

## Конспект

### Списки


**Список** – это структура данных, которая содержит упорядоченный набор элементов, т.е. хранит последовательность элементов.

Это легко представить, если вспомнить список покупок, в котором перечисляется, что нужно купить, с тем лишь исключением, что в списке покупок каждый элемент обычно размещается на отдельной строке, тогда как в Python они разделяются запятыми.

Список элементов должен быть заключен в квадратные скобки, чтобы Python понял, что это список.

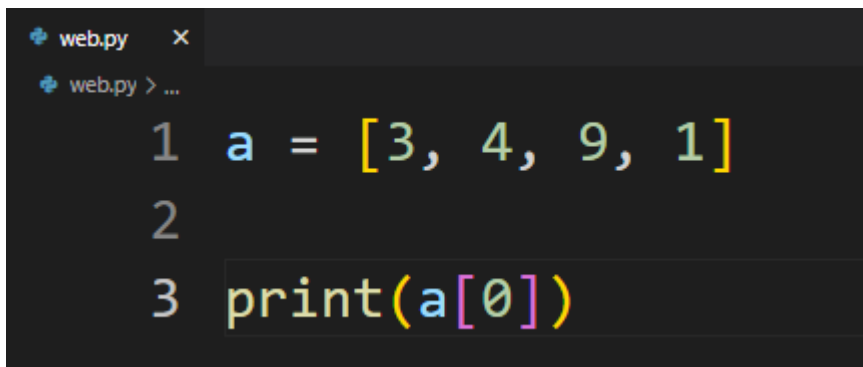
Как только список создан, можно добавлять, удалять или искать элементы в нём. Поскольку элементы можно добавлять и удалять, мы говорим, что список – это изменяемый тип данных, т.е. его можно модифицировать.

Создадим список и у каждого из элементов есть свой индекс:



```
syn.py x
Users > sachemmark > Desktop > Syn
1 a = [3, 4, 9, 1]
2 |
3 # 3 4, 9, 1
4 # 0 1 2 3
```

Чтобы обратиться к какому либо элементу массива, мы можем сделать такую штуку:



```
web.py x
web.py > ...
1 a = [3, 4, 9, 1]
2
3 print(a[0])
```

Так же важно понимать что есть отрицательная индексация.

```
web.py x
web.py > ...
1 a = [3, 4, 9, 1]
2
3 print(a[-1])
```

Теперь попробуем обратиться к индексу которого не существует.

То есть мы можем обращаться только к тем элементам которые существуют.

Теперь выборочно изменим один из элементов списка.

```
web.py x
web.py > ...
1 a = [3, 4, -150, 1, 2]
2 # 3 4 -150 1 2
3 # 0 1 2 3 4
4
5 a[1] = 100
6 print(a)
```

Еще мы можем добавлять элементы в список:

**append()** позволяет добавлять элемент в конце списка, который уже существует.

```
web.py x
web.py > ...
1 a = [3, 4, -150, 1, 2]
2 # 3 4 -150 1 2
3 # 0 1 2 3 4
4
5 a.append(190)
6 print(a[-1])
```

Мы можем узнать размер нашего списка.

**len()** возвращает длину (количество элементов) в объекте/

```
web.py x
web.py > ...
1 a = [3, 4, -150, 1, 2]
2 # 3 4 -150 1 2
3 # 0 1 2 3 4
4
5 print(len(a))
```

Теперь добавим значение в конец и изменим список.

```
web.py x
web.py > ...
1 a = [3, 4, -150, 1, 2]
2 # 3 4 -150 1 2
3 # 0 1 2 3 4
4 a.append(7)
5 print(len(a))
```

Выведем все элементы массива:

```
4
5 print(a)
6 print(*a)
7 print(*a, sep= "/")
```

Мы изучили несколько методов. Теперь изучим еще методы работы со списками.

**insert(i, x)** вставляет объект x в последовательность по индексу i.

```
web.py x
web.py > ...
1 a = [3, 4, -150, 1, 2]
2 # 3 4 -150 1 2
3 # 0 1 2 3 4
4
5 a.insert(2, 99)
6 print(*a)
7
```

Удалим элемент.

**pop()** возвращает значение элемента с индексом *i* , а также удаляет его из последовательности *sequence* . Необязательный аргумент - индекс *i* по умолчанию равен -1 . Так что по умолчанию эта операция производит действие с последним элементом последовательности.

Теперь удаляем последний элемент.

```
web.py x
web.py > ...
1 a = [3, 4, -150, 1, 2]
2 # 3 4 -150 1 2
3 # 0 1 2 3 4
4
5 a.pop()
6 print(*a)
7
```

Теперь удалим не последний элемент, а любой другой.

