



№ 4

# Циклы

# Цель

Познакомиться с циклами в Python: ***for, while***.

## Программа

1. Основные определения
2. Цикл while
3. Цикл for
4. Функция range()
5. Оператор continue
6. Оператор break
7. Использование else после циклов
8. Описание работы программы

# Основные определения

**Цикл** — конструкция языка программирования, предназначенная для многократного исполнения входящего в нее набора инструкций.

**Тело цикла** — набор выполняемых в цикле инструкций.

**Итерация** — единичное исполнение тела цикла.

**Условие выхода** — логическое выражение, истинность которого показывает, будет ли выполнена следующая итерация. Если условие ложно, то происходит выход из цикла.

# Основные определения

**Бесконечный цикл** — цикл, выполнение тела которого должно выполняться бесконечно (т.е. не завися от каких-либо условий).

**Цикл с предусловием** — цикл, в котором условие выхода проверяется перед выполнением итерации. Поэтому тело цикла может быть не выполнено ни разу (если с самого начала условие ложно).

**Цикл с параметром** — цикл, в котором указывается переменная и множество значений, которая будет принимать эта переменная.

# Цикл while

Реализацией цикла с предусловием является цикл **while** (англ. *while* - пока).

Синтаксис цикла **while**:

```
while <УСЛОВИЕ ВЫХОДА>:  
    <ТЕЛО ЦИКЛА>
```

Таким образом, выполняется <ТЕЛО ЦИКЛА> пока верно <УСЛОВИЕ ВЫХОДА>.

# Цикл while

## Бесконечный цикл

Используя `while`, можно написать бесконечный цикл:

```
while True:  
    <ТЕЛО ЦИКЛА>
```

# Цикл for



Реализацией цикла с параметром является цикл **for** (англ. *for* — для).

Синтаксис цикла **for**:

```
for <ПАРАМЕТР> in <МНОЖЕСТВО ЗНАЧЕНИЙ>:  
    <ТЕЛО ЦИКЛА>
```

Переменной <ПАРАМЕТР> присваивается значение первого элемента <МНОЖЕСТВА ЗНАЧЕНИЙ>, после чего выполняется <ТЕЛО ЦИКЛА>. Затем переменной <ПАРАМЕТР> присваивается следующее по порядку значение и так далее до тех пор, пока не будут перебраны все элементы <МНОЖЕСТВА ЗНАЧЕНИЙ>.

# Цикл for

## Параметр

В качестве <ПАРАМЕТРА> цикла необходимо указать имя переменной, которой будут присваиваться значения из <МНОЖЕСТВА ЗНАЧЕНИЙ>.

Например, назовем параметр цикла «i»:

```
for i in <МНОЖЕСТВО ЗНАЧЕНИЙ>:  
    <ТЕЛО ЦИКЛА>
```



# Цикл for

## Множество значений

<МНОЖЕСТВО ЗНАЧЕНИЙ> может быть задано произвольной последовательностью.

Например:

```
for i in 1, 2, 3:  
    <ТЕЛО ЦИКЛА>
```

Такой цикл выполнится 3 раза, переменной «i» будут последовательно присвоены значения 1, 2 и 3.

# Цикл for

## Множество значений

Во <МНОЖЕСТВО ЗНАЧЕНИЙ> могут быть выражения различных типов.

Например:

```
for i in 1, 2, 3, "one", "two", "three":  
    <ТЕЛО ЦИКЛА>
```

При первых трех итерациях цикла переменная «i» будет принимать значение типа int, при последующих трех — типа str.

# Цикл for

## Функция range()

Синтаксис функции range():

```
range(<СТАРТ>, <СТОП>, <ШАГ>)
```

Функция range() имеет следующие аргументы:

- <СТАРТ> — указывает целое число, с которого начинается последовательность (включительно, 0 по умолчанию).
- <СТОП> — указывает последнее целое число последовательности (исключительно, нужно указывать обязательно).
- <ШАГ> — определяет шаг: на сколько нужно увеличивать параметр при каждой итерации (1 по умолчанию).

