**Тема9. Цикл for**

**Аннотация.** Урок посвящен циклу for. Что такое цикл for и как создавать программы, повторяющие определенные действия? Понятие переменной цикла.

**Цикл for**

В Python существует две основных разновидности циклов:

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

Цикл for замечательно работает, если мы заранее знаем, сколько повторений (итераций) нам требуется сделать.

Рассмотрим код, который распечатает 10 раз слово "Привет":

for i in range(10):

 print('Привет')

Структура цикла for в Python выглядит так:

**for** название\_переменной\_цикла **in** range (количество повторений):

 блок кода

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

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

В предыдущих уроках мы считывали несколько чисел, при помощи нескольких команд input. С помощью цикла for можно считывать и обрабатывать сколько угодно чисел.

Рассмотрим следующий программный код:

for i in range(5):

 num = int(input())

 print('Квадрат вашего числа равен:', num \* num)

print('Цикл завершен')

Такая программа считывает 5 чисел и выводит на экран их квадраты вместе с поясняющей надписью. Поскольку вторая и третья строки выделены отступом, Python считает, что это тело цикла, которое выполняется 5 раз. Четвертая строка не содержит отступа, поэтому не является частью цикла и будет выполнена всего один раз, после того как цикл завершится.

**Примеры использования цикл**

Рассмотрим следующий программный код:

print('A')

print('B')

for i in range(5):

 print('C')

 print('D')

print('E')

Результатом выполнения такой программы будут строки

A

B

C

D

C

D

C

D

C

D

C

D

E

То есть сначала программа распечатает символы А и В, затем символы C и D пять раз, а затем распечатает символ Е один раз. Тело цикла состоит из двух строк: четвертой и пятой и именно они будут повторяться.

В программе может быть сколько угодно циклов. Например, если мы хотим, чтобы сначала 5 раз был распечатан символ С, а затем 5 раз символ D, мы можем использовать 2 цикла:

print('A')

print('B')

for i in range(5):

 print('C')

for i in range(5):

 print('D')

print('E')

Результатом выполнения такой программы будут строки:

A

B

C

C

C

C

C

D

D

D

D

D

E

**Примечания:**

**Примечание 1.** Однократное выполнение тела цикла называется **итерацией цикла**.

**Примечание 2.** Графическое представление цикла for имеет вид:



**Примечание 3.**Напомним, что **блоком кода** называют объединённые друг с другом строки. Они всегда связаны с определённой частью программы (например, с инструкцией if или for). В Python блоки кода формируются при помощи **отступов**.

**Примечание 4.** Слово for пишется маленькими буквами, первая строка должна заканчиваться двоеточием, и тело цикла должно быть выделено отступом.

**Задача 9\_1. Python is a wesome**

Напишите программу, которая выводит слова «Python is a wesome!» (без кавычек) 10 раз.

**Формат входных данных**

**Формат выходных данных**
Программа должна вывести 10 раз текст «Python is a wesome!», каждый на отдельной строке.

## Задача 9\_2.Повторяй за мной 1

Дано предложение и количество раз, которое его надо повторить. Напишите программу, которая повторяет данное предложение нужное количество раз.

**Формат входных данных**
В первой строке записано текстовое предложение, во второй — количество повторений.

**Формат выходных данных**
Программа должна вывести указанное текстовое предложение нужное количество раз. Каждое повторение должно начинаться с новой строки.

**Sample Input 1:**

Век живи - век учись.

10

**Sample Output 1:**

Век живи - век учись.

Век живи - век учись.

Век живи - век учись.

Век живи - век учись.

Век живи - век учись.

Век живи - век учись.

Век живи - век учись.

Век живи - век учись.

Век живи - век учись.

Век живи - век учись.

## Задача 9\_3. Последовательность символов

Напишите программу, которая использует **ровно три цикла for** для печати следующей последовательности символов:

AAA

AAA

AAA

AAA

AAA

AAA

BBBB

BBBB

BBBB

BBBB

BBBB

E

TTTTT

TTTTT

TTTTT

TTTTT

TTTTT

TTTTT

TTTTT

TTTTT

TTTTT

G

## Переменная цикла

Давайте еще раз взглянем на базовую структуру цикла for:

For название\_переменной\_цикла in range(количество повторений):

 блок кода

Не совсем понятно, для чего нужна и как работает переменная цикла.

Рассмотрим следующий код:

for i in range(10):

 print(i)

Результатом выполнения такого кода будет:

0

1

2

3

4

5

6

7

8

9

Когда цикл впервые начинает работу Python устанавливает значение переменной цикла i = 0. Каждый раз когда мы повторяем тело цикла Python увеличивает значение переменной на 1.

