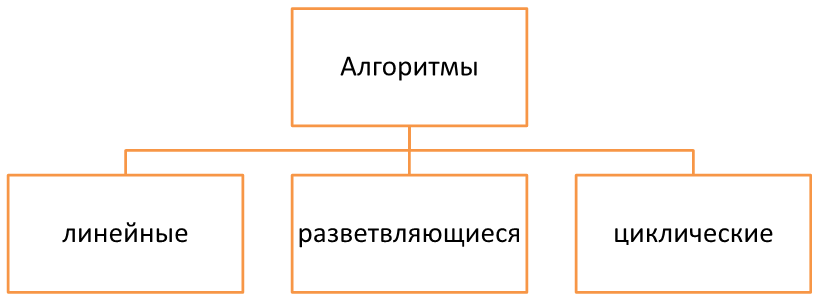
**Циклические конструкции**

**Основные алгоритмические конструкции**

****

* **Линейный алгоритм** — набор команд (указаний), выполняемых последовательно во времени друг за другом.
* **Разветвляющийся алгоритм** — алгоритм, содержащий хотя бы одно условие, в результате проверки которого может осуществляться разделение на несколько параллельных ветвей алгоритма.
* **Циклический алгоритм** — алгоритм, предусматривающий многократное повторение одного и того же действия (одних и тех же операций) над новыми исходными данными.

К циклическим алгоритмам сводится большинство методов вычислений, перебора вариантов. Цикл программы — последовательность команд (серия, тело цикла), которая может выполняться многократно (для новых исходных данных) до удовлетворения некоторого условия.

Цикл – это управляющая конструкция, многократно исполняющая блок кода.

Существуют три классических разновидности циклов:

* **цикл с предусловием;**
* **цикл с постусловием;**
* **цикл со счётчиком.**

Также для современных языков программирования характерен **foreach-цикл** (совместный цикл), который выполняет определённые действия для всех элементов из какого-то набора.

Первые два варианта циклических конструкций выполняют участок кода, пока указанное условие истинно. Они отличаются тем, когда проверяется условие: в начале каждого шага (итерации) или в конце. Циклы с предусловием и с постусловием взаимозаменяемы и любой из них можно переписать с использованием другого, при этом (по мнению создателей языка **Python**) циклы с постусловием используются значительно реже и хуже читаются из-за места указания условия, поэтому в **Python** существуют только циклы с предусловием (циклы “**while**”).

Цикл со счётчиком (цикл “**for**”) используется для повторения определённых действий конкретное количество раз, при этом доступен счётчик – номер текущей итерации.

В **Python** цикл “**for**” является по своей сути совместным циклом, то есть цикл со счётчиком реализуется при помощи **foreach**-цикла и функции, которая возвращает последовательность целых чисел. Как на самом деле работает цикл “**for**” в **Python** мы изучим позже при изучении коллекций и последовательностей. Пока что мы будем рассматривать цикл “**for**” как цикл со счётчиком.

Цикл **while** в **Python** – цикл с предусловием. Синтаксис:

**while условие:**

**операторы**

Здесь:

***условие***–это логическое выражение,

***операторы*** – это последовательность каких-либо других команд.

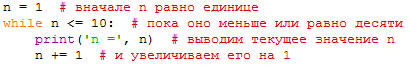
Блок операторов не может быть пустым. Если такая необходимость возникает, можно воспользоваться оператором ***pass***, который не делает ничего.

Блок операторов выполняется до тех пор, пока условие продолжения цикла истинно. Если оно изначально было ложным, то он не выполняется ни разу.

Эта конструкция похожа на оператор **if**, только выполняет блок операторов не один раз, а несколько, пока условие истинно. Все, что было сказано про условия, отступы и вложенные операторы справедливо и для оператора **while** (а также **for**, который будет рассмотрен позже).

Условие должно зависеть от какого-либо состояния, которое изменяется, иначе цикл будет выполняться бесконечно (пока пользователь не прервёт выполнение программы), если оно изначально было истинным.