# Цикл for

## Функция range()

Синтаксис функции range() с одним аргументом:

```
range(<СТОП>)
```

Пример использования в цикле:

```
for <ПАРАМЕТР> in range(N):  
    <ТЕЛО ЦИКЛА>
```

<ТЕЛО ЦИКЛА> выполнится N раз от 0 до N-1 с шагом 1.

# Цикл for

## Функция range()

Синтаксис функции range() с двумя аргументами:

```
range(<СТАРТ>, <СТОП>)
```

Пример использования в цикле:

```
for <ПАРАМЕТР> in range(N, M):  
    <ТЕЛО ЦИКЛА>
```

<ТЕЛО ЦИКЛА> выполнится M-N раз от N до M-1 с шагом 1.

# Цикл for

## Функция range()

Синтаксис функции range() с тремя аргументами:

```
range(<СТАРТ>, <СТОП>, <ШАГ>)
```

Пример использования в цикле:

```
for <ПАРАМЕТР> in range(N, M, K):  
    <ТЕЛО ЦИКЛА>
```

<ТЕЛО ЦИКЛА> выполнится  $(M-N)/K$  раз от N до M-1 с шагом K.

# Оператор continue

Оператор **continue** (с англ. продолжать) начинает следующий проход цикла, минуя оставшееся тело цикла.

Пример использования оператора continue с циклом for:

```
for <ПАРАМЕТР> in <МНОЖЕСТВО ЗНАЧЕНИЙ>:  
    if <УСЛОВИЕ>:  
        continue  
    <ТЕЛО ЦИКЛА>
```

Таким образом, если <УСЛОВИЕ> верно, то оставшееся <ТЕЛО ЦИКЛА> выполняться не будет. После оператора continue сразу начнется новая итерация цикла.

# Оператор break

Оператор **break** (с англ. *разрыв*) досрочно прерывает цикл.

Пример использования оператора **break** с циклом **while**:

```
while <УСЛОВИЕ ВЫХОДА>:  
    if <УСЛОВИЕ>:  
        break  
    <ТЕЛО ЦИКЛА>
```

Таким образом, если <УСЛОВИЕ> верно, то выполнения данного цикла будет закончено.



## Else после циклов

После <ТЕЛА ЦИКЛА> можно написать слово **else** и после него <БЛОК ИНСТРУКЦИЙ>. Он будет выполнен один раз после окончания цикла, когда проверяемое условие станет неверно.

Использования **else** после цикла **while**:

```
while <УСЛОВИЕ ВЫХОДА>:  
    <ТЕЛО ЦИКЛА>  
else:  
    <БЛОК ИНСТРУКЦИЙ>
```

# Else после циклов

Использование **else** после цикла **for**:

```
for <ПАРАМЕТР> in <МНОЖЕСТВО ЗНАЧЕНИЙ>:  
    <ТЕЛО ЦИКЛА>  
else:  
    <БЛОК ИНСТРУКЦИЙ>
```

# Описание работы программы

## Лесенка

По данному натуральному числу  $n$ , введенному с клавиатуры ( $1 \leq n \leq 9$ ), программа должна вывести лесенку из  $n$  ступенек.  $i$ -я ступенька состоит из чисел от  $1$  до  $i$  без пробелов.

Введите число: 5

1

12

123

1234

12345

# Описание работы программы

## Наименьший натуральный делитель

Написать программу, выводящую наименьший натуральный делитель (не равный 1), введенного числа.

Введите число: 55

Наименьший натуральный делитель: 5

# Описание работы программы

## Факториал

Написать программу, выводящую  $n!$  (произведение чисел от **1** до **n**), **n** вводится с клавиатуры.

Введите n: 4

$4! = 24$

# Описание работы программы

## Числа Фибоначчи

Последовательность чисел Фибоначчи:

*0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...*

Каждый член последовательности равен сумме двух предыдущих членов.

Написать программу, которая по заданному числу **n** выводит **n**-ый член последовательности Фибоначчи (**n** > **1**).

Введите число: 8

8 член последовательности Фибоначчи - 13

# Описание работы программы

## Степень двойки

Написать программу, которая проверяет, является ли введенное число степенью двойки.

Введите число: 32

32 является степенью двойки