Почему большинство программистов начинают цикл с 0, а не с 1? Раньше некоторые начинали цикл с 1, а некоторые с 0. Те и другие приводили весьма изощренные аргументы, споря о том, какой способ лучше. Но в конце концов победили сторонники второго варианта. С тех пор большинство начинает циклы с 0. В частности, в Python цикл for начинается с 0, однако это можно изменить.

Поскольку переменная цикла i увеличивается на 1 каждый раз, то ее можно использовать для отслеживания номера итерации, на которой мы находимся в циклическом процессе.

Рассмотрим следующий код:

for i in range(10):

 print(i, '-- Привет')

Результатом выполнения такого кода будет:

0 -- Привет

1 -- Привет

2 -- Привет

3 -- Привет

4 -- Привет

5 -- Привет

6 -- Привет

7 -- Привет

8 -- Привет

9 -- Привет

Если мы хотим начать с 1, то можем написать код:

for i in range(10):

 print(i + 1, '-- Привет')

 Результатом выполнения такого кода будет:

1 -- Привет

2 -- Привет

3 -- Привет

4 -- Привет

5 -- Привет

6 -- Привет

7 -- Привет

8 -- Привет

9 -- Привет

10 -- Привет

Обратите внимание, за счет выражения i + 1, мы начинаем вывод с 1, а не с 0.

## ****Имена переменных цикла****

Ранее говорилось, что имена переменных должны носить осмысленный характер и описывать их назначение. Однако для переменных цикла иногда делается исключения. В программировании для переменных цикла **обычно** используют буквы i, j, k.

Следующие две программы абсолютно одинаковые: в первой программе переменная цикла имеет название i, во второй программе number:

for i in range(10): for number in range(10):

 print(i) print(number)

Результатом выполнения обеих программ будет:

0

1

2

3

4

5

6

7

8

9

Почему для переменной циклов зарезервированы буквы i, j, k? Дело в том, что раньше программы использовались для математических расчетов, а в математике буквы a, b, c и x, y, z уже зарезервированы для других целей. Поэтому программисты выбрали для этой цели переменные i, j, k и это стало общепринятой практикой.

Бывают ситуации когда переменная цикла не используется в теле цикла. В таком случае, вместо того, чтобы давать ей имя, мы можем указать символ нижнего подчеркивания \_:

for \_ in range(5):

print('Python - awesome!')

Результатом выполнения такого кода будет:

Python - awesome!

Python - awesome!

Python - awesome!

Python - awesome!

Python - awesome!

   Если переменная цикла не используется в теле цикла, то указывайте вместо нее символ нижнего подчеркивания \_.

##

**Примечание.** Следует помнить, что правая граница цикла в Python всегда не включительна. Таким образом следующий код:

for i in range(5):

 print(i)

распечатает числа от 0 до 4:

0

1

2

3

4

Если требуется распечатать числа от 1 до 5, то мы пишем код:

for i in range(5):

 print(i +1)

## Задача9\_4. ****Повторяй за мной 2****

Напишите программу, которая считывает одну строку текста и выводит 10 строк, пронумерованных от 0 до 9, каждая с указанной строкой текста.

**Формат входных данных**
На вход программе подается одна строка текста.

**Формат выходных данных**
Программа должна вывести десять строк в соответствии с условием задачи.

**Sample Input 1:**

LeBron

**Sample Output 1:**

0 LeBron

1 LeBron

2 LeBron

3 LeBron

4 LeBron

5 LeBron

6 LeBron

7 LeBron

8 LeBron

9 LeBron

**Sample Input 2:**

Roman

**Sample Output 2:**

0 Roman

1 Roman

2 Roman

3 Roman

4 Roman

5 Roman

6 Roman

7 Roman

8 Roman

9 Roman

## Задача 9\_5. Квадрат числа

На вход программе подается натуральное число n. Напишите программу, которая для каждого из чисел от 0 до *n* (включительно) выводит фразу: «Квадрат числа [число] равен [число]» (без кавычек).

**Формат входных данных**
На вход программе подается натуральное число n.

**Формат выходных данных**
Программа должна вывести текст в соответствии с условием задачи.

**Sample Input 1:**

9

**Sample Output 1:**

Квадрат числа 0 равен 0

Квадрат числа 1 равен 1

Квадрат числа 2 равен 4

Квадрат числа 3 равен 9

Квадрат числа 4 равен 16

Квадрат числа 5 равен 25

Квадрат числа 6 равен 36

Квадрат числа 7 равен 49

Квадрат числа 8 равен 64

Квадрат числа 9 равен 81

**Тема 10. Цикл while**

**Аннотация.** Урок посвящен циклу while. Мы научимся создавать программы, повторяющие определенные действия пока выполняется некоторое условие.

