Как правило, циклы **for** используются либо для повторения какой-либо последовательности действий заданное число раз, либо для изменения значения переменной в цикле от некоторого начального значения до некоторого конечного.

Для повторения цикла некоторое заданное число раз n можно использовать цикл **fo**r вместе с функцией **range**:

**for i in range(n):**

**Тело цикла**

В качестве n может использоваться числовая константа, переменная или произвольное арифметическое выражение (например, 2 \*\* 10). Если значение n равно нулю или отрицательное, то тело цикла не выполнится ни разу.

Если задать цикл таким образом:

**for i in range(a, b):**

**Тело цикла**

то индексная переменная i будеть принимать значения от a до b – 1, то есть первый параметр функции range, вызываемой с двумя параметрами, задает начальное значение индексной переменной, а второй параметр — значение, которая индексная переменная принимать не будет. Если же a≥b, то цикл не будет выполнен ни разу. Например, для того, чтобы просуммировать значения чисел от 1 до n можно воспользоваться следующей программой:

**sum = 0**

**for i in range(1, n + 1):**

**sum += i**

В этом примере переменная i принимает значения 1, 2, …, n, и значение переменной sum последовательно увеличивается на указанные значения.

Наконец, чтобы организовать цикл, в котором индексная переменная будет уменьшаться, необходимо использовать функцию

**range**

 с тремя параметрами. Первый параметр задает начальное значение индексной переменной, второй параметр — значение, до которого будет изменяться индексная переменная (не включая его!), а третий параметр — величину изменения индексной переменной. Например, сделать цикл по всем нечетным числам от 1 до 99 можно при помощи функции

**range(1, 100, 2)**

, а сделать цикл по всем числам от 100 до 1 можно при помощи

**range(100, 0, -1)**

.

мы используем **цикл** в тех случаях, когда вам нужно повторить что-нибудь n-ное количество раз. Это проще понять, если взглянуть на пример. Мы используем встроенную [функцию Python](https://python-scripts.com/functions-python) range. [Функция range](https://python-scripts.com/range) создаст список длинной в «n» элементов

print(range(5)) # ответ: range(0, 5)

Как вы видите, функция **range** взяла целое число и вернула **объект range**. Функция range также принимает начальное значение, конечное значение и значение шага. Вот еще два примера:

*a = range(5, 10)*

*print(a) # range(5, 10)*

Что здесь произошло? Давайте почитаем слева на право, чтобы понять это. Для каждого числа в диапазоне 5 мы вводим число. Мы знаем, что если мы вызываем range со значением 5, мы получим список из 5 элементов. Так что каждый раз, проходя через цикл, она выводит каждый из элементов. [Цикл for](https://python-scripts.com/loops-for-while), показанный выше, может быть эквивалентом следующего:

Python

|  |  |
| --- | --- |
| 1  2 | for number in [0, 1, 2, 3, 4]:      print(number) |

Функция **range** лишь делает результат несколько меньшим. **Цикл for** может обходить любой итератор Python.

for number in range(10):

    if number % 2 == 0:

        print(number)

**i = 1**

**for color in 'red', 'orange', 'yellow', 'green', 'cyan', 'blue', 'violet':**

**print(i, '-th color of rainbow is ', color, sep = '')**

**i += 1**

В этом примере переменная **color** последовательно принимает значения ‘red’, ‘orange’ и т.д. В теле цикла выводится сообщение, которое содержит название цвета, то есть значение переменной **color**, а также номер итерации цикла – число, которое сначала равно 1, а потом увеличивается на один (инструкцией

**i += 1**

 с каждым проходом цикла).

