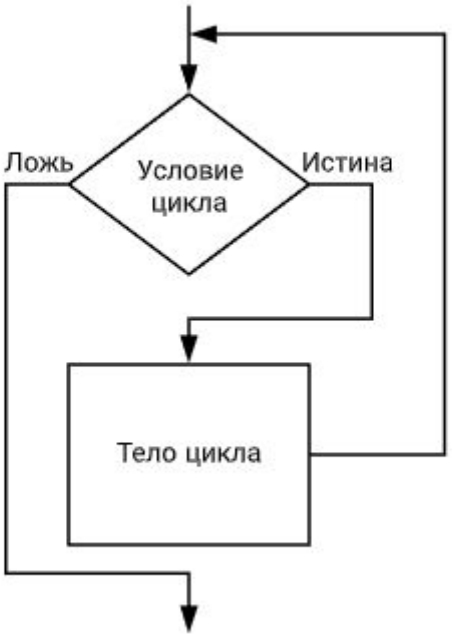
Цикл с предусловием

Цикл с предусловием это такой цикл, в котором условие продолжения проверяется перед повторяемыми действиями:



В этом случае тело цикла повторяется **пока** условие истинно. В случае, если условие изначально ложное, тело цикла не выполнится ни разу.

В Python данный вид цикла представлен циклом **while** (на английском означает “пока”).

```
while условие_цикла:
    тело цикла
```

Здесь условие цикла может записываться также, как мы бы записывали условие оператора **if**, только в отличие от if, тело может повторяться столько раз, пока условие истинно.

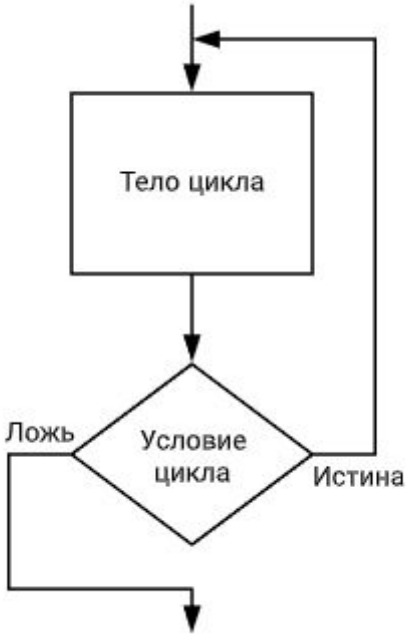
Стоит помнить, что в отличие от цикла for, при написании цикла while заранее может быть неизвестно сколько раз нужно, чтобы повторилось тело цикла. Из-за этого возможна ошибка, при которой программа никогда не выходит из цикла while, т.е. зависает.

Чтобы этого не происходило, нужно обеспечить циклу while возможность сделать условие ложным в процессе выполнения, чтобы закончить цикл:

```
a = 5
while a > 0: # чтобы завершиться, надо уменьшать a до 0
    # какие-то действия
    a -= 1 # уменьшаем, чтобы когда-нибудь закончить цикл
```

Цикл с постусловием

Цикл с постусловием это такой цикл, в котором сначала выполняется повторяемое действие и только затем проверяется условие продолжения:



В этом случае также действие продолжается пока условие истинно, однако из-за того, что первая проверка условия произойдет только после первого выполнения тела, оно выполнится хотя бы один раз даже если условие изначально ложно.

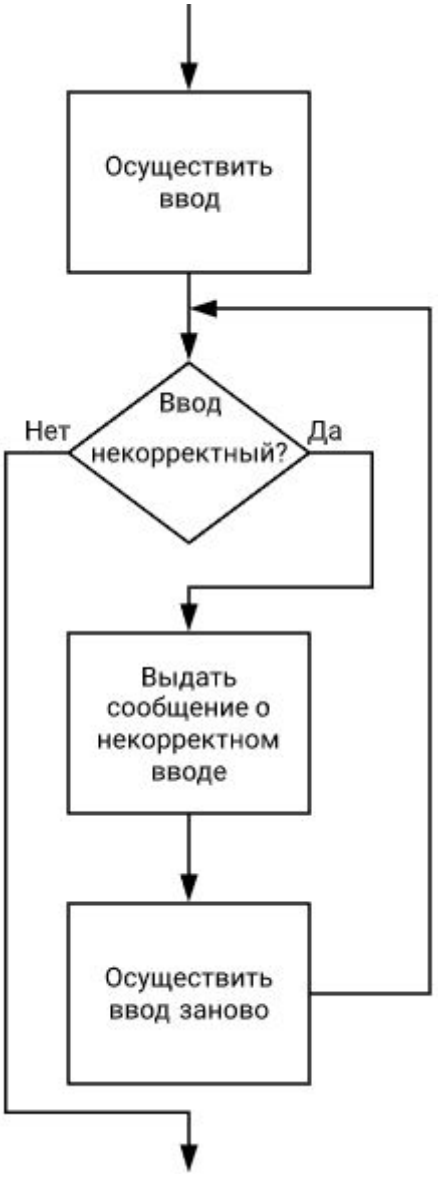
Из-за ограниченного применения цикла с постусловием, а также того, что он может быть заменен на цикл с предусловием в большинстве задач, цикл с постусловием отсутствует в Python.

Проверка корректности ввода

Очень часто в программе требуется, чтобы пользователь ввел не просто какое-то значение, а чтобы это значение удовлетворяло каким-то условиям (например, было положительным). Пользователь не всегда будет читать ваши рекомендации по вводу, а даже если прочитает, не всегда его квалификации может быть достаточно, чтобы понять их, поэтому пользователь может ввести все, что угодно. Так как наша программа может работать только с определенными значениями, все что угодно мы принять не можем, и в случае если ввод пользователя нас не удовлетворил, мы должны сообщить пользователю об этом и запросить ввод заново.

Если решать эту задачу с помощью **if**, пользователь может повторно ввести некорректное значение (по невнимательности, незнанию или злому умыслу). Известен, в частности, баг экрана входа в операционную систему Mac OS, когда в случае двух вводов подряд пустого пароля для пользователя root (суперадминистратор компьютера), пользователя пускали в систему с правами root.

Чтобы такого не происходило, неверный ввод стоит обрабатывать при помощи цикла до тех пор, пока он не окажется верным или пользователь не закроет программу:



Так, если например нам нужно, чтобы пользователь мог ввести только положительные значения, можно воспользоваться следующим кодом:

```
x = int(input("Введите положительное число: "))
while x <= 0: # пока ввод некорректный
    print("Нельзя вводить отрицательные числа или ноль!")
    x = int(input("Введите положительное число: "))
```

Операторы управления циклом

Для любых видов циклов (как for, так и while) поддерживаются операторы управления циклом.

Оператор **continue** позволяет не выполнять остаток тела цикла и перейти на следующую итерацию. В цикле while это вызовет проверку условия. Если условие осталось истинным, цикл продолжит повторение. Если условие оказалось ложным, цикл завершится. В цикле for переменная-счетчик получит свое следующее значение и будет выполнена проверка на окончание повторения как обычно.

```
for i in range(10):
    if i % 2 == 0: # четные пропустим
        continue
    # другие действия
```

Оператор **break** прерывает выполнение цикла.

```
for i in range(10):
    if i > 5: # на самом деле только до пяти
        break
    # другие действия
```

Из-за того, что **break** позволяет завершать цикл отличным от прописанного в определении цикла способом, это усложняет чтение кода программистом. Если есть возможность написать цикл без использования **break**, стоит им воспользоваться.