

## Список (list)

Списки в Python - упорядоченные изменяемые коллекции объектов произвольных типов (почти как массив, но типы могут отличаться).

Все методы списка (в ipython или тетради): **dir(list)** или **help(list)**

Help по 1 методу (например, append): list.append?

## Создать список

Пустой список:

```
a = []
a = ['apple', 'banana', 'wildberry']
b = [12, 34, -5, 16]
c = [12, 'apple', [3.14, 9.81], 'orange' ]
d = list('hello')
```

## Срезы (Slice)

- s[a:b:step] - подсписок с элементами от номера a (включительно) до номера b (не включительно), с шагом step

Можно изменять список при помощи срезов

```
>>> a = [1, 2, 3]
>>> a
[1, 2, 3]           # все элементы числа
>>> a[2] = [4, 5]
>>> a
[1, 2, [4, 5]]      # последний элемент - список, а не число
```

Изменять часть списка (удалить все, что слева, вставить в список то, что справа):

```
>>> a = [1, 2, 3]
>>> a
[1, 2, 3]
>>> a[1:2] = [4, 5]
>>> a
[1, 4, 5, 3]
```

💡 Удаляемая и вставляемая части могут быть разной длины.

## Методы работы со списками

<u>Python</u>	<u>Получилось</u>	<u>Комментарий</u>
a=[1, 2, 3], a[0]=7	[7, 2, 3]	элемент списка
len([1, 2, 3])	3	Длина
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	+ Склеить
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	* Повторить
[1, 2, 3] is [1, 2, 3]	True	Равны?
3 in [1, 2, 3]	True	Проверить, что есть
7 not in [1, 2, 3]	True	Проверить, что нет

for x in [1, 2, 3]: print x	1 2 3	Напечатать все элементы
--------------------------------	-------	-------------------------

## range(from, to, step)

```
>>> range(1, 5)
[1, 2, 3, 4]
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(1, 10, 2)
[1, 3, 5, 7, 9]
>>> range(20, 4, -5)
[20, 15, 10, 5]
>>> range(10, 20, -5)
[]
```

## Встроенные функции для работы со списками

Функция	Что делает
len(a)	Длина списка a
max(a)	Максимальный элемент списка a
min(a)	Минимальный элемент списка a
sum(a)	Сумма чисел (работает только с числами)
cmp(a, b)	Сравнить списки a и b
a = str.split(delimiter)	Сделать список a, разбив строку str на элементы по разделителям delimiter; по умолчанию делится по пробелам
s = str.join(a)	сделать из списка a строку, между элементами вставлять подстроку str
b = map(func, a)	применить функцию func к каждому элементу списка a b - не список, а map; чтобы его печатать, сделайте из него список print(list(b))

## Методы списков

Метод	Что делает
a.append(x)	Добавляет элемент x в список a
a.insert(i, x)	вставляет x на место номер i
a.extend(b)	a = a + b
a.remove(x)	удаляет x из списка a
a.count(x)	сколько раз элемент x входит в список a
a.index(x)	индекс первого вхождения x в список a или исключение (exception) <a href="#">ValueError?</a>
x = a.pop()	Удаляет последний элемент из списка a, возвращает этот элемент
a.reverse()	Лист в обратном порядке
a.clear()	Очищает список, то же самое, что del a[:]

<code>a.sort()</code>	Сортирует список
<code>a.sort(функция)</code>	Сортирует список элементов, пользуясь для "взвешивания" элемента функцией

## Разница между `append` и `extend`

```

a = [1, 2, 3]
a.append([4, 5])
print(a)           # [1, 2, 3, [4, 5]] в списке 4 элемента, последний элемент
- список [4, 5]

b = [1, 2, 3]
b.extend([4, 5])
print(b)           # [1, 2, 3, 4, 5] в списке 5 элементов

```

## Разница между `append` и `+`

Конкатенация (+) создает новый объект, а метод `append` нет. Поэтому `append` работает быстрее.

