

# Цикл For

# План урока

---

- Понятие цикла **for**
- Значимость цикла **for**

# Цикл **for**

---

Цикл **for** служит для перебора элементов коллекций и последовательностей, для которых реализован итератор (**Iterable**)

---

Цикл **for** автоматически перехватывает исключение **StopIteration**, завершая при этом цикл

---

Операторы **break** и **continue** работают по тому же принципу, что и в **while**

# Цикл `for`

## Конструкция блока `for`

```
ooo  
for <element> in <iterable>:  
    <code block>  
else:  
    <code block else>
```

Локальная переменная,  
которая будет пробегать  
по всем элементам нашей  
последовательности  
(`Iterable` объекту)

# Цикл `for`

## Конструкция блока `for`

```
...  
for <element> in <iterable>:  
    <code block>  
else:  
    <code block else>
```

Код, который будет  
повторяться

# Цикл `for`

## Конструкция блока `for`

```
...
for <element> in <iterable>:
    <code block>
else: ←
    <code block else>
```

Блок `else`,  
является  
опциональным

Блок `else` необязательный  
Для `else` логика та же, что и в `while`

# Функция range

Функция `range` – это функция, которая возвращает арифметическую прогрессию в виде итерируемого объекта

ooo	код
<pre>for i in range(2, 11, 3): ←     print(i)</pre>	

ooo	вывод
<pre>&gt;&gt;&gt; 2 &gt;&gt;&gt; 5 &gt;&gt;&gt; 8</pre>	

Вызываем `range` с начальным элементом 2, шагом 3 и с элементами до 11 включительно

# Функция range

Функция `range` – это функция, которая возвращает арифметическую прогрессию в виде итерируемого объекта

ooo | код

```
for i in range(2, 11, 3):
    print(i)
```

ooo | вывод

```
>>> 2
>>> 5
>>> 8
```

i - это локальная переменная, собственно, Iterable object

# Функция range

Функция `range` – это функция, которая возвращает арифметическую прогрессию в виде итерируемого объекта

ооо	код
<pre>for i in range(2, 11, 3):     print(i) ←</pre>	

ооо	вывод
<pre>&gt;&gt;&gt; 2 &gt;&gt;&gt; 5 &gt;&gt;&gt; 8</pre>	

Вывод  
значений  
в консоль

# Пример

Вывести все числа от 1 до 100,  
для которых число 17 является делителем

ooo | код

```
for i in range(1, 101):  
    if i % 17 == 0:  
        print(i)
```

ooo | вывод

```
>>> 17  
>>> 34  
>>> 51  
>>> 68  
>>> 85
```

Цикл for  
перебирает  
значения  
от 1 до 101 (не  
включительно)

# Пример

Вывести все числа от 1 до 100,  
для которых число 17 является делителем

ooo | код

```
for i in range(1, 101):
    if i % 17 == 0: <-
        print(i)
```

ooo | вывод

```
>>> 17
>>> 34
>>> 51
>>> 68
>>> 85
```

Проверяем  
кратность 17

# Пример

Вывести все числа от 1 до 100,  
для которых число 17 является делителем

ooo | код

```
for i in range(1, 101):
    if i % 17 == 0:
        print(i) ←
```

ooo | вывод

```
>>> 17
>>> 34
>>> 51
>>> 68
>>> 85
```

Печатаем i

# Цикл for

## Вложенные циклы

```
ooo
```

```
for <elem 1> in <iterable>:  
    <code block>  
    for <elem 2> in <inner iterable>:  
        <inner code block>
```

Во внешнем цикле  
мы перебираем  
элементы из **iterable** и  
записываем их в локальную  
переменную **elem 1**

# Цикл `for`

## Вложенные циклы

ооо

```
for <elem 1> in <iterable>:  
    <code block>  
    for <elem 2> in <inner iterable>:  
        <inner code block>
```



Запускаем еще один цикл `for`,  
где с помощью локальной  
переменной `elem 2` пробегаем  
по коллекции `inner iterable`

