

Циклы

Цель

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

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.

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, которое будет принимать эта переменная.

Реализацией цикла с предусловием является цикл *while* (англ. *while* — пока). Синтаксис цикла *while*:

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

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

Цикл *while* можно назвать самым универсальным циклом в Python, но он выполняется медленнее, чем цикл с параметром. Еще один минус цикла *while* состоит в том, что если <УСЛОВИЕ ВЫХОДА> задано неверно, то цикл может стать бесконечным.

Иногда бесконечный цикл нужно организовать специально. Это можно сделать, используя цикл *while*:

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

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* выполняется быстрее цикла *while*. Это связано с тем, что в нём нет логических проверок.

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>
```

<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.

Во <МНОЖЕСТВЕ 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*.

Функция *range()*

Для того, чтобы указать сколько раз нужно повторить тело цикла, можно использовать функцию *range()*. Синтаксис функции *range()*:

```
range(<СТАРТ>, <СТОП>, <ШАГ>)
```

Функция *range()* имеет такие аргументы:

- <СТАРТ> указывает целое число, с которого начинается последовательность (включительно). Если этот аргумент не указан, по умолчанию используется 0.
- <СТОП> нужно использовать всегда. Это последнее целое число последовательности (исключительно).
- <ШАГ> определяет шаг: насколько нужно увеличивать (в случае с отрицательными числами уменьшать) параметр при каждой итерации. Если аргумент <ШАГ> пропущен, по умолчанию используется 1.

<ШАГ> и <СТАРТ> указывать не обязательно.

Синтаксис функции *range()* с одним аргументом:

```
range(<СТОП>)
```

Если значение аргумента <СТОП> равно нулю или отрицательно, то тело цикла не выполнится ни разу.

Пример использования в цикле:

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

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

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 .

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 .

Операторы *break* и *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* (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. Помните, что злоупотребление операторами *break* и *continue* в программировании не поощряется.

Else после циклов

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

Описание работы программы

№1 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
```

№2 Наименьший натуральный делитель

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

```
Ведите число: 55
```

```
Наименьший натуральный делитель: 5
```

№3 Факториал

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

```
Введите n: 4
```

```
4! = 24
```

№4 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
```

№5 Степень двойки

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

Введите число: 32

32 является степенью двойки