**a.pop(3)**

**Допустим нам нужно очистить наш список.**

```
web.py x
web.py > ...
1 a = [3, 4, -150, 1, 2]
2 # 3 4 -150 1 2
3 # 0 1 2 3 4
4
5 a.clear()
6 print(*a)
7
```

**Теперь перевернем наш список.**

Функция `reverse()` в Python возвращает обратный итератор из указанного аргумента последовательности. Входной аргумент должен быть последовательностью, например кортежем, списком, строкой и т. д. Возвращаемый объект имеет обратный тип и является итератором

```
web.py x
web.py > ...
1 a = [3, 4, -150, 1, 2]
2 # 3 4 -150 1 2
3 # 0 1 2 3 4
4
5 a.reverse()
6 print(*a)
7
```

**Чтобы посчитать количество элементов используется функция `count`.**

```
web.py x
web.py > ...
1 a = [3, 4, -150, 1, 2, 4, 4, 4, 4]
2 # 3 4 -150 1 2
3 # 0 1 2 3 4
4
5 print(a.count(4))
6
```

**Научимся вводить наши списки.**

```
web.py x
web.py > ...
1 n = int(input("Введите количество значений списка: "))
2 spisok = []
3
4 for i in range(n):
5     a = int(input("Значения списка: "))
6     spisok.append(a)
7 print(spisok)
8
```

ввод наших значений.

for i in range - так как мы хотим чтобы список выполнялся n раз, то есть i будет принимать значения от 0 до n-1. список повторится от i до n-1.

далее вводим значения списка и добавляем его в конец

и выводим список.

Теперь сделаем ввод в одну строчку:

```
spisok = list(map(int, input().split()))  
print(spisok)
```

Input - ВВОД

Split - разделитель строки

Int - тип данных

map - применяет int к каждому значению из input split.

list - делает из всего этого список.

Это выглядит довольно страшно, но со временем я думаю вы разберетесь.

## Теперь поговорим про создание списков.

Создадим список из 100 элементов, каждый из которых будет равен 3.

Первый и не самый удобный вариант-

```
1 a = [3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
```

Второй способ:

```
1 a = [3 for i in range(10)]
2 print(a)
3 print(len(a))
4
```

Первое значение - начальное значение всех элементов списка.

С помощью фор мы пишем сколько раз мы запишем это значение.



---

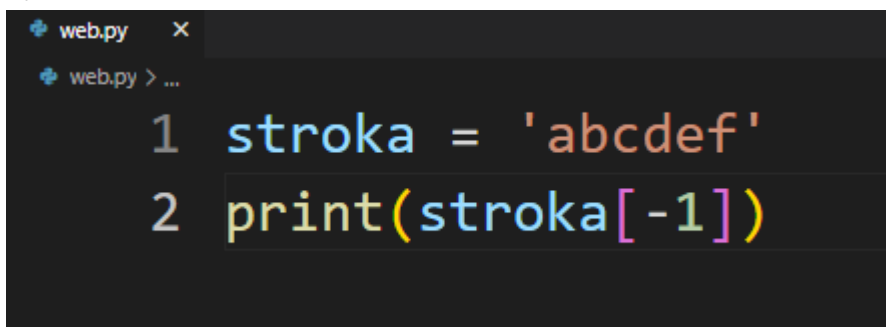
## Строки

---

Мы закончили этап со списками теперь перейдем к строкам, если что-то не понятно на данном этапе - ничего страшного мы еще будем это разбирать и повторять.

Строки достаточно похожи на списки, у них также есть индексация, но так же есть некоторые особенности которые мы сейчас рассмотрим.

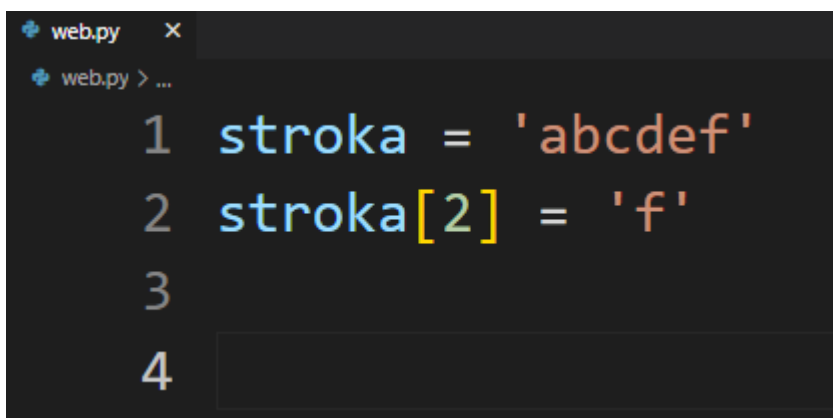
Тут также есть индексация:

A screenshot of a Python IDE window titled 'web.py'. The code shows two lines: '1 stroka = 'abcdef'' and '2 print(stroka[-1])'. The string 'abcdef' is highlighted in orange, and the index '-1' is highlighted in purple.

```
1 stroka = 'abcdef'
2 print(stroka[-1])
```

Теперь посмотрим отличия от списков:

Если допустим попробуем изменить один символ на другой у нас этого не получится.

A screenshot of a Python IDE window titled 'web.py'. The code shows three lines: '1 stroka = 'abcdef'', '2 stroka[2] = 'f'', and '3'. Line 4 is empty. The string 'abcdef' is highlighted in orange, and the index '2' is highlighted in yellow.

```
1 stroka = 'abcdef'
2 stroka[2] = 'f'
3
4
```

Потому что строка является неизменяемым типом данных.

Мы можем её пересоздать.

Строки можно складывать, то есть производить конкатинацию:

```
web.py x
web.py > ...
1 stroka = 'abcdef'
2 stroka2 = 'fefcba'
3 storka3 = stroka + stroka2
4
5 print(storka3)
```

Также можно подсчитать количество символов в строк:

```
web.py > ...
1 stroka = 'abcdef'
2 stroka2 = 'fefcba'
3 storka3 = stroka + stroka2
4
5 print(len(storka3))
```