# Практическое занятие: Реализация циклических алгоритмов в Python. Функция *range().* Синтаксис цикла *for,* цикла *while*.

# Содержание

1. Циклы в Python
2. Цикл for в Python
3. Цикл while в Python
4. Задания для самостоятельной работы
5. **Циклы в Python**

Циклы являются мощнейшим инструментом, предоставляемым высокоуровневыми языками программирования. Эти управляющие конструкции позволяют многократно выполнять требуемую последовательность инструкций. Программа повторяет определённое действие или действия несколько раз. Действие задано в теле цикла, число повторений — в его условии. В условии может быть указано вполне конкретное число повторений цикла, например пять или десять. А может быть более сложная проверка — например, анализ переменной или переменных из тела цикла. Одно повторение тела цикла называется *итерацией*.

Принципиальная блок-схема простого цикла:

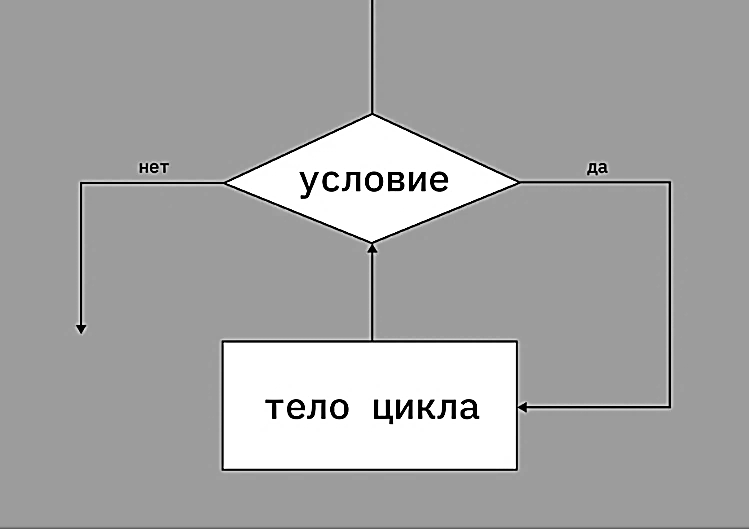


Рис. 1. Принципиальная схема простейшего цикла

Циклы в языке Python представлены двумя основными конструкциями: for и while.

Эти два типа циклов могут быть применены ко всем структурам данных языка Phyton: к числам и диапазонам чисел, к спискам чисел и символов, к кортежам, к словарям и так далее.

## Применение циклов

Концепция циклов — абстрактная концепция повторяющихся операций или действий. Повторяющиеся раз за разом операции окружают нас и в реальной жизни:

* смена времён года;
* смена дня ночью и наоборот;
* следование строгому расписанию каждый будний день, пока не наступят выходные.
* повторяющиеся день за днем занятия по расписанию;
* цикличность расписаний гражданского транспорта;

Всё это есть циклы, и представить нормальную жизнь без них попросту невозможно.

Повторяющиеся действия или операции на каждом шагу, практически, в каждой задаче касается и программирования.

Например, нужно последовательно отобразить числа от 0 до 100000. Без операций циклов, пришлось бы выполнять вручную это действие, что потребовало бы колоссального количества повторного кода и огромных временных затрат:

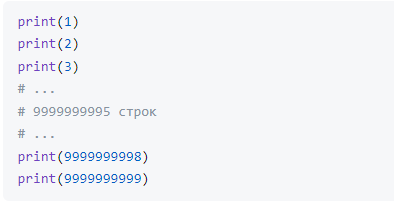


Рис.2 Без циклов повторяющуюся операцию ввода чисел пришлось бы вводить 999…99 раз

Циклы же позволяют уместить такую многокилометровую запись в изящную и простую для понимания конструкцию, состоящую всего из нескольких строчек кода.

1. **Цикл *for* в Python**

Цикл for используется для управления итерациями, когда их количество известно заранее.

