Списки Python — функции и методы Python list()

Списки объявляются в квадратных скобках [ ].

z =	[3,	7,	4,	2]
index	0	1	2	3

Вторая строка этой таблицы списка python — индекс элемента списка.

```
z = [3, 7, 4, 2]  # Создание списка
```

В python списки хранят упорядоченный набор элементов, которые могут быть разных типов. В примере, указанном выше элементы имеют один и тот же тип int. Не обязательно все элементы должны быть одного типа.

```
# Создание списка с разными типам данных
s = [3, True, 'Витя', 2.0]
```

Этот список содержит int, bool, string и float.

Доступ к элементам списка

Каждый элемент имеет присвоенный ему индекс. Важно отметить, в python индекс первого элемента в списке — 0.

z =	[3,	7,	4,	2]
index	0	1	2	3

Элемент с индексом 0 (выделен синим)

```
z = [3, 7, 4, 2]  # создаем список
# обращение к первому элементу списка с индексом 0
print(z[0])
# элемент с индексом 0 -> 3
```

Также поддерживается отрицательная индексация. Отрицательная индексация начинается с конца. Иногда её удобнее использовать для получения последнего элемента в списке, потому что не нужно знать длину списка, чтобы получить доступ к последнему элементу.

z =	[3,	7,	4,	2]
index	0	1	2	3
negative index	-4	-3	-2	-1

Элемент с индексом -1 (выделен синим)

```
КОПИРОВАТЬ
# выведите последний элемент списка
>>> print(z[-1])
2
```

Вы также можете получить доступ к одному и тому же элементу с использованием положительных индексов (как показано ниже). Альтернативный способ доступа к последнему элементу в списке z.

```
КОПИРОВАТЬ
>>> print(z[3])
2
```

Срезы(slice) списка
Срезы хороши для получения подмножества значений с вашего списка. На примере кода, приведенного ниже, он вернет список с элементами из индекса 0 и не включая индекс 2.

z =	[3,	7,	4,	2]
index	0	1	2	3

Первый индекс пишется (до : включительно), а последний (после :) и не учитывается

```
КОПИРОВАТЬ
# Создайте список
z = [3, 7, 4, 2]
# Вывод элементов с индексом от 0 до 2 (не включая 2)
print(z[0:2])
# вывод: [3, 7]
```

z =	[3,	7,	4,	2]
index	0	1	2	3

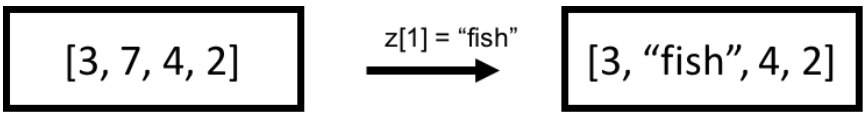
```
КОПИРОВАТЬ
# Все, кроме индекса 3
>>> print(z[:3])
[3, 7, 4]
```

z =	[3,	7,	4,	2]
index	0	1	2	3

Код, указанный ниже возвращает список с элементами начиная с индекса 1 до конца.

```
КОПИРОВАТЬ
# начиная с индекса 1 до конца списка
>>> print(z[1:])
[7, 4, 2]
```

Изменение элементов в списке



Списки в Python изменяемы. Это означает, что после создания списка можно обновить его отдельные элементы.

```
КОПИРОВАТЬ
z = [3, 7, 4, 2]  # Создание списка
# Изменяем элемент с индексом 1 на строку 'fish'
z[1] = 'fish'
print(z)


[3, 'fish', 4, 2]
```

Методы и функции списков python

У списков Python есть разные методы, которые помогают в программировании. В этом разделе рассматриваются все методы списков.

Метод Index

Метод index возвращает положение первого индекса, со значением x. В указанном ниже коде, он возвращает назад 0.

```
КОПИРОВАТЬ
# Создайте список
>>> z = [4, 1, 5, 4, 10, 4]
>>> print(z.index(4))
0
```

z =	[4,	1,	5,	4,	10,	4]
index	0	1	2	3	4	5

Вы также можете указать, откуда начинаете поиск.

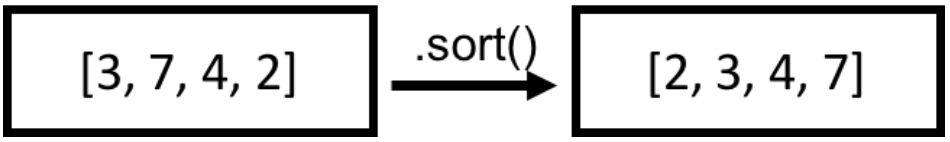
```
КОПИРОВАТЬ
>>> print(z.index(4, 3))
3
```

z =	[4,	1,	5,	4,	10,	4]
index	0	1	2	3	4	5

Метод Count
Метод count работает так, как звучит. Он считает количество раз, когда значение появляется в списке.