`a.append(x)` в конец работает как `a[len(a):] = [x]`

`a[:0] = [x]` - добавить в начало списка. Работают так же быстро, как `append`.

## В списках хранятся ссылки

В списке хранятся только ссылки на объекты.


Тут должен быть рисунок

```

m = [[1, 2, 3], [1, 2, 3], [1, 2, 3]]
print(m)           # [[1, 2, 3], [1, 2, 3], [1, 2, 3]]
m[0][0] = 10
print(m)           # [[10, 2, 3], [1, 2, 3], [1, 2, 3]]

a = [1, 2, 3]
m = [a, a, a]      # m содержит 3 ссылки на один и тот же список a
print(m)           # [[1, 2, 3], [1, 2, 3], [1, 2, 3]]
m[0][0] = 10
print(m)           # [[10, 2, 3], [10, 2, 3], [10, 2, 3]]

```

 Как изменить код последнего примера, чтобы для создания списка использовался все тот же список `a`, но изменение элемента `m[0][0]=10` не изменяло другие элементы матрицы?

## Копирование списка

Копировать список `a` можно:

- `list(a)`
- `a[:]`
- `a.copy()`

## Вложенные списки (как хранить матрицу)

Списки могут быть вложенными.

## Разница между `list()` и `[]`

Для создания пустого списка разницы нет.

Для создания списка из чего-то что можно перебрать по элементам `m`, используем `list(m)`, так как `[m]` создаст список с одним элементом `m`, а не список из элементов, из которых состоит `m`.

```
a1 = list(1, 2, 3) # ошибка, так нельзя, нужно передать ссылку на то, что  
можно перечислить  
a2 = [1, 2, 3]  
  
m = map(int, '10 20 30'.split())  
a3 = list(m)  
print(a3)          # [10, 20, 30]  
a4 = [m]  
print(a4)          # [<map object at 0xffd76f50>]
```

## `del` - удаление элемента, части списка или всей переменной

- `del` элемент
- `del` часть списка
- `del` переменная

```
a = [-1, 1, 66.25, 333, 333, 1234.5]  
del a[0]  
print(a)          # [1, 66.25, 333, 333, 1234.5]  
  
del a[2:4]  
print(a)          # [1, 66.25, 1234.5]  
  
del a[:]  
или a = []  
print(a)          # []
```

Можно удалять переменные

```
a = [-1, 1, 66.25, 333, 333, 1234.5]  
del a  
print(a)          # ошибка! переменной a больше нет!
```

## Генераторы списков (list comprehensions)

Про генераторы вообще поговорим в отдельном уроке. Сейчас получаем рецепты.

Общий вид как сделать список:

```
[выражение for переменная in последовательность]  
или  
[выражение for переменная in последовательность if условие]
```

Как сделать список, написав меньше кода?

Привычный вариант:

```
a = []  
for x in range(5):
```

```
a.append(x**2)
```

Если функция достаточно сложная, можно ее написать отдельно:

```
def sqr(x):  
    return x**2  
a = list(map(sqr, range(5)))
```

А если простая, то записать через lambda:

```
a = list(map(lambda x: x**2, range(10)))
```

Или, как большинство программистов на питоне, использовать list comprehensions

```
a = [x**2 for x in range(5)]           # [0, 1, 4, 9, 16]
```

Примеры:

```
>>> [(x, y) for x in [1,2,3] for y in [3,1,4] if x != y]  
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
```

то же самое:

```
>>> combs = []  
>>> for x in [1,2,3]:  
...     for y in [3,1,4]:  
...         if x != y:  
...             combs.append((x, y))  
...  
>>> combs  
[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]
```