Например, блок-схема цикла for имеет вид:

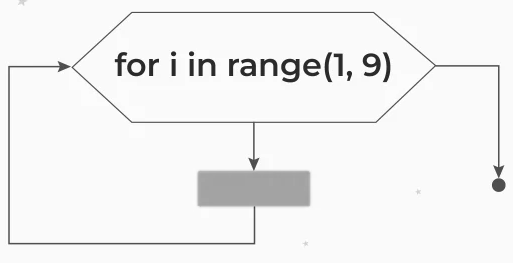


Рис.3 Блок-схема цикла for в Python

## Синтаксис цикла *for*:

For <имя\_переменной> in <массив>

<действия>

## Пример 1. Простейший цикл *for.*

Для того чтобы выполнить какую-либо инструкцию строго определенное число раз, воспользуемся функцией *range():*

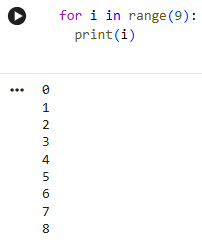


Рис.4. Цикл прохода по числам ряда.

Цикл можно организовать и на списке чисел без использования функции *range().* Для этого достаточно сформировать так называемый список – list(), используя синтаксис языка. Например, цикл можно организовать на любом списке чисел.

Пример 2. Цикл «читает» список - последовательность [1, 3, 5, 7, 9, 11, 13] и печатает их:

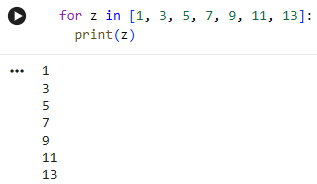


Рис. 5. Проход по списку с использованием цикла for.

Пример 3. Чтобы запустить цикл к списку можно обратиться и по имени этого списка:

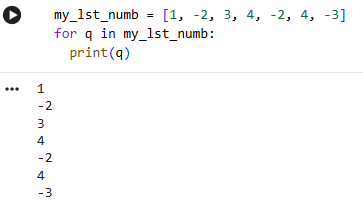


Рис.6. Проход по циклу всех значений списка, вызванного по имени

В цикле можно выполнять различные операции с переменными, например, как в следующем примере.

Пример 4. Среди первой сотни чисел натурального ряда отыскиваются числа, делящиеся на 13.

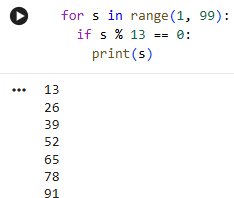


Рис. 7. Список чисел, делящихся на 13 среди первой сотни чисел натурального ряда.

В коде выше с помощью цикла for выполняется проход по числам от 1 до 99 (100 первых чисел натурального ряда) и проверяется делимость нацело их на число 13:



И числа, у которых после деления на 13 получается в остатке 0, выводятся на экран. Количество чисел, делящихся на 13, можно посчитать прямо в цикле for. Подсчет производится строкой кода в теле цикла for:

.

Например, следующим образом:

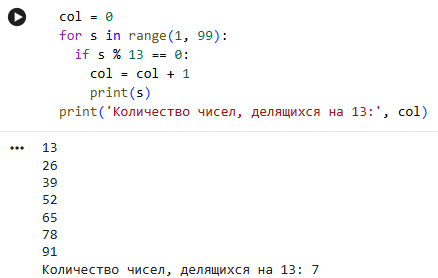


Рис. 8. Подсчет количества чисел, делящихся на 13 в первой сотне натурального ряда.

Переменная col задана вне цикла со значением 0, а меняет свое значение с каждой итерацией, если условие выполнено.

Задание. Внедрите в тело цикла код, позволяющий вычислить сумму чисел, делящихся на 13.

Задание. По аналогии с предыдущим примером проверить делимость чисел последовательности первых 300 натуральных чисел на 17, посчитать их количество и вычислить их сумму. В частности, на элементах списках можно c использованием for выполнять различные арифметические операции.

