
Яндекс

Яндекс Лицей

Методы split и join
Списочные выражения

Методы split и join

Метод split

Метод split разбивает строку по произвольному разделителю на список "слов":

- Вызывается без аргументов – разбивает по символам пустого пространства
- Вызывается с аргументом-строкой – разбивает по этой строке

Метод split

Метод split разбивает строку по произвольному разделителю на список "слов":

- Вызывается без аргументов – разбивает по символам пустого пространства
- Вызывается с аргументом-строкой – разбивает по этой строке

```
s = 'раз два три'
print(s.split())
```

Метод split

Метод split разбивает строку по произвольному разделителю на список "слов":

- Вызывается без аргументов – разбивает по символам пустого пространства
- Вызывается с аргументом-строкой – разбивает по этой строке

```
s = 'раз два три'
print(s.split())                    ['раз', 'два', 'три']
```

Метод split

Метод split разбивает строку по произвольному разделителю на список "слов":

- Вызывается без аргументов – разбивает по символам пустого пространства
- Вызывается с аргументом-строкой – разбивает по этой строке

```
s = 'раз два три'
print(s.split())                    ['раз', 'два', 'три']
print(' one two three '.split())
```

Метод split

Метод split разбивает строку по произвольному разделителю на список "слов":

- Вызывается без аргументов – разбивает по символам пустого пространства
- Вызывается с аргументом-строкой – разбивает по этой строке

```
s = 'раз два три'
print(s.split())                    ['раз', 'два', 'три']
print(' one two three '.split())    ['one', 'two', 'three']
```

Метод split

Метод split разбивает строку по произвольному разделителю на список "слов":

- Вызывается без аргументов – разбивает по символам пустого пространства
- Вызывается с аргументом-строкой – разбивает по этой строке

```
s = 'раз два три'
print(s.split())                    ['раз', 'два', 'три']
print(' one two three '.split())    ['one', 'two', 'three']
print('192.168.1.1'.split('.'))
```

Метод split

Метод split разбивает строку по произвольному разделителю на список "слов":

- Вызывается без аргументов – разбивает по символам пустого пространства
- Вызывается с аргументом-строкой – разбивает по этой строке

```
s = 'раз два три'
print(s.split())                    ['раз', 'два', 'три']
print(' one two three '.split())    ['one', 'two', 'three']
print('192.168.1.1'.split('.'))     ['192', '168', '1', '1']
```

Метод split

Метод split разбивает строку по произвольному разделителю на список "слов":

- Вызывается без аргументов – разбивает по символам пустого пространства
- Вызывается с аргументом-строкой – разбивает по этой строке

```
s = 'раз два три'
print(s.split())                    ['раз', 'два', 'три']
print(' one two three '.split())    ['one', 'two', 'three']
print('192.168.1.1'.split('.'))     ['192', '168', '1', '1']
print(s.split('a'))
```

Метод split

Метод split разбивает строку по произвольному разделителю на список "слов":

- Вызывается без аргументов – разбивает по символам пустого пространства
- Вызывается с аргументом-строкой – разбивает по этой строке

```
s = 'раз два три'
print(s.split())                    ['раз', 'два', 'три']
print(' one two three '.split())    ['one', 'two', 'three']
print('192.168.1.1'.split('.'))     ['192', '168', '1', '1']
print(s.split('а'))                 ['р', 'з дв', ' три']
```

Метод split

Метод split разбивает строку по произвольному разделителю на список "слов":

- Вызывается без аргументов – разбивает по символам пустого пространства
- Вызывается с аргументом-строкой – разбивает по этой строке

```
s = 'раз два три'
print(s.split())                    ['раз', 'два', 'три']
print(' one two three '.split())    ['one', 'two', 'three']
print('192.168.1.1'.split('.'))     ['192', '168', '1', '1']
print(s.split('a'))                 ['р', 'з дв', ' три']
print('A##B##C'.split('##'))
```

Метод split

Метод split разбивает строку по произвольному разделителю на список "слов":