Обратите внимание на порядок for и if.

```
>>> vec = [-4, -2, 0, 2, 4]  
>>> [x**2 for x in vec]           # create a new list with the values  
doubled  
[-8, -4, 0, 4, 8]  
  
>>> [x for x in vec if x >= 0]    # filter the list to exclude negative  
numbers  
[0, 2, 4]  
  
>>> [abs(x) for x in vec]         # apply a function to all the elements  
[4, 2, 0, 2, 4]  
  
                                # call a method on each element  
>>> freshfruit = [' banana', ' loganberry ', 'passion fruit ']  
>>> [weapon.strip() for weapon in freshfruit]  
['banana', 'loganberry', 'passion fruit']  
  
>>> [(x, x**2) for x in range(6)] # create a list of 2-tuples like  
(number, square)  
[(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25)]  
>>> # the tuple must be parenthesized, otherwise an error is raised  
>>> [x, x**2 for x in range(6)]  
File "<stdin>", line 1, in <module>  
    [x, x**2 for x in range(6)]  
        ^  
SyntaxError: invalid syntax
```

```
>>> vec = [[1,2,3], [4,5,6], [7,8,9]]
>>> [num for elem in vec for num in elem]    # flatten a list using a listcomp
with two 'for'
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

## Nested List Comprehensions (генерация вложенных списков)

Дана матрица

```
>>> matrix = [
...     [1, 2, 3, 4],
...     [5, 6, 7, 8],
...     [9, 10, 11, 12],
... ]
```

Надо ее транспонировать.

```
t = [[row[i] for row in m] for i in range(4)]
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

Чуть подробнее:

```
transposed = []
for i in range(4):
    transposed.append([row[i] for row in matrix])
```

еще подробнее:

```
transposed = []
for i in range(4):
    # the following 3 lines implement the nested listcomp
    transposed_row = []
    for row in matrix:
        transposed_row.append(row[i])
    transposed.append(transposed_row)
```

Дополнительный материал:

```
list(zip(*matrix))
```

Разберитесь сами, как сделать то же самое с помощью функции [zip\(\)](#)

## Примеры

### Сколько раз входит число 7 в список a

```
def mycount(a):
    c = 0
    for x in a:
        if x == 7:
            c = c + 1
    return c

a = [7, 9, -3, 7, 2, 1, 7]
print(mycount(a))
```

То же самое можно сделать, вызвав стандартную функцию count.

```
a.count(7)
```

## Номер первого вхождения числа 7 в список или -1, если числа 7 в нем нет

```
def first(a):
    for i in range(len(a)):
        if a[i] == 7:
            return i          # нашли первый раз число 7 - можно дальше не
                                # работать и уходить
    return -1                 # перебрали весь список и число не нашли,
                                # возвращаем -1

a = [17, 9, -3, 7, 2, 1, 7]
print(first(a))              # 3
print(first([1, 10, -4]))    # -1
```

Почти то же самое может сделать функция `index`. Но если числа в списке нет, то функция не будет ничего возвращать, а вызовет исключение `ValueError?` (мы еще не знаем, что делать с исключениями)

## Номер последнего вхождения числа 7 в список или -1, если числа 7 в списке нет

```
def last(a):
    ilast = -1                # в эту переменную запишем номер очередного
                                # найденного числа 7
    for i in range(len(a)):
        if a[i] == 7:         # нашли еще одно число 7
            ilast = i         # запомнили его номер в ilast, прежний номер в
                                # ilast стерли
    return ilast              # закончили перебирать весь список
                                # тут будет номер последней 7 или -1 (если 7
                                # нет)

a = [7, 9, -3, 7, 2, 1, 17]
print(last(a))               # 3
print(last([1, 10, -4]))     # -1
```

То же самое можно получить, если искать число в перевернутом списке.

