# СОДЕРЖАНИЕ

**Для чего нужно считывать данные в одну строку**

Часто задачи требуют ввода нескольких значений от пользователя одновременно. Это удобно, когда нужно обработать массив данных, например, список чисел или слов. В таких случаях считывание данных в одну строку позволяет сократить количество вводов и упростить код. Представьте, что вы создаёте программу для вычисления среднего балла студента. Вместо того, чтобы вводить каждую оценку по отдельности, можно ввести все значения в одной строке и затем обработать их с помощью методов, которые облегчат дальнейшие вычисления.

**Что такое метод split() и как он работает**

Метод **split()** используется для разделения строки на отдельные элементы по указанному разделителю и возвращает их в виде списка. Он принимает необязательный аргумент — символ-разделитель. Если разделитель не указан, split() по умолчанию удаляет любые пробелы (включая несколько подряд и другие пробельные символы).

```
data = "яблоко,банан,апельсин"
fruits = data.split(",")  # разделяем строку по запятой
print(fruits)  # ['яблоко', 'банан', 'апельсин']
```

Если аргумент не передан:

```
data = "   одно   два  три  "
words = data.split()  # убираем все лишние пробелы
print(words)  # ['одно', 'два', 'три']
```

### Что возвращает метод split() и в каком виде хранятся данные

Метод split() возвращает список строк, которые были разделены по указанному или стандартному разделителю. Эти строки всегда остаются в текстовом формате (тип str), даже если они содержат числа. Вот пример:

```
data = "12 45 78"
numbers = data.split()  # получаем список строк ['12', '45', '78']
print(numbers)  # ['12', '45', '78']
```

### Работа со списком строк: конвертация строк в числа

Когда мы считываем числа в одну строку, они хранятся как строки. Для того чтобы использовать их в арифметических операциях, нужно преобразовать каждый элемент списка в целое число с помощью функции int().

Чтобы быстро преобразовать строки в числа, можно использовать **списочное выражение**. Это короткий и эффективный способ создания списков. Пример списочного выражения:

```
data = "12 45 78"
numbers = [int(num) for num in data.split()]
print(numbers)  # [12, 45, 78]
```

Здесь мы сразу применяем функцию int() ко всем элементам списка, полученного с помощью метода split().

### Что такое списочное выражение и для чего оно нужно

Списочное выражение — это способ создать новый список, основываясь на уже существующем. Оно помогает сократить запись циклов и делает код

более читабельным. Вместо стандартного цикла for, где мы создаём список и добавляем в него элементы, списочные выражения позволяют сделать это в одной строке. Они особенно полезны, когда нужно применить одно и то же действие ко всем элементам списка.

**Синтаксис списочного выражения**

Синтаксис списочного выражения следующий:

```
[выражение for элемент in последовательность]
```

• выражение — операция, которую нужно применить к каждому элементу;

• элемент — каждый элемент из последовательности (например, списка);

• последовательность — исходный набор данных.

Пример:

```
squares = [x**2 for x in range(5)]  # создаём список квадратов чисел от 0 до 4
print(squares)  # [0, 1, 4, 9, 16]
```

**Подробное объяснение элементов списочного выражения**

Предположим, что у нас есть стандартный цикл for:

```
numbers = []
for i in range(5):
    numbers.append(i * 2)
```

Его можно переписать в виде списочного выражения:

```
numbers = [i * 2 for i in range(5)]
```

• i * 2 — выражение, которое выполняется для каждого элемента;

• for i in range(5) — цикл, который перебирает значения.

То есть, всё, что было в теле цикла, перемещается в начало списочного выражения.

**Условное выражение в списочном выражении**

Списочные выражения также поддерживают условные выражения if, позволяя добавлять в новый список только те элементы, которые удовлетворяют условию:

```
even_numbers = [x for x in range(10) if x % 2 == 0]
print(even_numbers)  # [0, 2, 4, 6, 8]
```

Здесь в новый список добавляются только чётные числа. Важно: в списочном выражении можно использовать условие if, но без else.

**Пример: нахождение суммы чётных чисел**

Рассмотрим задачу: вводится строка с целыми числами, необходимо найти сумму всех чётных чисел.

```
data = input("Введите числа через пробел: ")  # пример ввода: 1 2 3 4 5 6
numbers = [int(num) for num in data.split()]  # преобразуем строку в список чисел
even_sum = sum([num for num in numbers if num % 2 == 0])  # суммируем только чётные
print("Сумма чётных чисел:", even_sum)
```

Разбор:

- Используем split(), чтобы получить список строк;
- Применяем int(), чтобы преобразовать строки в числа;
- Далее с помощью списочного выражения оставляем только чётные числа;
- Используем функцию sum() для нахождения суммы чётных чисел.