- Вызывается без аргументов – разбивает по символам пустого пространства
- Вызывается с аргументом-строкой – разбивает по этой строке

```
s = 'раз два три'
print(s.split())                    ['раз', 'два', 'три']
print(' one two three '.split())    ['one', 'two', 'three']
print('192.168.1.1'.split('.'))     ['192', '168', '1', '1']
print(s.split('a'))                 ['р', 'з дв', ' три']
print('A##B##C'.split('##'))        ['A', 'B', 'C']
```

Метод join

join собирает из списка слов единую строку через произвольный разделитель (точнее, соединитель):

- всегда принимает один аргумент – список "слов", которые нужно "склеить" (конкатенировать)
- соединитель – та строка, у которой вызван метод

Метод join

join собирает из списка слов единую строку через произвольный разделитель (точнее, соединитель):

- всегда принимает один аргумент – список "слов", которые нужно "склеить" (конкатенировать)
- соединитель – та строка, у которой вызван метод

```
s = ['Тот', 'Кого', 'Нельзя', 'Называть']
print(''.join(s))
```

Метод join

join собирает из списка слов единую строку через произвольный разделитель (точнее, соединитель):

- всегда принимает один аргумент – список "слов", которые нужно "склеить" (конкатенировать)
- соединитель – та строка, у которой вызван метод

```
s = ['Тот', 'Кого', 'Нельзя', 'Называть']
print(''.join(s))
```

```
ТотКогоНельзяНазывать
```

Метод join

join собирает из списка слов единую строку через произвольный разделитель (точнее, соединитель):

- всегда принимает один аргумент – список "слов", которые нужно "склеить" (конкатенировать)
- соединитель – та строка, у которой вызван метод

```
s = ['Тот', 'Кого', 'Нельзя', 'Называть']
print(''.join(s))                          ТотКогоНельзяНазывать
print(' '.join(s))
```

Метод join

join собирает из списка слов единую строку через произвольный разделитель (точнее, соединитель):

- всегда принимает один аргумент – список "слов", которые нужно "склеить" (конкатенировать)
- соединитель – та строка, у которой вызван метод

```
s = ['Тот', 'Кого', 'Нельзя', 'Называть']
print(''.join(s))
print(' '.join(s))
```

```
ТотКогоНельзяНазывать
Тот Кого Нельзя Называть
```

Метод join

join собирает из списка слов единую строку через произвольный разделитель (точнее, соединитель):

- всегда принимает один аргумент – список "слов", которые нужно "склеить" (конкатенировать)
- соединитель – та строка, у которой вызван метод

```
s = ['Тот', 'Кого', 'Нельзя', 'Называть']
print(''.join(s))
print(' '.join(s))
print('-'.join(s))
```

```
ТотКогоНельзяНазывать
Тот Кого Нельзя Называть
```

Метод join

join собирает из списка слов единую строку через произвольный разделитель (точнее, соединитель):

- всегда принимает один аргумент – список "слов", которые нужно "склеить" (конкатенировать)
- соединитель – та строка, у которой вызван метод

```
s = ['Тот', 'Кого', 'Нельзя', 'Называть']
print(''.join(s))
print(' '.join(s))
print('-'.join(s))
```

```
ТотКогоНельзяНазывать
Тот Кого Нельзя Называть
Тот-Кого-Нельзя-Называть
```

Метод join

join собирает из списка слов единую строку через произвольный разделитель (точнее, соединитель):

- всегда принимает один аргумент – список "слов", которые нужно "склеить" (конкатенировать)
- соединитель – та строка, у которой вызван метод

```
s = ['Тот', 'Кого', 'Нельзя', 'Называть']
print(''.join(s))
print(' '.join(s))
print('-'.join(s))
print('! '.join(s))
```

```
ТотКогоНельзяНазывать
Тот Кого Нельзя Называть
Тот-Кого-Нельзя-Называть
```

Метод join

join собирает из списка слов единую строку через произвольный разделитель (точнее, соединитель):