Пример 5. Задан список чисел. Требуется проверить условие делимости чисел списка на заданное пользователем число (например, 2, 3, 5, 11, 19, 23 и другие) и вывод этих чисел на экран в виде списков в виде двух отсортированных по делимости чисел:

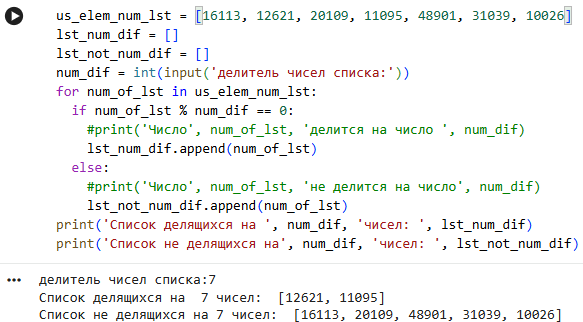


Рис. 9. Код, в котором в выполняющемся цикле for проверяются условия делимости чисел последовательности из списка на число, введенное пользователем, формируются два списка с помощью метода append(), в которые включаются проверенные на делимость числа.

Пример 6.

Используя оператор for пересчитываются от 0 до 9 значения двух выражений: квадраты чисел от 0 до 9 и значения выражения:

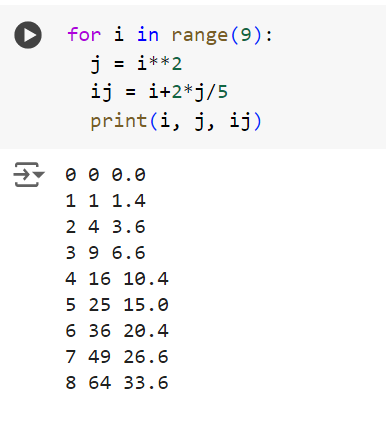


Рис.10. Результат выполнения цикла for инструкций в теле цикла.

Пример 7.

Отобразить список чисел из диапазона от 0 до 50 тех чисел, которые делятся и на 2, и на 3:

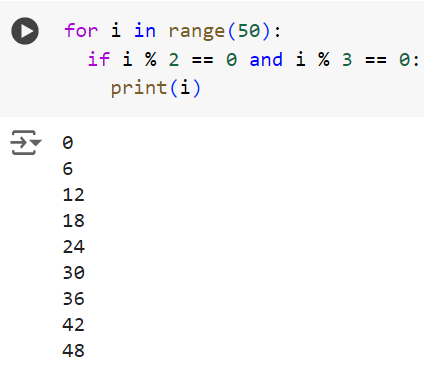


Рис. 11. Цикл прохода по числам списка и определение их делимости на 2 и на 3

Задание. Добавьте в программу примера 7 строки в цикл, с помощью которых программа вычислит отдельно сумму всех чисел делящихся на 2 и на 3 отдельно, и сумму чисел делящихся на 2 и на 3.

Цикл *for* в python — это средство для перебора последовательностей. С его помощью можно совершать обход строк, списков, кортежей и описанных выше итерируемых объектов. В простейшем случае он выглядит так:

Изображение выглядит как текст, Шрифт, белый, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис.12. Последовательность collection в цикле for.

Если последовательность collection состоит, скажем, из 4 элементов, цикл for будет поочерёдно обходить их, храня значение текущего элемента в переменной str.

Пример 8.

Перечислить времена года в порядке перечисления в списке с использованием цикла for:

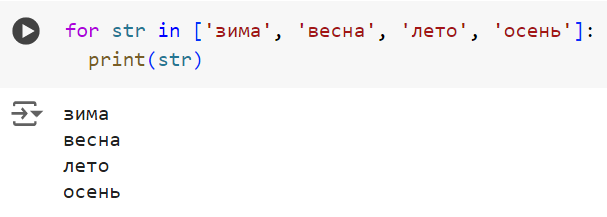


Рис. 13. Цикл по collection.name = str in [‘зима’, ‘весна’, ‘лето’, ‘осень’]

Пример 9.