**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

whilei<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 VS 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

whilei<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

whilei<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 с 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".

**Примечение 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('Helloworld!')

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.

## Обработка цифр числа

При изучении целых чисел (тип данных int), мы говорили про операцию целочисленного деления // и операцию нахождения остатка от деления одного целого числа на другое %. Используя цикл while и две данных операции, можно обработать цифры числа с произвольным количеством разрядов (цифр).

Пусть дано натуральное число n. Тогда:

* результатом операции n % 10 – является последняя цифра числа;
* результатом операции n // 10 – является число с удаленной последней цифрой.

Напишем программу, которая считывает натуральное число (целое положительное) и обрабатывает его цифры.

n = int(input())

while n != 0: # пока в числе есть цифры

 last\_digit = n % 10# получить последнюю цифру

# код обработки последней цифры

 n = n // 10# удалить последнюю цифру из числа

Цикл while работает до тех пор, пока в числе есть необработанные цифры. Тело цикла содержит:

1. процедуру получения последней цифры last\_digit = n % 10;
2. код обработки последней цифры;
3. процедуру удаления последней цифры из числа n = n // 10.

В качестве процедуры обработки может быть все, что угодно: вывод цифр, нахождение суммы, произведения цифр, нахождение наибольшей или наименьшей цифры, подсчет цифр удовлетворяющих некоторому условию и т.д.

Напишем программу, которая определяет, есть ли в числе цифра 7.

num = int(input())

has\_seven = False #сигнальная метка

while num != 0:

 last\_digit = num % 10

 if last\_digit == 7:

 has\_seven = True

 num = num // 10

if has\_seven == True:

 print('YES')

else:

 print('NO')

Решите задачи, используя цикл While:

## Задача 10\_1. Числа

На вход программе подаются два натуральных числа a и b, a<b.

Нужно вывести все числа в диапазоне от a до b. Включительно.

**Sample Input:**

3

8

**Sample Output:**

3

4

5

6

7

8

## Задача 10\_2. Нечётные числа

На вход программе подаётся натуральное число.

Нужно вывести в возрастающем порядке все натуральные нечётные числа до него. Каждое число нужно вывести в отдельной строке, само число выводить не нужно.

**Sample Input:**

12

**Sample Output:**

1

3

5

7

9

11

## Задача 10\_3. Улитка

Улитка ползёт вверх по столбу высотой n метров. Каждый день она проползает вверх на 3 метра, а каждую ночь съезжает вниз на 2 метра. За сколько дней она доползёт до вершины столба.

Для введённого n вычислите количество дней, необходимых для того, чтобы улитка добралась до вершины столба.

**Sample Input:**

5

**Sample Output:**

3

## Задача 10\_4. Сумма цифр

На вход подаётся число. Необходимо вычислить и вывести сумму цифр этого числа.

**Sample Input:**

34

**Sample Output:**

7

## Задача 10\_3. Количество цифр

Напишите программу, которая определяет, есть ли в числе цифра 3, если такая цифра есть, то считает их количество.

**Sample Input:**

334

**Sample Output:**

2

## Задача 10\_6. Куб числа

На вход программе подается натуральное число n. Напишите программу, которая для каждого из чисел от 0 до *n* (включительно) выводит фразу: «Квадрат числа [число] равен [число]» (без кавычек).

**Формат входных данных**
На вход программе подается натуральное число n.

**Формат выходных данных**
Программа должна вывести текст в соответствии с условием задачи.

**Sample Input 1:**

3

**Sample Output 1:**

Куб числа 0 равен 0

Куб числа 1 равен 1

Куб числа 2 равен 8

Куб числа 3 равен 27

## Задача 10\_7. Степень двойки

На вход программе подается натуральное число n (степень числа). Напишите программу, которая выводит фразу: «2 ^ [число]   =  [число]» (без кавычек).

**Формат входных данных**
На вход программе подается натуральное число n.

**Формат выходных данных**
Программа должна вывести текст в соответствии с условием задачи.

**Sample Input 1:**

5

**Sample Output 1:**

2 ^ 1 = 2

2 ^ 2 = 4

2 ^ 3 = 8

2 ^ 4 = 16

2 ^ 5 = 32