

Методы списков и строк

Методы списков

Операция	Описание	Пример
<code>x in a</code>	Проверка, что <code>x</code> содержится в <code>a</code>	<code>5 in [2, 3, 5]</code>
<code>x not in a</code>	Проверка, что <code>x</code> не содержится в <code>a</code> . То же, что и <code>not(x in a)</code>	<code>5 not in [2, 3, 5]</code>
<code>a + a2</code>	Конкатенация списков, то есть список, в котором сначала идут все элементы <code>a</code> , а затем все элементы <code>a2</code>	<code>[2, 4] + [5, 3] == [2, 4, 5, 3]</code>
<code>a * k</code>	Список <code>a</code> , повторенный <code>k</code> раз	<code>[2, 3] * 3 == [2, 3, 2, 3, 2, 3]</code>
<code>a[n]</code>	<code>n</code> -й элемент списка, отрицательные <code>n</code> — для отсчета с конца	<code>[2, 3, 7][0] == 2</code> <code>[2, 3, 7][-1] == 7</code>
<code>a[start:stop:step]</code>	Срез	<code>[2, 3, 7][:2] == [2, 3]</code>
<code>len(a)</code>	Длина списка	<code>len([2, 3, 7]) == 3</code>
<code>max(a)</code>	Максимальный элемент списка	<code>max([2, 3, 7]) == 7</code>
<code>min(a)</code>	Минимальный элемент списка	<code>min([2, 3, 7]) == 2</code>
<code>sum(a)</code>	Сумма элементов списка	<code>sum([2, 3, 7]) == 12</code>

Операция	Описание	Пример
<code>a.index(x)</code>	Индекс первого вхождения <code>x</code> в <code>a</code> (вызовет ошибку, если <code>x not in a</code> , то есть если <code>x</code> отсутствует в <code>a</code>)	<code>[2, 3, 7].index(7) == 2</code>
<code>a.count(x)</code>	Количество вхождений <code>x</code> в <code>a</code>	<code>[2, 7, 3, 7].count(7) == 2</code>
<code>a.append(x)</code>	Добавить <code>x</code> в конец <code>a</code>	<code>a = [2, 3, 7]</code> <code>a.append(8)</code> <code>a == [2, 3, 7, 8]</code>
<code>a.extend(a2)</code>	Добавить элементы коллекции <code>a2</code> в конец <code>a</code>	<code>a = [2, 3, 7]</code> <code>a.extend([8, 4, 5])</code> <code>a == [2, 3, 7, 8, 4, 5]</code>
<code>del a[n]</code>	Удалить <code>n</code> -й элемент списка	<code>[2, 3, 7]</code> <code>del a[1]</code> <code>a == [2, 7]</code>
<code>del a[start:stop:step]</code>	Удалить из <code>a</code> все элементы, попавшие в срез	<code>a = [2, 3, 7]</code> <code>del a[:2]</code> <code>a == [7]</code>
<code>a.clear()</code>	Удалить из <code>a</code> все элементы (то же, что <code>del a[:]</code>)	<code>a.clear()</code>
<code>a.copy()</code>	Копия <code>a</code> (то же, что <code>a[:]</code>)	<code>b = a.copy()</code>
<code>a += a2</code> <code>a *= k</code>	Заменить содержимое списка на <code>a + a2</code> и <code>a * k</code> соответственно	