В примере последовательность, что приведена ниже, collection.name = my\_frinds: состоит из 8 символьных элементов:

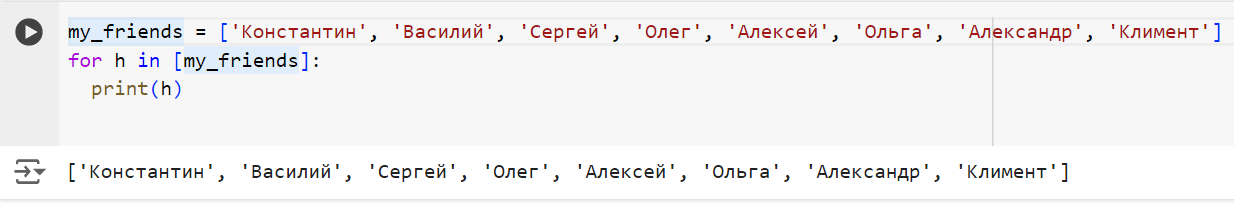


Рис. 14. Цикл по collection.name =my\_frends

Задание. Сформируйте цикл повторения приветствия 5 раз.

Пример 10.

Перечислить дни недели используя цикл for по переменной day\_week из списка collection.name = days\_week:



Рис. 15. Цикл day\_week in days\_in\_week.

Задание. Перечислить список предметов первого курса в КТЭК, используя переменную sp\_item из списка sp\_items:

sp\_items = [Химия, Физика, Математика, Английский язык, География, Информатика, Введение в специальность, Биология, Русский язык, Русская литература, Зарубежная литература, История, ОБЖ]

### Пример 11. На введенное пользователем число k программа генерирует k строк c k символами \*:

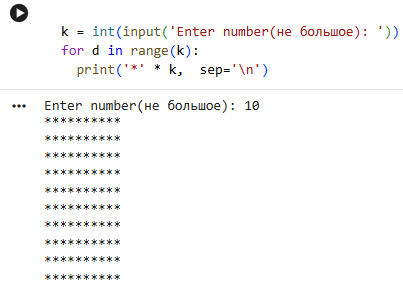


Рис. 16. Выполнение цикла k строк по k символов \* в строке

Пример 12. Программа, которая получает от пользователя число **n** и выводит u строк с результатом умножения чисел от **1** до **u** на символ **\***.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 17. Цикл формирования треугольника звездочек

### Задание. Написать цикл, возвращающий «перевернутый», относительно предыдущей задачи, треугольник. То есть треугольник начинается с максимального количества \* и заканчивается одной \*, то есть, как на рисунке (потребуются усилия):



Рис. 18. Цикл формирования треугольника звездочек

Пример 13. Сформировать треугольник, на четной строке которого максимальное количество \*.

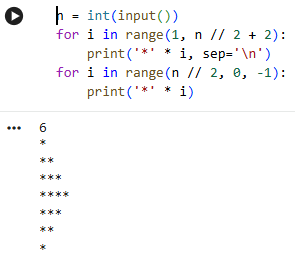


Рис. 18. Результат выполнения цикла

Задание. Сформировать треугольник, симметричный относительно вертикали аналогично предыдущему примеру, то есть как на рисунке (потребуются усилия):

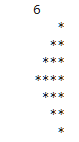


Рис. 19. Результат выполнения цикла

Задание. Сформировать треугольник, используя цикл for, то есть как на рисунке (потребуются усилия):



Рис. 20. Результат выполнения цикла

Задание. Сформировать треугольник, симметричный относительно горизонтали, аналогично предыдущему примеру, то есть как на рисунке (потребуются усилия):



Рис.21. Результат выполнения вашего кода должен быть таким.

Пример 14. В примере с помощью циклов for и двух символов: и формируется «снежинка»:

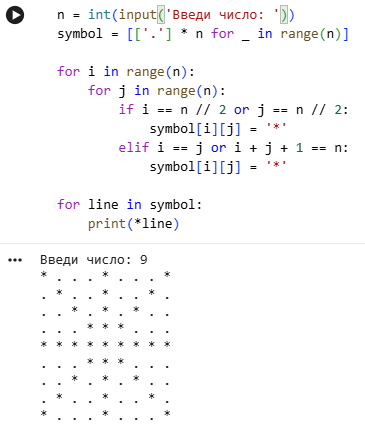


Рис. 22. Сформированный символ

Задание. Сформируйте «снежинку» с помощью тех же символов:

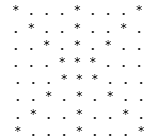


Рис. 23. Результат выполнения цикла

Пример 15.

