Цикл while

Как уже было сказано в предыдущем уроке существуют две основные разновидности цикла:

- циклы, повторяющиеся определенное количество раз (for, счетные циклы);
- циклы, повторяющиеся до наступления определенного события (while, условные циклы)

Мы уже рассмотрели работу цикла `for`, который является счетным циклом. Цикл `for` замечательно работает, если мы заранее знаем, сколько повторений (итераций) нам потребуется сделать. Но иногда нужно, чтобы цикл выполнялся до наступления некоторого события, и количество итераций в этом случае заранее оценить просто невозможно. И здесь на помощь приходит цикл `while`.

Структура цикла `while` в Python выглядит так:

```
while условие:
    блок кода
```

Двоеточие **(:)** в конце строки с инструкцией `while` сообщает Python, что дальше находится **блок команд**. В блок входят все строки, расположенные с отступом от строки с инструкцией `while`, вплоть до следующей строки без отступа.

Блок команд, который выполняется в цикле `while`, называется **телом цикла.**

Рассмотрим код, использующий цикл `while`, который распечатает 10 раз слово `Привет`:

```
i = 0
while i < 10:
    print('Привет')
    i += 1
```

Такой код можно легко заменить циклом `for`, поскольку мы заранее знаем сколько раз нужно выполнить тело цикла. Однако так бывает не всегда.

Напишем программу, которая считывает числа и выводит их квадраты, пока не будет введено -1. При такой постановке задачи мы не можем воспользоваться циклом `for`, так как не знаем сколько чисел будет предшествовать числу -1.

```
num = int(input())
while num != -1:
    print('Квадрат вашего числа равен:', num * num)
    num = int(input())
```

В качестве начального значения переменной `num`, мы используем первое из чисел.

Далее пока выполняется условие цикла, а именно, пока введенное число не равно -1, мы исполняем тело цикла. В тело цикла входит две команды:

1. напечатать квадрат введенного числа;
2. считать следующее число.

Важным являются два момента:

1. правильная инициализация переменной `num`;
2. изменение переменной `num` внутри цикла `while`.

Важно: если не изменять переменную `num` внутри цикла, то можно получить так называемый **бесконечный цикл**, который будет выполняться бесконечно много раз.

Цикл `while` очень похож на условный оператор `if`. Разница заключается в том, что в случае с условным оператором соответствующий блок кода будет выполняться только один раз, тогда как с циклом `while` блок кода будет выполнен многократно.

Цикл for заменяем на цикл while

Мы всегда можем заменить цикл `for` с помощью цикла `while`. Следующие две программы выводят числа от 0 до 100:

```
# используем for
for i in range(101):
    print(i)
```

```
# используем while
i = 0
while i < 101:
    print(i)
    i += 1
```

В первом цикле переменная `i` последовательно принимает значения от 0 до 100. Для цикла `while`, нам пришлось завести самостоятельно переменную `i` и придать ей начальное значение. Тело цикла `while` содержит аналогичную инструкцию вывода `print(i)`, однако помимо этого мы самостоятельно увеличиваем значение переменной `i` на 1, что делается автоматически в случае с циклом `for`.

Напишем программу, выводящую все числа, кратные 3, используя цикл `for` и `while`:

```
# используем for
for i in range(0, 100, 3):
    print(i)
```

```
# используем while
i = 0
while i < 100:
    print(i)
    i += 3
```

Не всегда, однако удается заменить цикл `while` с помощью цикла `for`. Если заранее не известно количество итераций, то необходимо использовать цикл `while` и только его.

Считывание данных до стоп значения

Часто при решении задач на цикл `while`, мы считываем данные, до тех пор пока пользователь не введет некоторое значение, которое называют **стоп значением**. Напишем программу, которая считывает числа и находит их сумму, до тех пор пока пользователь не введет слово `stop`:

```
text = input()
total = 0
while text != 'stop':
    num = int(text)
    total += num
    text = input()
print('Сумма чисел равна', total)
```

Такой код будет часто использоваться при решении задач.