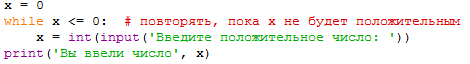
Пример 1



Результат



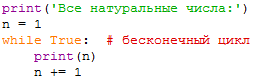
Пример 2



Результат



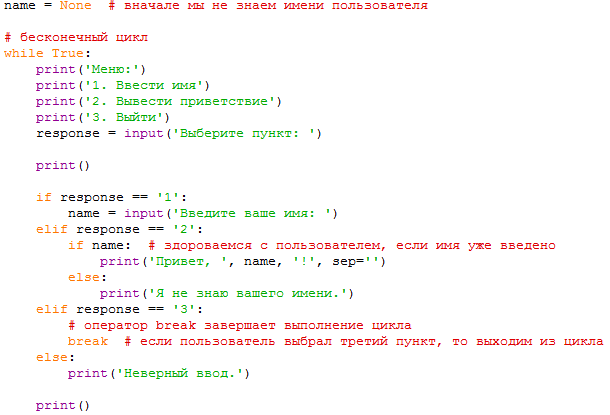
Если необходимо создать бесконечный цикл, в качестве условия принято использовать **True**:



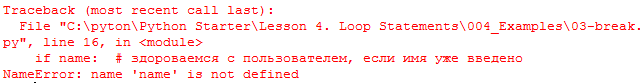
Чтобы прервать бесконечный цикл, следует нажать комбинацию клавиш **<Ctrl>+<C>**. В результате генерируется исключение **Keyboardinterrupt**, и выполнение программы будет остановлено. Следует учитывать, что прервать таким образом можно только цикл, который выводит данные.

Если необходимо досрочно завершить выполнение цикла, то можно прервать его при помощи оператора **break**.

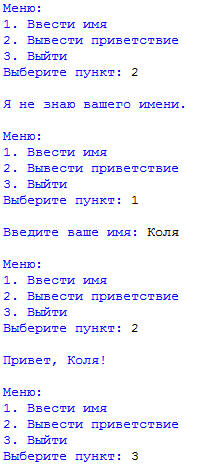
Пример 3



Обратите внимание на первый оператор. Если вы не определите значение None (отсутствие значения) для переменной name (имени), то если при первом запуске ввести 2, то выведется сообщение об ошибке.



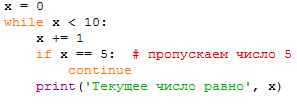
Результат работы программы:



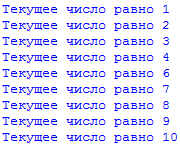
Один проход цикла через блок операторов называется **итерацией**.

Если необходимо прервать выполнение текущей итерации и перейти к началу следующей, то используется оператор **continue**.

Пример 4

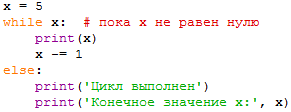


Результат выполнения

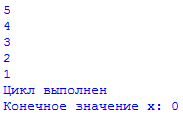


Оператор **while** также может иметь ветку **else** (по аналогии с **if**). Вначале каждой итерации интерпретатор проверяет истинность условия выполнения цикла, и если оно истинно, то исполняет ветку **while**, иначе он исполняет ветку **else** (если она присутствует) и завершает выполнение цикла, причём это может произойти и перед первой итерацией, если условие изначально было ложным. Однако если цикл был прерван оператором **break**, то ветка **else** не исполняется.

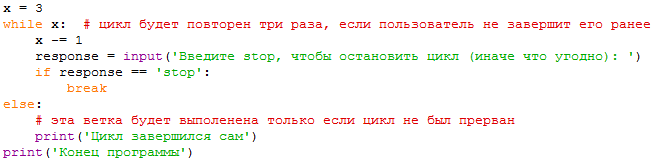
Пример 5



Результат выполнения



Пример 6

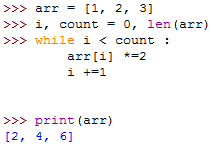


**Последовательность работы цикла while:**

1. Переменной - счетчику присваивается начальное значение.
2. Проверяется условие, если оно истинно, выполняются инструкции внутри цикла, иначе выполнение цикла завершается.
3. Переменная-счетчик изменяется на величину, указанную в параметре <Приращение>.
4. Переход к пункту 2.
5. Если внутри цикла не использовался оператор **break**, то после завершения выполнения цикла будет выполнен блок в инструкции **else**. Данный блок не является обязательным.