Разделить символы заданной строки знаком «\*»

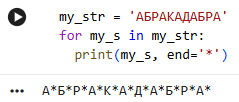


Рис. 24. Результат выполнения программы по разделению символов строки звездочкой \*.

Задание. В предыдущем примере вывести на экран все буквы «А» и отдельным сообщением (print(«Сообщение»)) посчитать их количество.

Задание. Создайте с помощью стандартного ввода в Python текстовое предложение или фразу, или длинное слово, типа «рентгеноэлектрокардиографический» и создайте цикл подсчета используемых в нем букв. Отобразите статистику использования букв в текстовой переменной.

Цикл for используется и для задания более сложных циклов, например, при задании матриц различной размерности.

Пример 16.

Составление матрицы с заданными размерами (то есть количеством строк и столбцов) и введенными элементами. Требуется составить матрицу по введенной размерности и вводимыми вручную ее элементами:

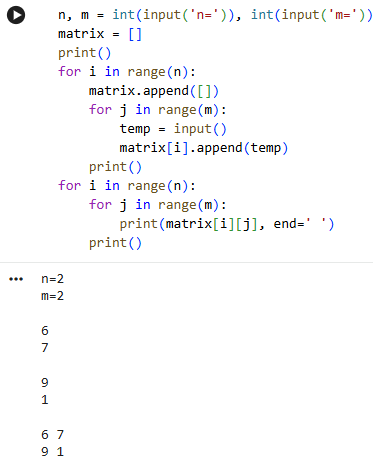


Рис. 25. Цикл формирования строк и столбцов матрицы размерности 2.

1. **Цикл *while* в Python**

Цикл for используется, когда количество итераций известно, а вот когда количество итераций неизвестно, используется цикл while.

Блок-схема while цикла имеет вид:

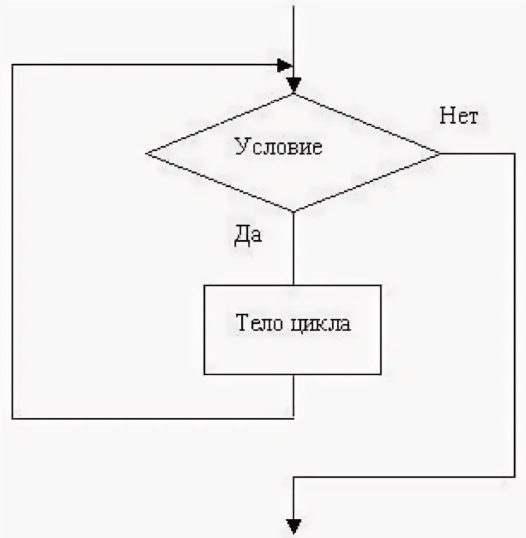


Рис. 26. Блок-схема цикла while.

Пример 17. Простейший пример цикла while: задано первоначальное значение переменной, равное 0. И пока число меньшее 8, печатать на экран следующее число, большее предыдущего значения на единицу:

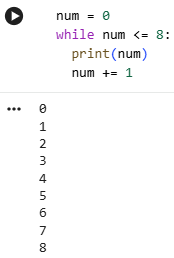


Рис. 27. Простейший пример с использованием цикла while

Пример 18. Код, реализующий классический вариант поиска НОД двух чисел:

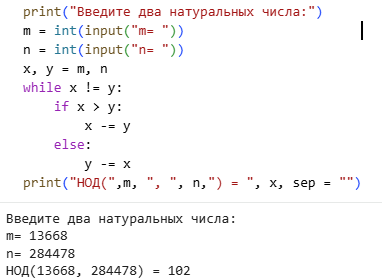


Рис.28. Вычисление наибольшего общего делителя для двух введенных чисел

Пример 19. Код вычисления факториала числа, вводимого стандартным методом ввода:

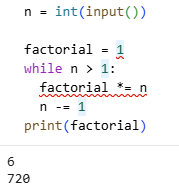


Рис. 29. Вычисление факториала числа с использованием цикла while

Пример 20. Пример программы с циклом while подсчета четных цифр во введенном числе:

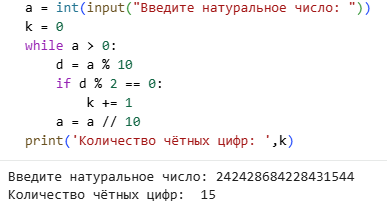


Рис. 30. Поиск четных цифр во введённом числе.

