Олимпиадное программирование
Занятие 3

Данил Браун

Весна 2024

Цикл for

Для чего может быть нужен цикл **`for`**?

Цикл for

Для чего может быть нужен цикл `for`?

(1) Нужно **повторить какой-то код несколько раз.**

Цикл for

Для чего может быть нужен цикл **for**?
(1) Нужно **повторить какой-то код несколько раз.**
Например, если нужно напечатать строку "Hello!" 5 раз, то можно сделать это с помощью цикла так (перед print обязательно сделать отступ в 4 пробела!):

```
for _ in range(5):
    print("Hello!")
```

Цикл for

Для чего может быть нужен цикл **for**?
(1) Нужно **повторить какой-то код несколько раз.**
Например, если нужно напечатать строку "Hello!" 5 раз, то можно сделать это с помощью цикла так (перед print обязательно сделать отступ в 4 пробела!):

```
for _ in range(5):
    print("Hello!")
```

Код выше делает то же самое, что и код ниже:

```
print("Hello!")
print("Hello!")
print("Hello!")
print("Hello!")
print("Hello!")
```

Цикл for

Для чего может быть нужен цикл **for**?
(1) Нужно **повторить какой-то код несколько раз.**
Например, если нужно напечатать строку `"Hello!"` 5 раз, то можно сделать это с помощью цикла так (перед `print` обязательно сделать отступ в 4 пробела!):

```
for _ in range(5):
    print("Hello!")
```

Код выше делает то же самое, что и код ниже:

```
print("Hello!")
print("Hello!")
print("Hello!")
print("Hello!")
print("Hello!")
```

Вместо 5 в `range` может быть указано другое число, равное нужному количеству повторений.

Цикл for

(2) Нужно выполнить один и тот же код несколько раз, **меняя в нём значение одной или нескольких переменных.**

Цикл for

(2) Нужно выполнить один и тот же код несколько раз, **меняя в нём значение одной или нескольких переменных.**

Допустим, мы хотим напечатать числа от 1 до 10. Сделать это с помощью цикла можно так:

```
for k in range(1, 11):
    print(k)
```

Цикл for

(2) Нужно выполнить один и тот же код несколько раз, **меняя в нём значение одной или нескольких переменных.**
Допустим, мы хотим напечатать числа от 1 до 10. Сделать это с помощью цикла можно так:

```
for k in range(1, 11):
    print(k)
```

Код выше эквивалентен коду ниже:

```
print(1)
print(2)
...
print(10)
```

Цикл for

(2) Нужно выполнить один и тот же код несколько раз, **меняя в нём значение одной или нескольких переменных.** Допустим, мы хотим напечатать числа от 1 до 10. Сделать это с помощью цикла можно так:

```
for k in range(1, 11):
    print(k)
```

Код выше эквивалентен коду ниже:

```
print(1)
print(2)
...
print(10)
```

Здесь после **for** появилась переменная `k` — она меняет свои значения от 1 (включительно) до 11 (исключительно). В предыдущем примере вместо переменной был написан знак подчёркивания _, потому что там у нас никакая перменная была не нужна, т.к. в коде ничего не менялось.

Цикл for

Ещё один пример. Вычисление суммы чисел от 0 до N:

```
N = int(input())
sum = 0
for k in range(N+1):
    sum = sum + k
print(sum)
```

Цикл for

Ещё один пример. Вычисление суммы чисел от 0 до N:

```
N = int(input())
sum = 0
for k in range(N+1):
    sum = sum + k
print(sum)
```

Можно проверить в оболочке для N = 10:

```
>>> print(1+2+3+4+5+6+7+8+9+10)
55
```

Цикл for

Ещё один пример. Вычисление суммы чисел от 0 до N:

```
N = int(input())
sum = 0
for k in range(N+1):
    sum = sum + k
print(sum)
```

Можно проверить в оболочке для N = 10:

```
>>> print(1+2+3+4+5+6+7+8+9+10)
55
```

Здесь `range(N+1)` означает то же самое, что и `range(0, N+1)`, т.е. если начинаем с нуля, то его можно не указывать.

Цикл for

Ещё один пример. Вычисление суммы чисел от 0 до N:

```
N = int(input())
sum = 0
for k in range(N+1):
    sum = sum + k
print(sum)
```

Можно проверить в оболочке для N = 10:

```
>>> print(1+2+3+4+5+6+7+8+9+10)
55
```

Здесь `range(N+1)` означает то же самое, что и `range(0, N+1)`, т.е. если начинаем с нуля, то его можно не указывать.

С каждой новой итерацией цикла переменная увеличивается на единицу, но мы можем указать другой шаг цикла (число, которое будет прибавляться каждый раз к переменной с каждой новой итерацией).

Цикл for

Пример с вычислением чётных степеней двойки. Переменная `p` пробегает значения от 0 (включительно) до 11 (исключительно) с шагом 2:

```
for p in range(0, 11, 2):
    print(p, 2**p)
```

Цикл for

Пример с вычислением чётных степеней двойки. Переменная `p` пробегает значения от 0 (включительно) до 11 (исключительно) с шагом 2:

```
for p in range(0, 11, 2):
    print(p, 2**p)
```

В результате получаем:

```
0 1
2 4
4 16
6 64
8 256
10 1024
```

Цикл for

Мы также можем указывать отрицательное число в качестве шага, но в этом случае необходимо правильно указать диапазон — первое число должно быть не меньше второго!

Цикл for

Мы также можем указывать отрицательное число в качестве шага, но в этом случае необходимо правильно указать диапазон — первое число должно быть не меньше второго! Следующий код выведет все степени двойки с десятой до нулевой:

```
for p in range(10, -1, -1):
    print(p, 2**p)
```

Аргументы sep и end для print

Функции `print` можно передавать не только то, что нужно напечатать, но и ещё два других **именованных** аргумента.

Аргументы sep и end для print

Функции `print` можно передавать не только то, что нужно напечатать, но и ещё два других **именованных** аргумента.

- `sep` означает строку, которая будет разделять (sep — separator) то, что Вы указали `print` через запятую. Например, `print(1, 2, 3)` напечатает три числа через пробел, а `print(1, 2, 3, sep=", ")` напечатает их через запятую с пробелом.

Аргументы sep и end для print

Функции `print` можно передавать не только то, что нужно напечатать, но и ещё два других **именованных** аргумента.

- `sep` означает строку, которая будет разделять (sep — separator) то, что Вы указали `print` через запятую. Например, `print(1, 2, 3)` напечатает три числа через пробел, а `print(1, 2, 3, sep=", ")` напечатает их через запятую с пробелом.
- `end` означает строку, которая будет добавляться в конец вместо переноса строки `"\n"`, который печатается всегда по умолчанию в конце строки при использовании `print`:

```
for a in range(10):
    # print(a)
    print(a, end=" ")
```