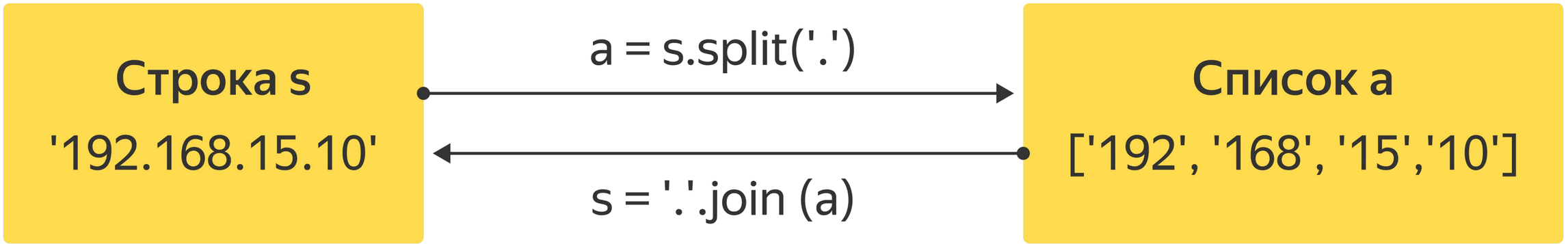
- всегда принимает один аргумент – список "слов", которые нужно "склеить" (конкатенировать)
- соединитель – та строка, у которой вызван метод

```
s = ['Тот', 'Кого', 'Нельзя', 'Называть']
print(''.join(s))
print(' '.join(s))
print('-'.join(s))
print('! '.join(s))
```

```
ТотКогоНельзяНазывать
Тот Кого Нельзя Называть
Тот-Кого-Нельзя-Называть
Тот! Кого! Нельзя! Называть
```

Методы split и join

split служит для преобразования строки в список



join служит для преобразования списка в строку

Методы split и join

split и join – методы строк.

У объектов других типов этих методов нет.

Методы split и join

split и join – методы строк.

У объектов других типов этих методов нет.

```
[1, 2, 3].join([4, 5, 6])
```

Методы split и join

split и join – методы строк.

У объектов других типов этих методов нет.

```
[1, 2, 3].join([4, 5, 6])
```

```
AttributeError: 'list' object has no attribute 'join'
```

Методы split и join

split и join – методы строк.

У объектов других типов этих методов нет.

```
[1, 2, 3].join([4, 5, 6])
{1, 2, 3}.join({4, 5, 6})
```

```
AttributeError: 'list' object has no attribute 'join'
```

Методы split и join

split и join – методы строк.

У объектов других типов этих методов нет.

```
[1, 2, 3].join([4, 5, 6])
{1, 2, 3}.join({4, 5, 6})
```

```
AttributeError: 'list' object has no attribute 'join'
AttributeError: 'set' object has no attribute 'join'
```

Списочные выражения

Списочные выражения

Они же list comprehensions.

Позволяют создавать списки в цикле for, не записывая цикл целиком

Списочные выражения

Они же list comprehensions.

Позволяют создавать списки в цикле for, не записывая цикл целиком

```
squares = []
for i in range(8):
    squares.append(i ** 2)
print(squares)
```

Списочные выражения

Они же list comprehensions.

Позволяют создавать списки в цикле for, не записывая цикл целиком

```
squares = []
for i in range(8):
    squares.append(i ** 2)
print(squares)
```

```
[0, 1, 4, 9, 16, 25, 36, 49]
```

Списочные выражения

Они же list comprehensions.

Позволяют создавать списки в цикле for, не записывая цикл целиком

```
squares = []
for i in range(8):
    squares.append(i ** 2)
print(squares)
```

```
# с помощью списочного выражения:
squares = [i ** 2 for i in range(8)]
print(squares)
```

```
[0, 1, 4, 9, 16, 25, 36, 49]
```

Списочные выражения

Они же **list comprehensions**.

