СОДЕРЖАНИЕ

Цикл for в Python

Циклы — один из важнейших инструментов в программировании. Цикл **for** позволяет выполнить определённый блок кода несколько раз, перебирая элементы какой-либо последовательности (списка, строки, диапазона чисел и т.д.). В отличие от цикла while, который используется, когда заранее неизвестно, сколько итераций потребуется (например, мы моем посуду, пока она не закончится), цикл for подходит, когда известно количество шагов или итераций (например, нужно помыть ровно пять тарелок). Если задача требует обработки каждой части чего-то определённого, например, каждый элемент списка или каждую букву в строке, цикл for — это удобный инструмент.

Синтаксис цикла for

Цикл for в Python имеет следующий синтаксис:

```
for переменная in последовательность:
    блок_кода
```

• Переменная принимает значение каждого элемента последовательности на каждой итерации.

• Последовательность может быть любой коллекцией данных (список, строка, диапазон).

• Блок кода выполняется на каждой итерации.

Пример:

```
for i in range(5):
    print(i)
```

Этот цикл выведет числа от 0 до 4.

Функция range()

Функция **range()** создаёт последовательность чисел, которая используется в цикле for. Она может принимать от 1 до 3 параметров:

- range(n) — возвращает числа от 0 до n-1.
- range(start, end) — возвращает числа от start до end-1.
- range(start, end, step) — возвращает числа от start до end-1 с шагом step.

Если шаг не указан, он по умолчанию равен 1. При этом важно, чтобы первый аргумент был меньше второго, иначе последовательность будет пустой.

Примеры:

```
for i in range(3):
    print(i)  # 0, 1, 2
```

```
for i in range(1, 6):
    print(i)  # 1, 2, 3, 4, 5
```

```
for i in range(1, 10, 2):
    print(i)  # 1, 3, 5, 7, 9
```

Функцию range() также удобно использовать, когда перед вами, например, стоит задача вывести все числа от 1 до 100, кратные 3. При исходном решении программа выглядела бы так:

```
for i in range(1, 101):
    if i % 3 == 0:
        print(i)
```

Но можно к решению этой задачи подойти несколько иначе. Наименьшее число из диапазона от 1 до 100, кратное 3, это и есть само число 3. Тогда, при создании функции range() можно указать 3, как начальное число, конечное число оставить прежним, а шаг счётчик сделать равным трём. В таком случае наша программа приобретает вид:

```
for i in range(3, 101, 3):
    print(i)
```

Таким образом, мы не только сокращаем программный код, но и оптимизируем его, так как используем не полный перебор всех чисел из диапазона, а сразу выводим нужный ответ.

Перебор строки

В Python строки — это неизменяемые последовательности символов, и их можно перебирать с помощью цикла for.

Пример 1: Перебор строки с индексом:

```
string = "hello"
for i in range(len(string)):
    print(string[i])
```

Пример 2: Перебор строки без индексов:

```
string = "hello"
for char in string:
    print(char)
```

Оба примера дают одинаковый результат, но если вам не нужны индексы, то проще использовать второй вариант.

Важное замечание: строки в Python неизменяемы, поэтому нельзя изменить символ в строке, обращаясь к нему через индекс. Например, запись string[0] = 'H' вызовет ошибку.

Функция enumerate()

Иногда в цикле нужно одновременно и перебирать элементы, и получать их индекс. Для этого используется функция enumerate(), которая возвращает пару (индекс, элемент).

Пример:

```
string = "hello"
for index, char in enumerate(string):
    print(index, char)
```

Это очень удобно, когда требуется работать с индексами и элементами одновременно.

Изменение счётчика в цикле for

Цикл for в Python автоматически обновляет значение переменной на каждой итерации. Не стоит пытаться вручную изменить значение переменной, это нарушит логику работы цикла. Если возникает необходимость пропустить какое-то значение, можно использовать оператор continue, который пропускает текущую итерацию и переходит к следующей. А если нужно полностью выйти из цикла, можно использовать break.

То есть, если вы, например, хотите вывести все числа от 1 до 5, но при этом игнорируя 2, не стоит писать так:

```
for i in range(1, 6):
    if i == 2:
        i += 1
    print(i, end=' ')
# получите 1 3 3 4 5
```

Есть различные способы, как решить эту задачу:

```
for i in range(1, 6):
    if i != 2:
        print(i, end=' ')
# получите 1 3 4 5
```

```
for i in range(1, 6):
    if i == 2:
        continue
    print(i, end=' ')
# получите 1 3 4 5
```