Операция	Описание	Пример
<code>a.insert(n, x)</code>	Вставить <code>x</code> в <code>a</code> на позицию <code>n</code> , подвинув последующую часть дальше	<code>a = [2, 3, 7]</code> <code>a.insert(0, 8)</code> <code>a == [8, 2, 3, 7]</code>
<code>a.pop(n)</code>	Получить <code>n</code> -й элемент списка и одновременно удалить его из списка <code>a.pop()</code> == <code>a.pop(-1)</code>	<code>a = [2, 3, 7]</code> <code>a.pop(1) == 3</code> <code>a == [2, 7]</code>
<code>a.remove(x)</code>	Удалить первое вхождение <code>x</code> в <code>a</code> , в случае <code>x not in a</code> — ошибка	<code>a = [2, 3, 7]</code> <code>a.remove(3)</code> <code>a == [2, 7]</code>
<code>a.reverse()</code>	Изменить порядок элементов в <code>a</code> на обратный (перевернуть список)	<code>a = [2, 3, 7]</code> <code>a.reverse()</code> <code>a == [7, 3, 2]</code>
<code>a.sort()</code>	Отсортировать список по возрастанию	<code>a = [3, 2, 7]</code> <code>a.sort()</code> <code>a == [2, 3, 7]</code>
<code>a.sort(reverse=True)</code>	Отсортировать список по убыванию	<code>a = [3, 2, 7]</code> <code>a.sort(reverse = True)</code> <code>a == [7, 3, 2]</code>
<code>bool(a)</code>	Один из способов проверить список на пустоту	

Выделение **жирным** символизирует, что эти методы изменяют сам список `a`

Методы строк

Операция	Описание	Пример
<code>s2 in s</code>	Проверка, что подстрока <code>s2</code> содержится в <code>s</code>	<code>'m' in 'team'</code>
<code>s2 not in s</code>	Проверка, что подстрока <code>s2</code> не содержится в <code>s</code> . То же, что <code>not (s2 in s)</code>	<code>'I' not in 'team'</code>
<code>s + s2</code>	Конкатенация (склейка) строк, то есть строка, в которой сначала идут все символы из <code>s</code> , а затем все символы из <code>s2</code>	<code>'tea' + 'm' == 'team'</code>
<code>s * k</code>	Строка <code>s</code> , повторенная <code>k</code> раз	<code>'oo' * 3 == 'oooooo'</code>
<code>s[n]</code>	<code>n</code> -й элемент строки, отрицательные <code>n</code> — для отсчета с конца	<code>'team'[2] == 'a'</code> <code>team[-1] == 'm'</code>
<code>s[start:stop:step]</code>	Срез строки	<code>'mama'[:2] == 'ma'</code>
<code>len(s)</code>	Длина строки	<code>len('abracadabra') == 11</code>
<code>s.find(s2)</code> <code>s.rfind(s2)</code>	Индекс начала первого или последнего вхождения подстроки <code>s2</code> в <code>s</code> (вернёт -1, если <code>s2 not in s</code>)	<code>s = 'abracadabra'</code> <code>s.find('ab') == 0</code> <code>s.rfind('ab') == 7</code> <code>s.find('x') == -1</code>
<code>s.count(s2)</code>	Количество неперекрывающихся вхождений <code>s2</code> в <code>s</code>	<code>'abracadabra'.count('a') == 5</code>

Операция	Описание	Пример
<code>s.startswith(s2)</code> <code>s.endswith(s2)</code>	Проверка, что <code>s</code> начинается с <code>s2</code> или оканчивается на <code>s2</code>	<code>'abracadabra'.startswith('abra')</code>
<code>s += s2</code> <code>s *= k</code>	Заменить содержимое строки <code>s</code> на <code>s + s2</code> и <code>s * k</code> соответственно	
<code>s.isdigit()</code> <code>s.isalpha()</code> <code>s.isalnum()</code>	Проверка, что в строке <code>s</code> все символы — цифры, буквы (включая кириллические), цифры или буквы, соответственно	<code>'100'.isdigit()</code> <code>'abc'.isalpha()</code> <code>'Е315'.isalnum()</code>
<code>s.islower()</code> <code>s.isupper()</code>	Проверка, что в строке <code>s</code> не встречаются большие буквы, маленькие буквы. Обратите внимание, что для обеих этих функций знаки препинания и цифры дают <code>True</code>	<code>'hello!'.islower()</code> <code>'123PYTHON'.isupper()</code>
<code>s.lower()</code> <code>s.upper()</code>	Строка <code>s</code> , в которой все буквы (включая кириллические) приведены к верхнему или нижнему регистру, т.е. заменены на строчные (маленькие) или заглавные (большие)	<code>'Привет!'.lower() == 'привет!'</code> <code>'Привет!'.upper() == 'ПРИВЕТ!'</code>
<code>s.capitalize()</code>	Строка <code>s</code> , в которой первая буква — заглавная	<code>'привет'.capitalize() == 'Привет'</code>
<code>s.lstrip()</code> <code>s.rstrip()</code>	Строка <code>s</code> , у которой удалены символы пустого пространства (пробелы,	<code>'Привет!'.strip() == 'Привет!'</code>