Позволяют создавать списки в цикле for, не записывая цикл целиком

```
squares = []
for i in range(8):
    squares.append(i ** 2)
print(squares)
```

```
[0, 1, 4, 9, 16, 25, 36, 49]
```

```
# с помощью списочного выражения:
squares = [i ** 2 for i in range(8)]
print(squares)
```

```
[0, 1, 4, 9, 16, 25, 36, 49]
```

Списочные выражения

Добавим условие:

```
even_squares = []
for i in range(10):
    if i % 2 == 0:
        even_squares.append(i ** 2)
print(even_squares)
```

Списочные выражения

Добавим условие:

```
even_squares = []
for i in range(10):
    if i % 2 == 0:
        even_squares.append(i ** 2)
print(even_squares)
```

```
[0, 4, 16, 36, 64]
```

Списочные выражения

Добавим условие:

```
even_squares = []
for i in range(10):
    if i % 2 == 0:
        even_squares.append(i ** 2)
print(even_squares)
# с помощью списочного выражения:
even_squares2 = [i ** 2 for i in range(10) if i % 2 == 0]
print(even_squares2)
```

```
[0, 4, 16, 36, 64]
```

Списочные выражения

Добавим условие:

```
even_squares = []
for i in range(10):
    if i % 2 == 0:
        even_squares.append(i ** 2)
print(even_squares)
# с помощью списочного выражения:
even_squares2 = [i ** 2 for i in range(10) if i % 2 == 0]
print(even_squares2)
```

```
[0, 4, 16, 36, 64]
[0, 4, 16, 36, 64]
```

Списочные выражения

Несколько циклов:

```
print([i * j for i in range(3) for j in range(3)])
```

Списочные выражения

Несколько циклов:

```
print([i * j for i in range(3) for j in range(3)])
```

```
[0, 0, 0, 0, 1, 2, 0, 2, 4]
```

Списочные выражения

Считываем числа, записанные в строчку:

```
a = [int(x) for x in '976 929 289 809 7677'.split()]
print(a[3])
```

Списочные выражения

Считываем числа, записанные в строчку:

```
a = [int(x) for x in '976 929 289 809 7677'.split()]
print(a[3])
```

```
809
```

Списочные выражения

Считываем числа, записанные в строчку:

```
a = [int(x) for x in '976 929 289 809 7677'.split()]
print(a[3])
evil, good = [int(x) for x in '666 777'.split()]
print(evil, good, sep='\n')
```

```
809
```

Списочные выражения

Считываем числа, записанные в строчку:

```
a = [int(x) for x in '976 929 289 809 7677'.split()]
print(a[3])
evil, good = [int(x) for x in '666 777'.split()]
print(evil, good, sep='\n')
```

```
809
666
777
```

Списочные выражения

Использование метода join вместе со списочным выражением:

```
print(', '.join(str(i) + '^2=' + str(i ** 2) for i in range(1, 10)))
```

Списочные выражения

Использование метода join вместе со списочным выражением:

```
print(', '.join(str(i) + '^2=' + str(i ** 2) for i in range(1, 10)))
```

```
1^2=1, 2^2=4, 3^2=9, 4^2=16, 5^2=25, 6^2=36, 7^2=49, 8^2=64, 9^2=81
```

Решение задачи "Глория Скотт"

Решение задачи "Глория Скотт"

```
message = input()
words = message.split()
secret_words = words[2::3]
secret_message = ' '.join(secret_words)
print(secret_message)
```

Решение задачи "Глория Скотт"

```
message = input()
words = message.split()
secret_words = words[2::3]
secret_message = ' '.join(secret_words)
print(secret_message)
```

или

```
print(' '.join(input().split()[2::3]))
```

Решения серии задач "Списочная квадратура"

Решения серии задач "Списочная квадратура"

```
n = int(input())
a = [i ** 2 for i in range(n)]
for x in a:
    print(x)
```

Решения серии задач "Списочная квадратура"

```
n = int(input())
a = [i ** 2 for i in range(n)]
for x in a:
    print(x)
```

```
n = int(input())
a = [i ** 2 for i in range(n)]
print(' '.join(str(x) for x in a))
```

Решения серии задач "Списочная квадратура"

```
n = int(input())
a = [i ** 2 for i in range(n)]
for x in a:
    print(x)
```

```
n = int(input())
a = [i ** 2 for i in range(n)]
print(' '.join(str(x) for x in a))
```

```
a = [int(x) for x in input().split()]
print(' '.join(str(x ** 2) for x in a))
```


Яндекс