Циклы for и while

В языке python есть два ключевых слова `for` и `while` которые применяются для описания циклов.

Цикл for

Пусть дана задача: вывести на экран числа от `0` до `9`. Её можно решить следующим образом:

```
print(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
```

или

```
print(0, end=" ")
print(1, end=" ")
print(2, end=" ")
print(3, end=" ")
print(4, end=" ")
print(5, end=" ")
print(6, end=" ")
print(7, end=" ")
print(8, end=" ")
print(9, end=" ")
```

Этот код решает поставленную задачу, но он получается громоздким, плохо поддерживаемым, так ещё и при изменении диапазона придётся всегда его править.

Для решения подобных задач сущестуют циклы, а задачу перебора диапазона хорошо решает цикл `for`. Цикл `for` в языке `Python` пробегает по указанному объекту и для каждого элемента выолняет тело цикла:

```
for ПЕРЕМЕННАЯ_ЦИКЛА in ОБЪЕКТ:
    БЛОК_ТЕЛО_ЦИКЛА
```

Например:

```
for i in range(10):
    print(i, end=" ")
```

Фунция `range(n)` создаёт список длинны `n` с элементами от `0` до `n-1`, а уже цикл `for` перебирает все элементы списка. На первой итерации в переменную `i` записывается значение `0`, на второй - `1`, на третьей - `2` итд. В результате выполнения этого кода на экран будет выведено `0 1 2 3 4 5 6 7 8 9`, что и требовалось.

Функция `range`, использовавшаяся в этом может быть вызвана с другим набором параметров. Формально общий вид у неё такой:

```
range(start, stop, step)
```

где start - начало диапазона (по умолчанию 0), stop - конец диапазона, а step - шаг (по умолчанию 1). Например:

```
range(10)       # будет создан список [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
range(2, 10)    # будет создан список [2, 3, 4, 5, 6, 7, 8, 9]
range(2, 10, 2) # будет создан список [2, 4, 6, 8]
```

значение step может быть и отрицательным: например, если требуется вывести числа от 0 до -9 это можно сделать так

```
for i in range(0, -10, -1):
    print(i, end=" ")
```

То, что цикл for в Python перебирает элементы списка позволяет перебирать элементы строк, т.е. отдельные символы. Следующий код выведет на экран символы строки Hello, world!, по одному символу на строке:

```
for c in 'Hello, world!':
    print(c)
```

Результат работы:

```
H
e
l
l
o
,

w
o
r
l
d
!
```

Цикл while

Цикл while является более универсальным средством, чем for, т.к. он может решать те же задачи, но он несколько более трудоёмкий в применении. Выглядит он следующим образом:

```
while УСЛОВИЕ_ЦИКЛА:
    БЛОК_ТЕЛО_ЦИКЛА
```

Например, так будет выглядеть решение предыдущей задачи с

использованием цикла while:

```
i = 0
while i < 10:
    print(i, end=" ")
    i += 1
```

Цикл будет выполняться, пока условие цикла истино.

Оператор continue

Ключевое слово continue используется тогда, когда нужно начать следующую итерацию цикла, минуя оставшиеся иструкции в теле цикла. Оператор может применяться как в цикле while, так и for

Следующий код выведет на экран все нечётные числа в диапазоне от 0 до 9:

```
for i in range(10):
    if i%2 != 0:        # Проверяем, если i нечётное
        continue        # Пропускаем итерацию
    print(i, end=" ")
```

Аналогично цикл while

```
i = 0
while i < 10:
    if i%2 != 0:        # Проверяем, если i нечётное
        i += 1          # Увеличиваем i на единицу
        continue        # Пропускаем итерацию
    print(i, end=" ")
    i += 1
```

В результате работы на экране появится строка 2 4 6 8.

Важно! Нужно очень аккуратно работать с циклом while и оператором continue из-за того, что цикл while автоматически не изменяет переменную цикла, как это делает for.

Оператор break

Оператор break нужен, если у нас есть какая-то ситуация, при которой выполнение цикла нужно прекратить досрочно. Оператор может применяться как в цикле while, так и for

Пусть требуется выводить все числа в диапазоне от 1 до 9, пока не встретим число, делящееся на 6.

```
for i in range(1, 10):
    if i%6 == 0:        # Проверяем, делится ли i на 6
        break           # Прерываем цикл
    print(i, end=" ")
```

```
i = 1
while i < 10:
    if i%6 == 0:        # Проверяем, делится ли i на 6
        break           # Прерываем цикл
    print(i, end=" ")
    i += 1
```

В результате работы на экране появится строка `1 2 3 4 5`.

else

Ключевое слово `else`, использованное вместе с циклом `for` и `while`, проверяет, был ли произведён выход из цикла инструкцией `break`. Блок инструкций внутри `else` выполнится только в том случае, если выход произошел из цикла без помощи `break`.

Пусть требуется выводить все числа в диапазоне от `1` до `5`, пока не встретим число, делящееся на `6`. А если такое число не встретилось - вывести строку `В диапазоне нет чисел, делящихся на 6`.

```
for i in range(1, 6):
    if i%6 == 0:        # Проверяем, делится ли i на 6
        break           # Прерываем цикл
    print(i, end=" ")
else:
    print('В диапазоне нет чисел, делящихся на 6')
```

```
i = 1
while i < 6:
    if i%6 == 0:        # Проверяем, делится ли i на 6
        break           # Прерываем цикл
    print(i, end=" ")
    i += 1
else:
    print('В диапазоне нет чисел, делящихся на 6')
```

В результате работы на экран будет выведено:

```
1 2 3 4 5 В диапазоне нет чисел, делящихся на 6
```