## Поменять два элемента списка местами

Поменяем `a[0]` и `a[1]` местами:

```
a = [7, 9, -3, 47, 2, 1, 17]
a[0], a[1] = a[1], a[0]
print(a)                      # [9, 7, -3, 47, 2, 1, 17]
```

## Списки списков (матрицы)

### Создаем список списков (

Сделаем таблицу (матрицу) из 3 строк и 4 столбцов и присвоим ячейкам числа от 1 до 12

```
m = [
    [1, 2, 3, 4],          # строка 0
    [5, 6, 7, 8],          # строка 1
    [9, 10, 11, 12]         # строка 2
```

```
]
```

К ячейке мы достаемся по номеру строки `irow` и номеру столбца `icol`

**`m[irow][icol]`**

```
m[1][2] = 100      # вместо числа 7 в таблице число 100
```

## Печатаем матрицу. 3 разных способа.

### Способ 1. Изменяем НОМЕР строки и НОМЕР столбца

```
print('variant 1')
for irow in range(len(m)):                # длина 1D списка m -
    количество строк                      # длина списка m[irow]
    for icol in range(len(m[irow])):      # печатаем 1 число и ставим пробел
        print ( m[irow][icol], end=' ')  # после печати 1 строки
    print()                                # переходим на новую строку
```

`irow`, `icol` - номер строки и номер столбца

### Способ 2. Берем строку и элемент (число) в строке

```
print('variant 2')
for row in m:                             # row - это строка, первая строка [1,
    2, 3, 4]
    for x in row:                          # x - число в этой строке row
        print ( x, end=' ')
    print()
```

Никаких номеров, только список `row` и элемент этого списка `x`.

### Способ 3. Печатаем строку (список) через оператор \*

```
print('variant 3')
for row in m:
    print(*row)
```

## Читаем матрицу целых чисел

Первое число - сколько строк будет в таблице.

```
m = []
n = int(input())
for irow in range(n):
    m.append( list(map(int, input().split())) )
```

Или через list comprehension

```
n = int(input())
m = [list(map(int, input().split())) for irow in range(n)]
```

Если `n` не найдено, но с клавиатуры задают 1 таблицу, то

```
import sys
m = []
for line in sys.stdin:
    m.append(list(map(int, line.split())))
```

Или в одну строку:



```
import sys
m = [list(map(int, line.split())) for line in sys.stdin ]
```

Если вы понимаете, что делает этот код, можете им пользоваться.

## Задачи (списки)

### 1 - Срезы строки

Дан список `a = [10, 7, -6, 11, 13, 5, 1, 8, 13]`

Что вывести	Output
первый элемент	10
элемент с номером 3	11
последний элемент	13
предпоследний элемент	8
первые 6 элементов	10 7 -6 11 13 5
весь список, кроме последних 3 элементов	10 7 -6 11 13 5
все элементы с четными номерами (считая, что индексация начинается с 0);	10 -6 13 1 13
все элементы с нечетными номерами	7 11 5 8
все элементы в обратном порядке	13 8 1 5 13 11 -6 7 10
все элементы в обратном порядке, начиная с предпоследнего	8 1 5 13 11 -6 7 10
все элементы списка через один в обратном порядке, начиная с предпоследнего;	8 5 11 7
длину списка	9

### 3 - Вставить между буквами точки

Получите новую строку, вставив между двумя символами исходной строки точки. Выведите полученную строку. Пример:

Input	Output
python	p.y.t.h.o.n

### 5 - Найдите минимальное и максимальное число

Дана последовательность целых чисел (на одной строке). Напечатайте минимальное и максимальное число.

Input	Output
7 19 -3 8 -11 0 56	-11 56

### 5a - Найдите минимальное и максимальное число

Дана последовательность целых чисел (на многих строках). Напечатайте минимальное и максимальное число.

Input	Output
7 19 -3 8 -11 0 56	-11 56

Рекомендуем из полученных списков чисел для строки сделать один общий список всех чисел.

### 5b - Найдите минимальное и максимальное число

Дана последовательность целых чисел (на многих строках). Напечатайте минимальное и максимальное число.

Input	Output
7 19 -3 8 -11 0 56	-11 56

Надо экономить память.

