Цикл for

Цикл `for`, также называемый циклом с параметром, в языке Питон богат возможностями. В цикле `for` указывается переменная и множество значений, по которому будет пробегать переменная. Множество значений может быть задано списком, кортежем, строкой или диапазоном.
Вот пример использования цикла, где в качестве множества значений используется кортеж (набор констант):

```
for prime_number in 2, 3, 5, 7, 11, 13, 17, 19:
    print(prime_number)
```

В этом примере переменная `prime_number` последовательно принимает значения `2, 3, 5, 7, 11, 13, 17, 19`. На каждом шаге цикла выводится очередное простое число. Как и в условном операторе, для того чтобы указать операторы, которые должны выполняться на каждом шаге цикла, используется отступ относительно строки, начинающейся словом `for`. Рекомендуем в качестве отступа использовать 4 пробела, а не, например, табуляцию (она может по-разному выглядеть в различных текстовых редакторах).
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`.
Но это скорее нетипичное использование цикла `for`.

Функция `range`

Как правило, циклы `for` используются либо для повторения каких-либо 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 `for` 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. Под телом цикла понимается любой набор операторов (присваивания, печати, условных, цикла и др.) языка Python.
Например,

```
for i in range(10):
    print('Hello!')
```

Здесь слово `Hello!` будет напечатано 10 раз (скопируйте данную программу и проверьте, как это будет выглядеть).
Немного изменим нашу программу:

```
for i in range(10):
    print('Hello', i)
```

Теперь наши слова пронумерованы индексной переменной `i`, которая последовательно принимает значения 0, 1, …, 9. То есть в общем случае `n` означает количество повторений, а сами значения индексной переменной меняются от `0` до `n-1` включительно.
Если задать цикл таким образом:

```
for i in range(a, b):
    Тело цикла
```

то индексная переменная `i` будет 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 — первое значение, которая индексная переменная принимать ***не будет***. Если же `b` ≥ `a`, 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 к значению переменной `sum` последовательно добавляются указанные значения (операция += означает увеличение значения переменной на величину выражения, стоящего в правой части).

Усложним нашу программу. Пусть нам требуется найти минимум среди `n` введенных чисел (значение `n` вводится в программу ранее). Вспомним, как мы искали минимум среди трех чисел `a`, `b` и `c`:

```
min = a
if b < min:
    min = b
if c < min:
    min = c
```

Мы не можем считать `n` чисел в `n` различных переменных, но в нашем алгоритме они, по сути, не нужны: можно считывать очередное значение в одну и ту же переменную, сравнивать его с текущим значением минимума и забывать про него. Первое значение можно сразу считать в переменную `min`, тем самым задав ему начальное значение:

```
min = int(input())
for i in range(1, n):
    a = int(input())
    if a < min:
        min = a
print(min)
```

Обратите внимание, что в цикле значение переменной a считывается `n-1` раз, а не `n`.

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)`.

Более формально: цикл

```
for i in range(a, b, d)
```

при `d > 0` задает значения индексной переменной `i = a`, `i = a + d`, `i = a + 2 * d` и так для всех значений, для которых `i < b`. Если же `d < 0`, то переменная цикла принимает все значения `i > b`.

В качестве `a`, `b` или `d` можно использовать не только константы и переменные, но и выражения.

Задачи и указания к ним

A: 3529

B: 3530 Используйте операцию возведения в степень, для получения границ цикла.

C: 3532

D: 3533

E. 1370 В этой задаче решения с массивами (списками и другими структурами данных, для тех, кто их знает) будут проигнорированы.

F: 3536 Используйте операцию умножения строки на число (см. занятие 1) для печати первой, третьей и четвертой строк. Для печати второй строки используйте цикл `for`. Для того чтобы печать в цикле производилась в строку, а не в столбик, измените спецификатор `end` оператора печати. А после цикла для перевода строки перед печатью третьей строки используйте пустой `print()`. Следите за количеством пробелов между флажками. Также нужно помнить, что для печати символа '\' нужно использовать два таких символа подряд '\\', так как сам по себе он является служебным.

G: 3538 В этой задаче решения с использованием инструкции `if` будут проигнорированы.

H: 3541 В этой задаче решения с массивами и аналогичными структурами данных будут проигнорированы.

I: 3543 В этой задаче используется "информатический" подход к нахождению корней, а именно перебор всех возможных чисел из интересующего нас диапазона, и проверка, является ли очередное число корнем уравнения. Этот прием называется "перебор по ответу".

J: 3553 В этой задаче решения с массивами и аналогичными структурами данных будут проигнорированы.