# Цикл `for`

## Вложенные циклы

```
...
for <elem 1> in <iterable>:
    <code block>
    for <elem 2> in <inner iterable>:
        <inner code block>
```

Оператор **break**, вызванный во внутреннем цикле, прервет только внутренний цикл

Оператор **continue**, вызванный во внутреннем цикле, прерывает итерацию только внутреннего цикла

# Пример

Вывести в консоль таблицу  
умножения размером  $n \times n$

ооо	ввод
<pre>n = int(input())  i = 1 while i &lt;= n:     j = 1     while j &lt;= n:         print(f"{i}x{j}={i*j}", end=" ")         j += 1     print("")     i += 1</pre>	

# Пример

Вывести в консоль таблицу  
умножения размером  $n \times n$

ооо	ввод
<pre>n = int(input()) ← for i in range(1, n + 1):     for j in range(1, n + 1):         print(f"{i}{x}{j}={i*j}", end=" ")     print("")</pre>	

Считываем с консоли  
переменную  $n$   
(целое число)

# Пример

Вывести в консоль таблицу умножения размером  $n \times n$

ооо	ввод
<pre>n = int(input()) for i in range(1, n + 1):&lt;     for j in range(1, n + 1):         print(f"\{i}\x0d\{j}\={i*j}", end=" ")     print("")</pre>	

Во внешнем цикле переменной  $i$  пробегаем значения от 1 до  $n$  включительно

# Пример

Вывести в консоль таблицу умножения размером  $n \times n$

ооо	ввод
<pre>n = int(input()) for i in range(1, n + 1):     for j in range(1, n + 1):←         print(f"{i}x{j}={i*j}", end=" ")     print("")</pre>	

Внутри запускаем цикл  
**for j in range(1, n+1)**

# Пример

Вывести в консоль таблицу умножения размером  $n \times n$

ооо	ввод
<pre>n = int(input()) for i in range(1, n + 1):     for j in range(1, n + 1):         print(f"\{i}\x0d\{j}\={i*j}", end=" ")     print("")</pre>	

Выводим элементы  
и в конце итерации внешнего  
цикла выводим символ  
перевода на другую строку

# Когда использовать `for`, а когда `while`?

---

**For** лучше использовать, когда нужно пройти по **заранее определенной** коллекции или последовательности с **известным** количеством элементов

---

**While** мы должны использовать тогда, когда мы **не знаем** заранее, когда мы **завершим** цикл

# Пример

Считывать целые числа из консоли  
и выводить квадраты этих чисел  
с новой строки до тех пор, пока  
не будет введено число 0

ooo | ввод

```
num = int(input())
while num != 0:
    print(num ** 2)
    num = int(input())
```

# Цикл for

Пример с использованием list

ooo | код

```
numbers = [4, 9, -3, 5]
for number in numbers:
    print(number)
else:
    print('end')
```

ooo | вывод

```
>>> 4
>>> 9
>>> -3
>>> 5
>>> end
```

Набор чисел  
(упорядоченная  
коллекция)

# Цикл for

## Пример с использованием list

ooo | код

```
numbers = [4, 9, -3, 5]
for number in numbers:
    print(number)
else:
    print('end')
```

ooo | вывод

```
>>> 4
>>> 9
>>> -3
>>> 5
>>> end
```

Проходим  
по элементам  
последовательности  
и выводим их

# Цикл for

Пример с использованием list

ooo | код

```
numbers = [4, 9, -3, 5]
for number in numbers:
    print(number)
else:
    print('end')
```

ooo | вывод

```
>>> 4
>>> 9
>>> -3
>>> 5
>>> end
```

В блоке else  
печатаем «end»

# Цикл for

Пример с использованием list

ooo | код

```
numbers = [4, 9, -3, 5]
for number in numbers:
    print(number)
else:
    print('end')
```

ooo | вывод

```
>>> 4
>>> 9
>>> -3
>>> 5
>>> end
```