbaaa".lstrip("a")  
'bbbaabbaaa'
```

```
>>> "aaaabbbaabbaaa".rstrip("a")  
'aaaabbbaabb'
```

Методы разбиения и объединения

дефолтное разбиение

```
>>> "aba b".split()  
['aba', 'b']
```

указания разделяющей последовательности

```
>>> "aba b".split("a")  
[' ', 'b', ' b']
```

объединение

```
>>> "-".join(("one", "two"))  
'one-two'
```


Работа с подстроками

```
# startswith / endswith
```

```
>>> "hello world".startswith('hello')
```

```
True
```

```
>>> "hello world".startswith('hello', -12, -5)
```

```
True
```

```
>>> "hello world".startswith('hello', 6, 9)
```

```
False
```

```
>>> "hello world".endswith('world')
```

```
True
```

```
>>> "hello world".endswith('world', 6)
```

```
True
```

```
>>> "hello world".endswith('world', 6, 9)
```

```
False
```

Работа с подстроками

```
# startswith / endswith
```

```
>>> "banana".find("na")
```

```
2
```

```
>>> "banana".find("na", 3)
```

```
4
```

```
>>> "banana".find("na", 0, 3)
```

```
-1
```

```
>>> "banana".rfind("na")
```

```
4
```

```
>>> "banana".rfind("na", 0, 5)
```

```
2
```

```
>>> "banana".rfind("na", 0, 3)
```

```
-1
```

Работа с подстроками

```
# index / rindex
```

```
>>> "banana".index("na")
```

```
2
```

```
>>> "banana".index("na", 3)
```

```
4
```

```
>>> "banana".index("na", 0, 3)
```

```
ValueError: substring not found
```

```
>>> "banana".rindex("na")
```

```
4
```

```
>>> "banana".rindex("na", 0, 5)
```

```
2
```

```
>>> "banana".rindex("na", 0, 3)
```

```
ValueError: substring not found
```


Практическая часть