Бесконечный цикл

Всегда, кроме редких случаев, цикл `while` должен содержать возможность завершиться. То есть в цикле что-то должно сделать проверяемое условие ложным. Если цикл не имеет возможности завершиться, то он называется **бесконечным циклом**. Бесконечный цикл продолжает повторяться до тех пор, пока программа не будет прервана. Бесконечные циклы обычно появляются, когда программист забывает написать программный код внутри цикла, который делает проверяемое условие ложным. В большинстве случаев следует избегать применение бесконечных циклов.

Пример бесконечного цикла:

```
i = 0
total = 0
while i < 10:
    total += i
```

Так как в теле цикла не происходит изменения переменной `i`, то условие `i < 10` остается истинным и цикл выполняется бесконечно много раз.

Бесконечные циклы можно использовать в связке с оператором прерывания `break`. Об этом будет рассказано в следующих уроках.

Примечания

Примечание 1. Цикл `while` получил свое название из-за характера своей работы: он выполняет некую задачу до тех пор, **пока** условие является истинным. Слово *while* на английском означает как раз "пока".

Примечание 2. Цикл `while` называют циклом с **предусловием**, поскольку выполнению тела цикла предшествует проверка условия (сначала проверяется условие, а уже затем выполняется тело цикла).

Примечание 3. Однократное выполнение тела цикла называется **итерацией** цикла.

Примечание 4. Цикл `while` может не выполниться ни одного раза. Например, следующий код:

```
i = -1
while i > 0:
    print('Hello world!')
```

не выведет текст, поскольку условие `i > 0` ложно с самого начала.

Примечание 5. Графическое представление цикла `while` имеет вид:

Примечание 6. Условие в цикле `while`, как и в условном операторе `if`, может содержать логические операции `or, and, not`.

Когда цикл `while` проверяет свое условие: до или после того, как он выполнит итерацию?

Вопросы:

1. Сколько раз сообщение «Python awesome!» будет напечатано в приведенном ниже фрагменте кода?

```
count = 10
while count < 1:
    print('Python awesome!')
```

2. Сколько раз сообщение «Python awesome!» будет напечатано в приведенном ниже фрагменте кода?

```
count = 1
while count < 10:
    print('Python awesome!')
    count += 1
```

3. Что покажет приведенный ниже фрагмент кода?

```
i = 7
a = 5
while i < 11:
    a += i
    i += 2
print(a)
```

Решение задач

Сумма чисел

На вход программе подается последовательность целых чисел, каждое число на отдельной строке. Концом последовательности является любое отрицательное число. Напишите программу, которая выводит сумму всех членов данной последовательности.

Формат входных данных

На вход программе подается последовательность чисел, каждое число на отдельной строке.

Формат выходных данных

Программа должна вывести сумму членов данной последовательности.

Входные данные	Выходные данные
1 2 3 4 5 6 7 8 9 10 -5 45 2 3	55
4 -5	4
10 10 -6 43	20

Количество пятерок

На вход программе подается последовательность целых чисел от 1 до 5, характеризующее оценку ученика, каждое число на отдельной строке. Концом последовательности является любое отрицательное число, либо число большее 5. Напишите программу, которая выводит количество пятерок.

Формат входных данных

На вход программе подается последовательность чисел, каждое число на отдельной строке.

Формат выходных данных

Программа должна вывести количество пятерок.

Входные данные	Выходные данные
1 1 2 2 3 4 4 5 5 5	4

5 -17	
5 1 6	1

Заплатите чеканной монетой

При оплате некой услуги принимаются только чеканные монеты. Существуют монеты с номиналами 1,5,10,25. Напишите программу, которая определяет какое минимальное количество чеканных монет нужно заплатить.

Формат входных данных

На вход программе подается одно натуральное число, цена за услугу.

Формат выходных данных

Программа должна вывести минимально возможное количество чеканных монет для оплаты.

Входные данные	Выходные данные
49	7
1	1
5	1