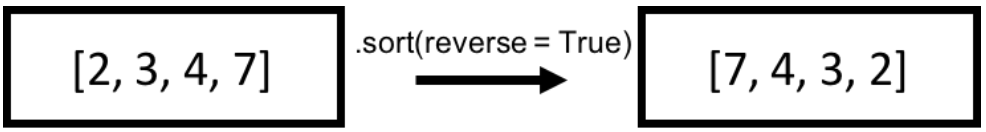
```
КОПИРОВАТЬ
>>> random_list = [4, 1, 5, 4, 10, 4]
>>> print(random_list.count(4))
3
```

Метод Sort



Сортировка списка — фактическим кодом будем: z.sort()
Метод sort сортирует и меняет исходный список.

```
z = [3, 7, 4, 2]
z.sort()
print(z)


[2, 3, 4, 7]
```



Сортировка списка с наибольшего значения к наименьшему
Вышеуказанный код сортирует список чисел от наименьшего к наибольшему. Код, указанный ниже, показывает, как вы можете сортировать список от наибольшего к наименьшему.

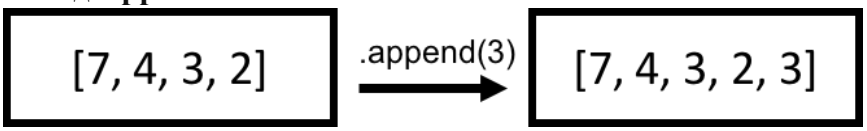
```
КОПИРОВАТЬ
# Сортировка и изменение исходного списка от наивысшего к наименьшему
z.sort(reverse = True)
print(z)


[7, 4, 3, 2]
```

Следует отметить, что вы также можете отсортировать **список строк** от А до Я (или A-Z) и наоборот.

```
КОПИРАТЬ
# Сортировка списка строками
names = ["Стив", "Рейчел", "Майкл", "Адам", "Джессика", "Лестер"]
names.sort()
print(names)
```

```
['Адам', 'Джессика', 'Лестер', 'Майкл', 'Рейчел', 'Стив']
```

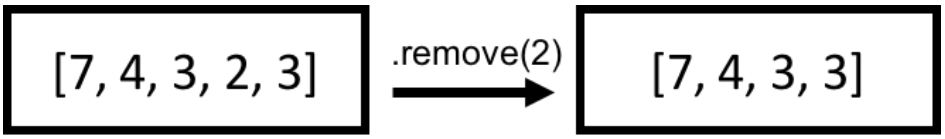
Метод Append



Добавьте значение 3 в конец списка
Метод append добавляет элемент в конец списка. Это происходит на месте.ОВАТЬ

```
z = [7, 4, 3, 2]
z.append(3)
print(z)


[7, 4, 3, 2, 3]
```

Метод Remove



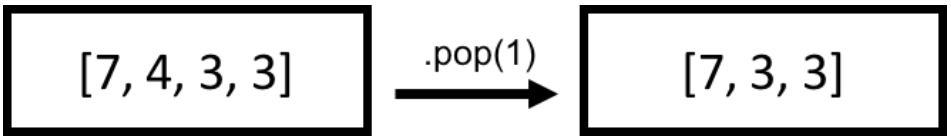
Метод remove удаляет первое вхождение значения в списке.

```
КОПИРОВАТЬ
z = [7, 4, 3, 2, 3]
z.remove(2)
print(z)
```

Код удаляет первое вхождение значения 2 из списка z.

```


[7, 4, 3, 3]
```

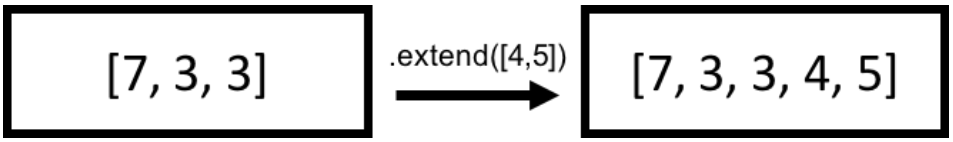
Метод Pop



z.pop(1) удаляет значение в индексе 1 и возвращает значение 4
Метод pop удаляет элемент в указанном индексе. Этот метод также вернет элемент, который был удален из списка. В случае, если вы не указали индекс, он по умолчанию удалит элемент по последнему индексу.

```
КОПИРОВАТЬ
z = [7, 4, 3, 3]
print(z.pop(1))
print(z)

4
[7, 3, 3]
```