Не делайте общий список всех чисел. Сделайте список минимальных в строках чисел и найдите в нем минимум.

Сделайте список максимальных= в строках чисел и найдите в нем максимум.

### 6 - Отсортируйте числа

Дана последовательность целых чисел (на одной строке). Напечатайте ее по возрастанию.

Input	Output
7 19 -3 8 -11 0 56	-11 -3 0 7 8 19 56

### 7 - Оценки

Студент за семестр получил оценки. Отбросьте 2 самых плохих оценки и посчитайте его средний балл.

### Еще задачи

#### Задача arr\_00 дважды напечатать

Дан массив. Напечатайте его два раза

Input	Output
7 19 -3 8 -11 0 56	7 19 -3 8 -11 0 56 7 19 -3 8 -11 0 56

#### Задача arr\_01 сначала четные потом нечетные

#### Задача arr\_1 в обратном порядке

#### Задача - все 0 заменить на 100

#### Задача - последний 0 заменить на 100

#### Задача - первый 0 заменить на 100

#### Задача arr\_02 индекс числа

#### Задача arr\_2 индекс числа или -1

#### Задача arr\_21 индекс последнего найденного

#### Задача arr\_22 индекс первого найденного

#### Задача arr\_31 поменять парами местами (четное количество чисел)

Пример меняем первый и следующий.

#### Задача arr\_32 поменять парами местами (Нечетное количество чисел)

## Задача agg\_33 поменять местами первые и последние (четное количество чисел)

Пример меняем первый и последний.

## Задача agg\_34 поменять местами первые и последние (НЕчетное количество чисел)

Задача agg\_41 реверсируем первые K чисел списка (из питона контекста), K меньше длины списка

Задача agg\_42 реверсируем первые K чисел списка (из питона контекста), K может быть больше длины списка

Задача сдвиг на  $k < \text{len}(a)$

Задача сдвиг на k любое положительное

Задача сдвиг на k любое положительное или отрицательное  
прочие задачи

### Задача 1

Дан массив. Напечатайте из него сначала отрицательные числа, а потом положительные и 0.

Input	Output
7 19 -3 8 -11 0 56	-3 -11 7 19 8 0 56

### Задача 2

Дан массив. Найдите сумму четных ЧИСЕЛ.

Input	Output
7 19 -3 8 -11 0 56	64

$$8 + 0 + 56 = 64$$

### Задача 3

Дан массив. Найдите сумму чисел на местах с четными НОМЕРАМИ (номера идут с 0)

Input	Output
7 19 -3 8 -11 0 56	49

$$7 + -3 + -11 + 56 = 49$$

### Задача 4

Дан массив. Первое четное число заменить на 100.

Input	Output
7 19 -3 8 -11 0 56	7 19 -3 100 -11 0 56

### Задача 5

Дан массив. Последнее четное число заменить на 100.

Input	Output
-------	--------

7 19 -3 8 -11 0 56 1	7 19 -3 8 -11 0 100 1
----------------------	-----------------------

### Задача 6

Дан массив. Поменять местами первое четное и последнее нечетное. В массиве есть и четные и нечетные числа

Input	Output
7 19 -3 8 -11 0 56	7 19 -3 -11 8 0 56

### Задача 7

Дан массив. Поменять местами первое четное и последнее нечетное. Если в массиве только четные или только нечетные, то ничего не менять.

Input	Output
7 19 -3 8 -11 0 56	7 19 -3 -11 8 0 56
2 -4 8 6	2 -4 8 6

### Задача 8. Шифр Цезаря - сделать алфавит

Для шифра Цезаря алфавит получается из старого сдвигом на N позиций

Так при сдвиге на 3 получаем новый алфавит

abcdefghijklmnopqrstuvwxyz

defghijklmnopqrstuvwxyzabc

И фраза "i have a dog." получается "l kdyh d grj."