

Тема 4

Python 2025

Списки. Цикл for



Списки = контейнеры

```
my_list = [100, 'Python', 25.1648489, False, [5, True]]
```



Индексируем:

```
my_list = [100, 'Python', 25.1648489, False, [5, True]]
```

0

1

2

3

4

Индексируем:

```
my_list = [100, 'Python', 25.1648489, False, [5, True]]
```

0

1

2

3

4

my_list[4]

->

[5, True]

my_list[4][0]

->

5

Индексируем:

```
my_list = [100, 'Python', 25.1648489, False, [5, True, 'text']]
```

my_list: как добраться до 'e' из 'text'?

Индексируем:

```
my_list = [100, 'Python', 25.1648489, False, [5, True, 'text']]
```

```
my_list[4] -> [5, True, 'text']
```

```
my_list[4][2] -> 'text'
```

```
my_list[4][2][1] -> 'e'
```

Индексируем:

```
my_list = [100, 'Python', 25.1648489, False, [5, True, 'text']]
```

```
my_list[-2:] -> [False, [5, True, 'text']]
```

```
my_list[1::2] -> ['Python', False]
```

все, что состоит из более
простых элементов,
называется *итерлируемым*
объектом

итерлируемый =
перебираемый
поэлементно

Итерируются:

- строки 'hello'
- списки [100, 46.97935]
- кортежи (100, 46.97935)
- другие контейнеры (*потом*)

**Кортежи отличаются от
списков только тем, что их
нельзя изменить**

**Их назначение - хранение (и
защита от наших случайных
ошибок)**

Списки изменить **МОЖНО**:

```
my_list = [100, 'текст']
```

```
my_list[0] = 'изменено' # присвоение
```

```
my_list -> ['изменено', 'текст']
```

**Списки - для
изменений**

**Кортежи - только
хранение**

новый прекрасный **цикл for**

- не уйдет в бесконечность
- не надо отсчитывать шаги*
- сам остановится
- работает только с *итерируемыми* объектами (*строки, списки, кортежи + иные контейнеры*)

МИНИМАЛИСТИЧНЫЙ СИНТАКСИС:

```
for i in 'hello' :
```

```
    print(i)
```

*# на каждом шаге
происходит следующее:
 $i = \text{строка}[0]$,
затем $i = \text{строка}[1]$ и т.д.*

МИНИМАЛИСТИЧНЫЙ СИНТАКСИС:

```
for i in [10, 20, 30] :
```

```
    print(i)
```

*# на каждом шаге
происходит следующее:*

i = 10,

затем i = 20 и т.д.

сложное:

```
c = [1, 16541, 35164, 66.44538]
```

```
for i in c:
```

```
    print(100000 + i) # печатаем результат  
                        сложения i и 100000
```

```
print(c)
```

изменится ли список c?

сложное:

```
c = [1, 16541, 35164, 66.44538]
```

```
for i in c:
```

```
    print(100000 + i)
```

```
print(c)
```

*# мы не меняем исходный
список, складываем только
для печати == показа на
экране*

изменится ли список c? **НЕТ**

Как менять:

$c = [1, 16541, 35164, 66.44538]$

$c[0] = c[0] + 100000$ # справа налево

$c[1] = c[1] + 100000$

$c[2] = c[2] + 100000$

$c[3] = c[3] + 100000$

Как менять **нормально**:

$c = [1, 16541, 35164, 66.44538]$

~~$c[0] = c[0] + 100000$~~

~~$c[1] = c[1] + 100000$~~

~~$c[2] = c[2] + 100000$~~

~~$c[3] = c[3] + 100000$~~

- перебрать циклом *for*

индексы от **0 до 4**

- обращаться поиндексно $c[i]$

- $c[i] += 1000000$

Как менять **нормально**:

```
c = [1, 16541, 35164, 66.44538]
```

```
for i in range(0, 4):  
    c[i] += 100000
```

Как менять **нормально**:

c = [1, 16541, 35164, 66.44538]

for i in range(0, 4):

c[i] += 100000

*Еще чуть-чуть
усложним: откуда мы
знаем, что длина
списка c - 4?*

Как менять **нормально**:

```
c = [1, 16541, 35164, 66.44538]
```

```
for i in range(0, len(c)):
    c[i] += 100000
```

а еще 0 можно опустить, как в срезе:

```
# range(len(c))
```

range - это генератор числовых последовательностей

range(начало, конец, шаг)

сравните со срезом:

my_list[начало:конец:шаг]

range(10) # от 0 по умолчанию до 9

range(1, 10) # от 1 до 9

range(10, 50) # от 10 до 49

range(0, 100, 5) # 0, 5, 10, 15... 95

range(len(список)) *# сгенерируй последовательность
от 0 до длины списка
== сгенерируй индексы*

объединим:

```
for i in range(len(c)): # i = 0, 1, 2, 3  
    print(c[i], 'печатаем элемент')  
    print(i, 'печатаем его индекс')
```

Перебрать список:

```
for i in список:  
    print(i)
```

Перебрать индексы:

```
for i in range(len(список)):  
    print(i) # i - 0, 1, 2....
```

задачи:

- ИЗМЕНИТЬ список
поиндексно (редко)
- перебрать **ДВА** списка
разом

```
books = ['название1',  
         'название2',  
         'название3',  
         'название4',  
         'название5']
```

```
authors = ['автор1',  
           'автор2',  
           'автор3',  
           'автор4',  
           'автор5']
```

Цикл for может перебрать
ИЛИ названия, **ИЛИ** авторов

А задание - напечатайте строку вида:

"автор1 написал книгу название1"

```
for book in books:  
    # откуда автора брать?
```

```
for author in authors:  
    # откуда название брать?
```

У них есть **ОБЩЕЕ** - индексы

```
books = ['название1',  
        'название2',  
        'название3',  
        'название4',  
        'название5']
```

0
1
2
3
4

```
authors = ['автор1',  
           'автор2',  
           'автор3',  
           'автор4',  
           'автор5']
```

```
for i in range(len(authors)): # или books?  
    print( authors[i] )  
    print( books[i] )  
    print( f'{authors[i]} написал книгу {books[i]}' )
```



*дай человеку **if**, списки и цикл **for** -
и он запрограммирует вообще
всё*