Метод Extend



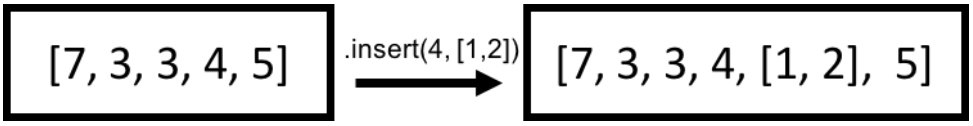
Метод extend расширяет список, добавляя элементы. Преимущество над append в том, что вы можете добавлять списки.
Добавим [4, 5] в конец z:РОВАТЬ

```
z = [7, 3, 3]
z.extend([4,5])
print(z)

[7, 3, 3, 4, 5]
```

То же самое можно было бы сделать, используя +.

```
>>> print([1,2] + [3,4])
[7, 3, 3, 4, 5]
```

Метод Insert



Вставляет [1,2] с индексом 4
Метод insert вставляет элемент перед указанным индексом.

```
z = [7, 3, 3, 4, 5]
z.insert(4, [1, 2])
print(z)

[7, 3, 3, 4, [1, 2], 5]
```

Простые операции над списками

Метод	**Описаниее**
x in s	True если элемент x находится в списке s
x not in s	True если элемент x не находится в списке s
s1 + s2	Объединение списков s1 и s2
s * n , n * s	Копирует список s n раз
len(s)	Длина списка s, т.е. количество элементов в s
min(s)	Наименьший элемент списка s
max(s)	Наибольший элемент списка s
sum(s)	Сумма чисел списка s
for i in list()	Перебирает элементы слева направо в **цикле for**

Примеры использование функций со списками:

```
КОПИРОВАТЬ
>>> list1 = [2, 3, 4, 1, 32]
>>> if 2 in list1:
```

```
            print (True)
 # 2 в list1?
True
>>> if 33 not in list1:
            print(True)
# 33 не в list1?
True
>>> print(len(list1))  # количество элементов списка
5
>>> print(max(list1))  # самый большой элемент списка
32
>>> print(min(list1))  # наименьший элемент списка
1
>>> print(sum(list1)) # сумма чисел в списке
42
# генератор списков python (list comprehension)
>>> x =  [i for i in range(10)]
>>> print(x)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> print(list1.reverse()) # разворачивает список
[32, 1, 4, 3, 2]
```

Операторы + и * для списков

\+ объединяет два списка.

```
КОПИРОВАТЬ
list1 = [11, 33]
list2 = [1, 9]
list3 = list1 + list2
print(list3)


[11, 33, 1, 9]
```

* копирует элементы в списке.

```
КОПИРОВАТЬ
list4 = [1, 2, 3, 4]
list5 = list4 * 3
print(list5)

[1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4]
```

Оператор in и not in

Оператор in проверяет находится ли элемент в списке. При успешном результате он возвращает True, в случае неудачи, возвращает False .

```
КОПИРОВАТЬ
>>> list1 = [11, 22, 44, 16, 77, 98]
>>> if 22 in list1:
        print(True)
True
```

Аналогично not in возвращает противоположный от оператора in результат.

```
>>> if 22 not in list1:
        print(True)

False
```

Итерация по списку с использованием цикла for

Список — последовательность. Ниже способ, которые вы можете использовать для цикла, чтобы перебрать все элементы списка.

```
list1 = [1,2,3,4,5]
for i in list1:
    print(i, end=" ")


1 2 3 4 5
```

Преобразование списка в строку

Как преобразовать список в строку?
Для преобразования списка в строку используйте метод join(). В Python это выглядит так: ",".join(["a", "b", "c"]) -> "a,b,c".
Разделитель пишут в кавычках перед join, в список должен состоять из строк.

Вот несколько полезных советов для преобразования списка в строку (или любого другого итерабельного, такого как tuple).

Во-первых, если это список строк, вы можете просто использовать join() следующим образом.

```
mylist = ['spam', 'ham', 'eggs']
print('  '.join(mylist))


‘spam ham eggs’
```

Используя тот же метод, вы можете также сделать следующее:

```
>>> print('\n'.join(mylist))
spam
ham
eggs
```

Однако этот простой метод не работает, если список содержит не строчные объекты, такие как целые числа. Если вы просто хотите получить строку с разделителями-запятыми, вы можете использовать этот шаблон:

```
list_of_ints = [80, 443, 8080, 8081]
print(str(list_of_ints).strip('[]'))


80, 443, 8080, 8081
```

Или же этот, если ваши объекты содержат квадратные скобки:

```
>>> print(str(list_of_ints)[1:-1])
80, 443, 8080, 8081
```

В конце концов, вы можете использовать map() чтобы преобразовать каждый элемент в список строки и затем присоединиться к ним:

```
КОПИРОВАТЬ
>>> print(', '.join(map(str, list_of_ints)))
80, 443, 8080, 8081
>>> print('\n'.join(map(str, list_of_ints)))
80
443
8080
8081
```

Тест на знание списков в Python

Как создать список?

- l = list(1, 2, 3)
- l = [1, 2, 3]
- Все варианты верны
- l = list[1, 2, 3]