С помощью цикла **while** можно перебирать и элементы различных структур. Но в этом случае следует помнить, что цикл **while** работает медленнее цикла **for**. В качестве примера умножим каждый элемент списка на 2.

Пример 7



Цикл со счётчиком (цикл **for**) – это цикл, в котором переменная – счётчик итераций изменяет своё значение от начального до конечного с определённым шагом.

Цикл **for** со счётчиком в **Python**:

**for переменная in range(конечное\_значение):**

**операторы**

или

**for переменная in range(начальное\_значение, конечное\_значение):**

**операторы**

или

**for переменная in range(начальное\_значение, конечное\_значение, шаг):**

**операторы**

В первом варианте переменная-счётчик принимает значение от нуля включительно до заданного значения, не включая его; во втором задаётся начальное значение вместо нуля; в третьем указывается ещё и шаг изменения переменной-счётчика (по умолчанию 1).

Пример 8



Результат выполнения:



Пример 9



Результат выполнения:



Пример 10



Результат выполнения:

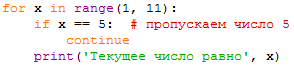


Можно задать начальное значение больше, чем конечное, и шаг, равный -1, и получить обратный цикл.

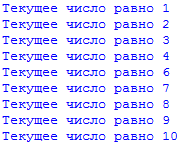
Перед началом каждой итерации переменная-счётчик принимает следующее значение из сгенерированной интерпретатором последовательности. Таким образом, любые изменения переменной-счётчика будут потеряны на следующей итерации. После завершения цикла переменная-счётчик сохраняется и равна последнему значению, которое она принимала.

Операторы **break** и **continue** в цикле **for** имеют такой же смысл, как и в цикле **while**.

Пример 11

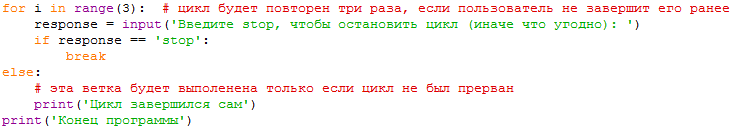


Результат выполнения:



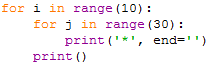
Аналогично циклу **while**, цикл **for** также может иметь ветку **else**.

Пример 12

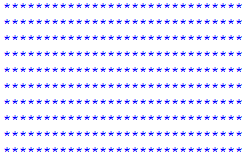


Вложенные циклы – это циклы, находящиеся внутри других циклов. Цикл, который вложен в тело другого, называется внутренним циклом. Цикл, в тело которого вложен другой цикл, называется внешним.

Пример 13



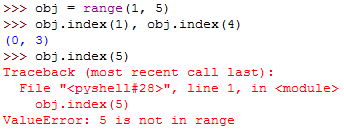
Результат выполнения:



Начиная с версии **Python 3.2**, объект **range** поддерживает два метода:

* index (<Значение>) - возвращает индекс элемента, имеющего указанное значение. Если значение не входит в объект, возбуждается исключение **Va1ueError**.

Пример 14



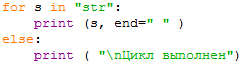
* count (<Значение>) - возвращает количество элементов с указанным значением. Если элемент не входит в объект, то возвращается значение 0.

Пример 15



В цикле **for** вместо функции **range()** можно использовать любую последовательность (строку, список или кортеж).

Пример 16. Перебор букв в слове.



Результат выполнения:



Пример 17. Вывод каждого элемента списка и кортежа на отдельной строке.

