

Методы списков и строк

Методы списков

	_	_
Операция	Описание	Пример
x in a	Проверка, что х содержится в а	5 in [2, 3, 5]
x not in a	Проверка, что х не содержится в а. То же, что и not (х in a)	5 not in [2, 3, 5]
a + a2	Конкатенация списков, то есть список, в котором сначала идут все элементы а, а затем все элементы а2	[2, 4] + [5, 3] == [2, 4, 5, 3]
a * k	Список а, повторенный k раз	[2, 3] * 3 == [2, 3, 2, 3, 2, 3]
a[n]	n-й элемент списка, отрицательные n — для отсчета с конца	[2, 3, 7][0] == 2[2, 3, 7][-1] == 7
a[start:stop: step]	Срез	[2, 3, 7][:2] == [2, 3]
len(a)	Длина списка	len([2, 3, 7]) == 3
max(a)	Максимальный элемент списка	max([2, 3, 7]) == 7
min(a)	Минимальный элемент списка	min([2, 3, 7]) == 2
sum(a)	Сумма элементов списка	sum([2, 3, 7]) == 12



Операция	Описание	Пример
a.index(x)	Индекс первого вхождения х в а (вызовет ошибку, если х not in a, то есть если х отсутствует в а)	[2, 3, 7].index(7) == 2
a.count(x)	Количество вхождений x в а	[2, 7, 3, 7].count(7) == 2
a.append(x)	Добавить x в конец а	<pre>a = [2, 3, 7] a.append(8) a == [2, 3, 7, 8]</pre>
a.extend(a2)	Добавить элементы коллекции a2 в конец a	<pre>a = [2, 3, 7] a.extend([8, 4, 5]) a == [2, 3, 7, 8, 4, 5]</pre>
del a[n]	Удалить n-й элемент списка	[2a =, 3, 7] del a[1] a == [2, 7]
<pre>del a[start:stop: step]</pre>	Удалить из а все элементы, попавшие в срез	a = [2, 3, 7] del a[:2]a == [7]
a.clear()	Удалить из а все элементы (то же, что dela[:])	a.clear()
a.copy()	Копия а (то же, что а [:])	b = a.copy()
a += a2 a *= k	Заменить содержимое списка на а + а2 и а * k соответственно	



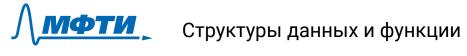
Операция	Описание	Пример
a.insert(n, x)	Вставить х в а на позицию n, подвинув последующую часть дальше	<pre>a = [2, 3, 7] a.insert(0, 8) a == [8, 2, 3, 7]</pre>
a.pop(n)	Получить n-й элемент списка и одновременно удалить его из списка a.pop() == a.pop(-1)	a = [2, 3, 7] a.pop(1) == 3 a == [2, 7]
a.remove(x)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

Выделение жирным символизирует, что эти методы изменяют сам список а

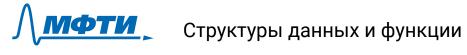


Методы строк

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



Операция	Описание	Пример
s.startswith(s 2) s.endswith(s2)	Проверка, что s начинается с s2 или оканчивается на s2	'abracadabra'.startswith ('abra')
s += s2s *= k	Заменить содержимое строки s на s + s2 и s * k соответственно	
<pre>s.isdigit() s.isalpha() s.isalnum()</pre>	Проверка, что в строке s все символы — цифры, буквы (включая кириллические), цифры или буквы, соответственно	'100'.isdigit() 'abc'.isalpha() 'E315'.isalnum()
s.islower() s.isupper()	Проверка, что в строке s не встречаются большие буквы, маленькие буквы. Обратите внимание, что для обеих этих функций знаки препинания и цифры дают True	<pre>'hello!'.islower() '123PYTHON'.isupper()</pre>
s.lower() s.upper()	Строка s, в которой все буквы (включая кириллические) приведены к верхнему или нижнему регистру, т.е. заменены на строчные (маленькие) или заглавные (большие)	'Привет!'.lower() == 'привет!' ''Привет!'.upper() == 'ПРИВЕТ!'
s.capitalize()	Строка s, в которой первая буква — заглавная	'привет'.capitalize() == 'Привет'
s.lstrip() s.rstrip()	Строка s, у которой удалены символы пустого пространства (пробелы,	' Привет! '.strip() == 'Привет!'



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



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"Meня зовут {name} Mhe {age} лет.")
```

После выполнения программа выведет.

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

В качестве подставляемого значения внутри фигурных скобок может быть:



МФТИ Структуры данных и функции

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

#имя переменной

#методы объектов

f-строки позволяют выполнять базовые арифметические операции

$$f''(\{x\} + \{y\}) / 2 = \{(x + y) / 2\}"$$

И даже вызывать функции

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