В списке значений могут быть выражения различных типов, например:

**for i in 1, 2, 3, 'one', 'two', 'three':**

**print(i)**

При первых трех итерациях цикла переменная i будет принимать значение типа **int**, при последующих трех — типа **str**.

|  |
| --- |
| #! Программа по вычислению факториала    number = int(input("Введите число: "))  factorial = 1  for i in range(1, number+1):      factorial \*= i  print("Факториал числа", number, "равен", factorial) |

Вначале вводим с консоли число. В цикле определяем переменную i, в которую сохраняются числа из коллекции, создаваемой функцией range.

Функция range здесь принимает два аргумента - начальное число коллекции (здесь число 1) и число, до которого надо добавлять числа (то есть number +1).

Допустим, с консоли вводится число 6, то вызов функции range приобретает следующую форму:

|  |  |
| --- | --- |
| 1 | range(1, 6+1): |

Эта функция будет создавать коллекцию, которая будет начинаться с 1 и будет последовательно наполняться целыми числами вплоть до 7. То есть это будет коллекция [1, 2, 3, 4, 5, 6].

При выполнении цикла из этой коллекции последовательно будут передаваться числа в переменную i, а в самом цикле будет происходить умножение переменной i на переменную factorial. В итоге мы получим факториал числа.

Консольный вывод программы:

Введите число: 6

Факториал числа 6 равен 720

**Функция range**

Функция range имеет следующие формы:

* **range(stop)**: возвращает все целые числа от 0 до stop
* **range(start, stop)**: возвращает все целые числа в промежутке от start (включая) до stop (не включая). Выше в программе факториала использована именно эта форма.
* **range(start, stop, step)**: возвращает целые числа в промежутке от start (включая) до stop (не включая), которые увеличиваются на значение step

Примеры вызовов функции range:

|  |  |
| --- | --- |
| 1  2  3  4 | range(5)            # 0, 1, 2, 3, 4  range(1, 5)         # 1, 2, 3, 4  range(2, 10, 2)     # 2, 4, 6, 8  range(5, 0, -1)     # 5, 4, 3, 2, 1 |

Например, выведем последовательно все числа от 0 до 4:

|  |  |
| --- | --- |
| 1  2 | for i in range(5):      print(i, end=" ") |

333

|  |
| --- |
| a,b =int(input()),int(input()) |
|  | s = "" |
|  | for i in range(a,b+1): |
|  | if i%2==0: |
|  | print(i,end = " ") |

334

|  |
| --- |
| a=int(input()) |
|  | b=int(input()) |
|  | c=int(input()) |
|  | d=int(input()) |
|  | for i in range (a,b): |
|  | if(i%d==c): |
|  | print(i, end=" ") |

335

|  |
| --- |
| import math |
|  | a=int(input()) |
|  | b=int(input()) |
|  | for i in range(a,b): |
|  | if math.sqrt(i)%1==0: |
|  | print(i) |

339

|  |
| --- |
| a = int(input()) |
|  |  |
|  | for i in range(2,a+1): |
|  | if a%i==0: |
|  | print(i) |
|  | break |

340

|  |
| --- |
| a=int(input()) |
|  | for i in range(1,a+1): |
|  | if a%i==0: |
|  | print(i, end=" ") |

341

|  |
| --- |
| a=int(input()) |
|  | c=0 |
|  | for i in range(1,a+1): |
|  | if a%i==0: |
|  | c+=1 |
|  |  |
|  | print(c) |

342

|  |
| --- |
| sum=0; |
|  | for x in range(0,100): |
|  | sum+= int(input()) |
|  |  |
|  | print(sum) |

343

|  |
| --- |
| a=int(input()) |
|  | sum=0 |
|  | for i in range(a): |
|  | sum+=int(input()) |
|  |  |
|  | print(sum) |

Обычно фразы break в pyton связаны с условными операторами.

edibles = ["отбивные", "пельмени","яйца","орехи"]

for food in edibles:

if food == "пельмени":

print("Я не ем пельмени!")

break

print("Отлично, вкусные " + food)

else:

print("Хорошо, что не было пельменей!")

print("Ужин окончен.")

