Урок Вложенные циклы

Вложенные циклы

Аннотация

В этом уроке мы рассмотрим вложенные циклы, позволяющие запустить цикл внутри циклического оператора. Приведем несколько примеров вложенности разных циклов, а также применение операторов break и continue со вложенными циклами.

1. Вложенные циклы. Принцип работы

Часто бывают ситуации, когда один и тот же набор действий необходимо выполнить несколько раз для каждого повторяющегося действия. Например, мы уже несколько раз сталкивались с задачами, когда программа получает от пользователя данные до сигнала остановки, — для этого используется цикл. А теперь представьте, что после ввода данных или числа с ними надо сделать какие-либо действия, которые тоже требуют цикла (например, вычислить факториал), тогда нам нужен еще один цикл, внутри первого.

Вложенные циклы

Циклы называются вложенными (т. е. один цикл находится внутри другого), если внутри одного цикла во время каждой итерации необходимо выполнить другой цикл. Так для каждого витка внешнего цикла выполняются все витки внутреннего цикла. Основное требование для таких циклов: чтобы **все** действия вложенного цикла располагались внутри внешнего.

При использовании вложенных циклов стоит помнить, что изменения, внесенные внутренним циклом в какие-либо данные, могут повлиять на внешний.

Давайте рассмотрим следующую задачу: необходимо вывести в строку таблицу умножения для заданного числа. Задача решается так:

```
k = int(input())
for i in range(1, 10):
    print(i, '*', k, '=', k * i, sep='', end='\t')
```

А если нам нужно вывести таблицу умножения для всех чисел от 1 до k?

Очевидно, что в этом случае предыдущую программу нужно повторить k раз, где вместо k будут использоваться числа от 1 до k включительно.

Эту задачу можно записать двумя циклами, где для каждого значения внешнего цикла будут выполняться все значения внутреннего цикла.

Программа будет выглядеть так:

```
k = int(input())
for j in range(1, k + 1):
    for i in range(1, 10):
        print(i, '*', j, '=', j * i, sep='', end='\t')
    print()
```

Проанализируем работу данной программы. Выполнение программы начинается с внешнего цикла. Итератор `j` внешнего цикла `for` меняет свое значение от начального (`1`) до конечного (`k`). Обратите внимание: чтобы включить число k в рассматриваемый диапазон, в заголовке цикла указавается промежуток от `1` до `k + 1`. Затем циклически выполняется следующее:

1. Проверяется условие `j < k + 1`.
2. Если оно соблюдается, выполняется оператор в теле цикла, т. е. выполняется внутренний цикл.
   - Итератор `i` внутреннего цикла `for` будет изменять свои значения от начального (1) до конечного (10), не включая 10

   Затем циклически выполняется следующее:

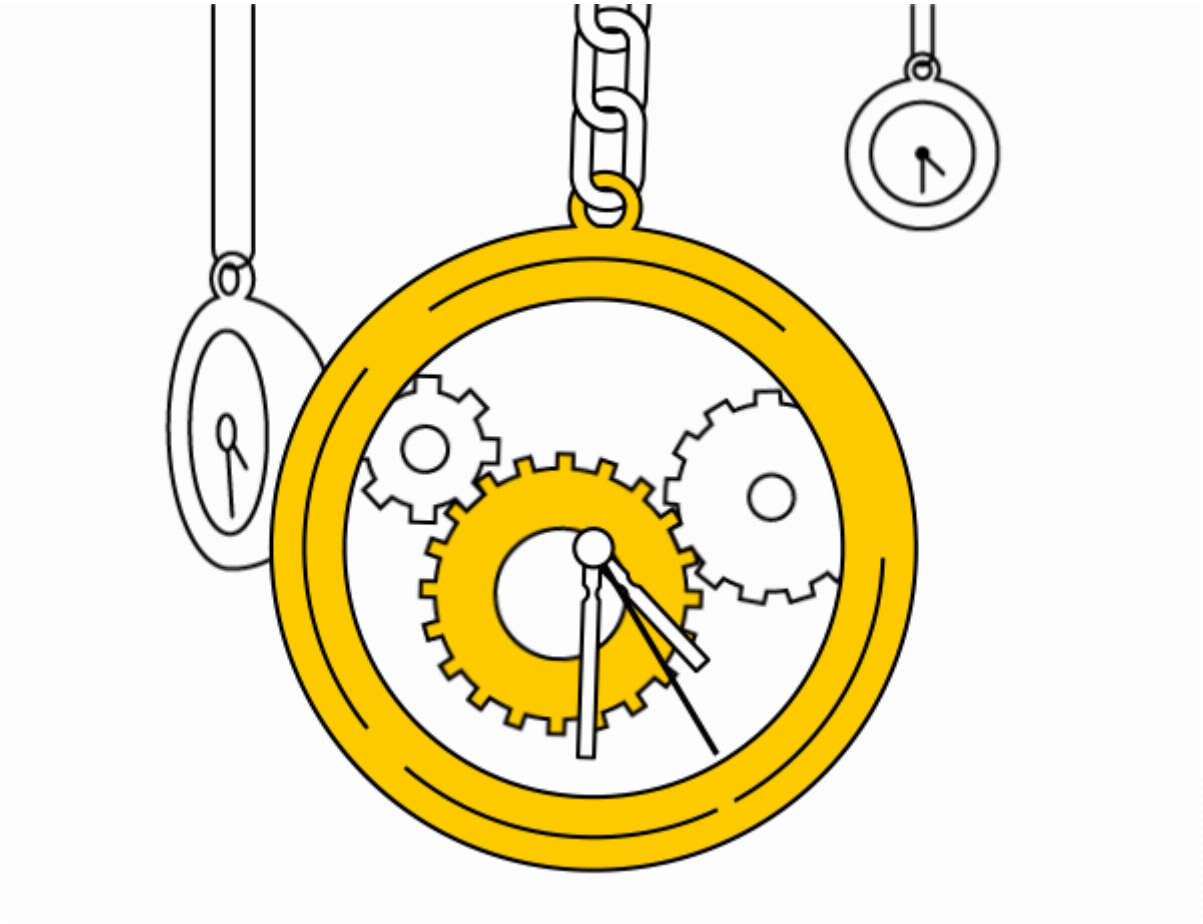
   - Проверяется условие `i < 10`
   - Если оно удовлетворяется, выполняется оператор в теле цикла, т. е. оператор `print(i, '*', j, '=', j * i, sep='', end='\t')`, выводящий на экран строку таблицы умножения в соответствии с текущими значениями переменных `i` и `j`
   - Затем значение итератора `i` внутреннего цикла увеличивается на единицу, и оператор внутреннего цикла `for` проверяет условие `i < 10`. Если условие соблюдается, выполняется тело внутреннего цикла при неизменном значении итератора внешнего цикла до тех пор, пока выполняется условие `i < 10`
   - Если условие `i < 10` не удовлетворяется, т. е. как только `i` станет равен или больше 10, оператор тела цикла не выполняется, внутренний цикл завершается и управление в программе передается за пределы оператора `for` внутреннего цикла, т. е. выполняется перевод строки, вызванный использованием функции `print()` (строка 5), а затем возвращается к оператору `for` внешнего цикла
3. Значение итератора внешнего цикла `j` увеличивается на единицу, и проверяется условие `j < k +`

1. Если условие не соблюдается, т. е. как только `j` станет больше `k`, оператор тела цикла не выполняется, внешний цикл завершается и управление в программе передается за пределы оператора `for` внешнего цикла, т. е. в данном случае программа завершает работу.

Таким образом, на примере печати таблицы умножения показано, что при вложении циклов внутренний цикл выполняется полностью от начального до конечного значения параметра при неизменном значении параметра внешнего цикла. Затем значение параметра внешнего цикла изменяется на единицу, и опять от начала и до конца выполняется вложенный цикл. И так до тех пор, пока значение параметра внешнего цикла не станет больше конечного значения, определенного в операторе `for` внешнего цикла.

2. Графическое представление вложенных циклов

Работу циклов можно сравнить с вращением связанных шестеренок разного размера:



Внешний цикл — это как бы большая шестеренка, за один свой оборот (виток цикла) внешний цикл заставляет вращаться вложенный цикл (меньшую шестеренку) несколько раз.

Обратите внимание: такая иллюстрация точна в случае, если число повторов вложенного цикла не зависит от того, какой именно (1-й, n-й или иной) виток делает внешний цикл, а так бывает не всегда.

3. Оператор break и continue во вложенных циклах

Рассмотрим другую задачу: представьте, что необходимо распечатать все строки таблицы

умножения для чисел от 1 до 10, кроме строки для числа k.

Тогда нам нужно будет пропустить выполнение внутреннего цикла, когда придет k-я строка.

Это можно сделать через оператор `continue`, который просто прервет выполнение данного витка цикла и перейдет к следующей итерации цикла:

```
k = int(input())
for j in range(1, 10):
    if j == k:
        continue
    for i in range(1, 10):
        print(i, '*', j, '=', j * i, sep='', end='\t')
    print()
```

Важно!

Обратите внимание: если оператор `break` или `continue` расположен внутри вложенного цикла, он действует именно на вложенный цикл, а не на внешний. Нельзя выскочить из вложенного цикла сразу на самый верхний уровень.

А теперь попробуйте вывести всю таблицу умножения, кроме столбца k.

Вот еще одна программа, которая использует вложенные циклы и оператор `break`. Она учит пользователя вводить числа палиндромы — программа выполняется до тех пор, пока не будет введено число палиндром:

```
print('Тренажер по вводу палиндрома:')
while True:
    print('Введите число палиндром:')
    number = n = int(input())
    reverse = 0
    while n > 0:
        reverse = reverse * 10 + n % 10
        n //= 10
    if number == reverse:
        print('Вы ввели палиндром! Программа остановлена.')
        break
    print('Введенное число не палиндром, попробуйте еще раз.')
```