Задания для самостоятельной работы

Задание 1. Сформируйте цикл в котором при их обходе чисел последовательности будет проверяться условие делимости числа на 3 и 5 или 7 и 11 м выводиться список этих чисел.

Задание 2. Постройте алгоритм вычисления НОД для трех чисел.

Задание 3. Перестроить пример 20 так, чтобы в программе вычислялось количество нечетных цифр и их сумма.

В циклах for и while в языке Python можно использовать ещё два оператора:

● *break -* прерывает цикл, если выполнено конкретное условие или несколько условий одновременно. Его используют, если при каком-то значении переменной работу цикла нужно немедленно прекратить.

● *continue*:оператор**,** наоборот, мгновенно заставляет программу перейти к началу цикла и начать следующую итерацию. Его используют, чтобы при определённых условиях пропустить часть действий из тела цикла и сразу начать новое повторение.

Пример 6. Пример кода с использованием оператора *break* в цикле while.

В примере реализуется чтение списка чисел коллекции с помощью цикла **for** и реализован выход из цикла с помощью оператора *break*, если в последовательности встретилась единица.

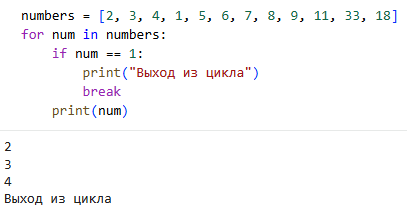


Рис. 31. Простейший пример с использованием оператора цикла while

Пример 7. Пример кода с использованием оператора *continue* в цикле while.

В примере реализуется алгоритм поиска чисел, делящихся на 7 в введенном диапазоне и найденные числа суммируются:

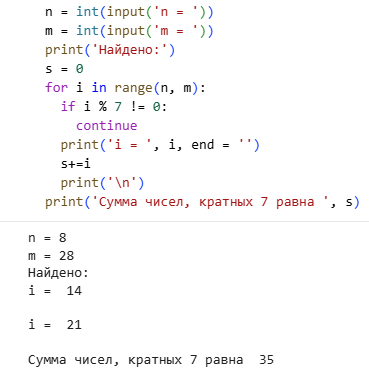


Рис. 32. Пример использования оператора continue в цикле while

Задания для самостоятельной работы

1. Найти все числа от 1 до 200, делящиеся на 5 и на 7 одновременно.

2. Найти все числа от 1 до 1000, делящиеся на 5 или на 7, или на 11.

3. Найти все числа от 10 до 1000, заканчивающиеся на цифру 3.

4. Найти все числа от 100 до 999 сумма цифр которых равна 18.

5. Написать программу возвращающую последовательность:

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0

0 0 0 0 0 0

0 0 0 0 0

0 0 0 0

0 0 0

0 0

0

6. Написать программу возвращающую последовательность:

0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1

0 0 0 0 0 0 0 0 0

1 1 1 1 1 1 1 1 1

0 0 0 0 0 0 0 0 0

7. Написать программу возвращающую последовательность:

0 0 0 0 0 0

1 1 1 1 1 1

2 2 2 2 2 2

3 3 3 3 3 3

4 4 4 4 4 4

5 5 5 5 5 5

8. Написать программу возвращающую последовательность:

0

1 1

2 2 2

3 3 3 3

4 4 4 4 4

5 5 5 5 5 5

6 6 6 6 6 6 6

7 7 7 7 7 7 7 7

9. Написать программу, в которой реализовать цикл поиска числа в заданной последовательности, указав индекс найденного элемента и количество элементов, оставшихся до конца последовательности.