Строки. Арифметические выражения. Eval()

В Python математические выражения могут быть представлены как строки.

Примеры:

```
expression1 = "2 + 3 * 5"
expression2 = "10 / 2 - 3"
expression3 = "(4 + 6) * (2 - 1)"
```

Такие строки можно использовать для хранения выражений, которые будут вычисляться позже.

Функция eval()

Функция **eval()** используется для вычисления строки как кода Python.

В контексте арифметических выражений это означает вычисление результата математической операции, представленной строкой. **Синтаксис:**

```
result = eval(expression)
```

*где, где expression — строка, содержащая корректное Python-выражение.

Пример №1. Простое математическое выражение

```
expression = "2 + 3 * 5"
result = eval(expression)
print(result)
```

Вывод >>> 17

Пример №2. Выражение со скобками

```
expression = "(4 + 6) * (2 - 1)"
result = eval(expression)
print(result)
```

Вывод >>> 10

Пример №3. Строковые выражения могут быть созданы динамически в зависимости от условий. Это позволяет использовать строки для построения сложных вычислений.

```
a = 5
b = 10
operation = "+"
expression = f"{a} {operation} {b}"
result = eval(expression)
print(result)
```

Вывод >>> 15

Математическая составляющая строковых выражений

eval() учитывает стандартные математические правила порядка выполнения операций:

1. Скобки () — наивысший приоритет.
2. Унарные операторы, такие как отрицание - (например, -3).
3. Умножение *, деление /, целочисленное деление //, остаток %.
4. Сложение + и вычитание -.

Производительность и асимптотика eval()

Производительность **eval()** зависит от сложности строки. Общая оценка:

- **O(n):** Простые арифметические выражения (a + b, a * b).
- **O(log(n))** или хуже: Выражения с использованием функций вроде логарифмов (math.log) или операций со степенями (a**b).
- **O(n^2):** Комплексные выражения с большим количеством скобок или вложений (парсинг строки замедляется).

Заметки