Операция	Описание	Пример
<code>s.strip()</code>	табуляции) в начале, в конце или с обеих сторон	
<code>s.ljust(k, c)</code> <code>s.rjust(k, c)</code>	Добавляет справа или слева нужное количество символов <code>c</code> , чтобы длина <code>s</code> достигла <code>k</code>	<code>'Привет'.ljust(8, '!')</code> <code>== 'Привет!!'</code>
<code>s.join(a)</code>	<code>a</code> — список строк, тогда <code>s.join(a)</code> — эти строки, «склеенные» через <code>s</code>	<code>'+'.join(['Вася', 'Маша']) == 'Вася+Маша'</code>
<code>s.split(s2)</code>	Список слов строки <code>s</code> (подстрок, разделенных строками <code>s2</code>)	<code>'Раз два три!'.split('a') == ['Р', 'з дв', ' три!']</code>
<code>s.replace(s2, s3)</code>	Строка <code>s</code> , в которой все неперекрывающиеся вхождения <code>s2</code> заменены на <code>s3</code>	<code>'Раз два три!'.replace('а', 'я') == 'Ряз два три!'</code>
<code>list(s)</code>	Список символов из строки <code>s</code>	<code>list('Привет') == ['П', 'р', 'и', 'в', 'е', 'т']</code>
<code>bool(s)</code>	Проверка, что строка не пустая	
<code>int(s)</code> <code>float(s)</code>	Если в строке <code>s</code> записано целое (дробное) число, получить это число, иначе ошибка	<code>int('25') == 25</code>
<code>str(x)</code>	Получить какое-то представление в виде строки любого объекта <code>x</code>	<code>str(25) == '25'</code>

Функции dir и help

Функция `dir()` возвращает список методов любого объекта.

```
print(dir([]))
```

Выведет

```
['__add__', '__class__', '__contains__', '__delattr__',  
 '__delitem__', '__dir__', '__doc__', '__eq__', '__format__',  
 '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__',  
 '__iadd__', '__imul__', '__init__', '__init_subclass__',  
 '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__',  
 '__new__', '__reduce__', '__reduce_ex__', '__repr__',  
 '__reversed__', '__rmul__', '__setattr__', '__setitem__',  
 '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear',  
 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove',  
 'reverse', 'sort']
```

Функция `help()` выводит справку по любому объекту или методу.

```
help([].insert)
```

Выведет

```
Help on built-in function insert:
```

```
insert(index, object, /) method of builtins.list instance
```

```
    Insert object before index.
```

f-строки

f-строки, которые буквально означают formatted string, задаются с помощью литерала `f` перед кавычками. Пример использования:

```
name = "Аркадий"  
age = 14  
print(f"Меня зовут {name} Мне {age} лет.")
```

После выполнения программа выведет.

```
Меня зовут Аркадий. Мне 14 лет.
```

В качестве подставляемого значения внутри фигурных скобок может быть:

- `f"Меня зовут {name}"` `#имя переменной`
- `f"Третий месяц в году -{month[2]}"` `#элемент списка или словаря`
- `f"Имя: {name.upper()}"` `#методы объектов`

f-строки позволяют выполнять базовые арифметические операции

```
f"({x} + {y}) / 2 = {(x + y) / 2}"
```

И даже вызывать функции

```
f"13 / 7 = {round(13/7)}"
```