Если мы запустим этот код, получим следующий результат:

Отлично, вкусные отбивные

Я не ем пельмени!

Ужин окончен.

Удалим “пельмени” из нашего списка материалов и получим следующее:

Отлично, вкусные отбивные

Отлично, вкусные яйца

Отлично, вкусные орехи

Хорошо, что не было пельменей!

Ужин окончен.

Предположим, отвращение к пельменям у нас не настолько велико, чтобы перестать потреблять другую пищу. Тогда это приводит к продолжению работы цикла с оператором continue. В следующем маленьком скрипте мы используем оператор continue, чтобы продолжить, итерацию по списку, когда мы сталкиваемся с пельменями. Поэтому продолжайте мешать нам потреблять спам!

edibles = ["отбивные", "пельмени","яйца","орехи"]

for food in edibles:

if food == "пельмени":

print("Я не ем пельмени!")

continue

print("Отлично, вкусные " + food)

# здесь может быть код для наслаждения едой:-)

else:

print("Ненавижу пельмени!")

print("Ужин окончен.")

Результат будет следующим:

Отлично, вкусные отбивные

Я не ем пельмени!

Отлично, вкусные яйца

Отлично, вкусные орехи

Ненавижу пельмени!

Ужин окончен.

## Итерация по спискам с функцией range()

Если вам нужно [**получить доступ к индексам списка**](https://pythonru.com/osnovy/spiski-python-funkcii-i-metody-python-list#toc-2), не очевидно как использовать цикл for для этой задачи. Мы можем получить доступ ко всем элементам, но индекс элемента остается недоступным. Есть способ получить доступ как к индексу элемента, так и к самому элементу. Для этого используйте [**функцию range()**](https://pythonru.com/osnovy/funkcija-range-v-python-3-rabota-s-ciklami-i-float) в сочетании с функцией длины len():

fibonacci = [0,1,1,2,3,5,8,13,21]

for i in range(len(fibonacci)):

print(i,fibonacci[i])

Вы получите следующий вывод:

0 0

1 1

2 1

3 2

4 3

5 5

6 8

7 13

8 21

**Примечание.** Если вы примените len() к list или tuple, получите соответствующее количество элементов этой последовательности.

## Подводные камни итераций по спискам

Если вы перебираете список, лучше избегать изменения списка в теле цикла. Чтобы наглядно увидеть, что может случиться, посмотрите на следующий пример:

colours = ["красный"]

for i in colours:

if i == "красный":

colours += ["черный"]

if i == "черный":

colours += ["белый"]

print(colours)

Что выведет print(colours)?

['красный', 'черный', 'белый']

Чтобы избежать этого, лучше всего работать с копией с помощью срезов, как сделано в следующем примере:

colours = ["красный"]

for i in colours[:]:

if i == "красный":

colours += ["черный"]

if i == "черный":

colours += ["белый"]

print(colours)

В результате вы получите следующее:

['красный', 'черный']

Мы изменили список colours, но данное изменение не повлияло на цикл. Элементы, которые должны быть итерированы, остаются неизменными во выполнения цикла.

## Enumerate в python 3

Enumerate — встроенная [**функция Python**](https://pythonru.com/osnovy/funkcii-v-python). Большинство новичков и даже некоторые продвинутые программисты не знают о ней. Она позволяет нам автоматически считать итерации цикла. Вот пример:

for counter, value in enumerate(some\_list):

print(counter, value)

Функция enumerate также принимает необязательный аргумент (значение начала отсчета, по умолчанию 0), который делает ее еще более полезной.

my\_list = ['яблоко', 'банан', 'вишня', 'персик']

for c, value in enumerate(my\_list, 1):

print(c, value)

# Результат:

# 1 яблоко

# 2 банан

# 3 вишня

# 4 персик

Все материалы по циклу for мы собрали в теме [**#цикл for**](https://pythonru.com/tag/cikl-for)