Изучение операторов и выражений в Python: арифметические, логические и сравнительные операции

Python предоставляет разнообразные операторы для работы с данными и переменными. Понимание этих операторов критически важно для эффективного программирования на Python. В этой статье мы подробно рассмотрим все типы операторов и выражений в Python с практическими примерами.

Арифметические операторы Python

Арифметические операторы выполняют математические вычисления над числовыми значениями.

Арифметический оператор	Знак	Описание
Сложение	+	Складывает два числа
Вычитание	-	Вычитает второе число из первого
Умножение	*	Умножает два числа
Деление	/	Делит первое число на второе (результат float)
Остаток от деления	%	Возвращает остаток от деления
Целочисленное деление	//	Возвращает целую часть от деления
Возведение в степень	**	Возводит первое число в степень второго

```
a = 10
b = 3

sum_result = a + b  # 13
difference_result = a - b  # 7
product_result = a * b  # 30
division_result = a / b  # 3.3333333333333335
remainder_result = a % b  # 1
integer_division_result = a // b  # 3
power_result = a ** b  # 1000
```

Операторы сравнения в Python

Операторы сравнения используются для сравнения значений и возвращают логические значения True или False.

Оператор	Знак	Описание
Равно	==	Проверяет равенство значений
Не равно	!=	Проверяет неравенство значений
Больше	>	Проверяет, больше ли первое значение второго
Меньше	<	Проверяет, меньше ли первое значение второго
Больше или равно	>=	Проверяет, больше или равно ли первое значение второму
Меньше или равно	<=	Проверяет, меньше или равно ли первое значение второму

```
x = 5
y = 10

equal_result = x == y  # False
not_equal_result = x != y  # True
greater_than_result = x > y  # False
less_than_result = x < y  # True
greater_than_or_equal_result = x >= y  # False
less_than_or_equal_result = x <= y  # True
```

Логические операторы Python

Логические операторы используются для объединения условных выражений.

Логический оператор	Слово	Описание
Логическое И	and	Возвращает True, если оба условия истинны
Логическое ИЛИ	or	Возвращает True, если хотя бы одно условие истинно
Логическое НЕ	not	Инвертирует логическое значение

```python
a = True
b = False

and_result = a and b  # False
or_result = a or b  # True
not_result = not a  # False

# Практический пример
age = 25
has_license = True

can_drive = age >= 18 and has_license  # True
```

Операторы присваивания Python

Операторы присваивания используются для присвоения значений переменным и выполнения операций одновременно.

Оператор	Знак	Описание
Присваивание	=	Присваивает значение переменной
Сложение с присваиванием	+=	Прибавляет значение к переменной
Вычитание с присваиванием	-=	Вычитает значение из переменной
Умножение с присваиванием	*=	Умножает переменную на значение
Деление с присваиванием	/=	Делит переменную на значение
Остаток от деления с присваиванием	%=	Присваивает остаток от деления
Целочисленное деление с присваиванием	//=	Присваивает результат целочисленного деления
Возведение в степень с присваиванием	**=	Возводит переменную в степень

```
x = 5
x += 3  # x = x + 3, теперь x равен 8
x -= 2  # x = x - 2, теперь x равен 6
x *= 2  # x = x * 2, теперь x равен 12
x /= 4  # x = x / 4, теперь x равен 3.0
x %= 2  # x = x % 2, теперь x равен 1.0
```

Операторы принадлежности и тождественности

Эти операторы проверяют принадлежность элементов к последовательностям и тождественность объектов.

Оператор	Слово	Описание
Принадлежит	in	Возвращает True, если элемент присутствует в последовательности
Не принадлежит	not in	Возвращает True, если элемент отсутствует в последовательности
Тождественно	is	Возвращает True, если оба операнда указывают на один объект
Не тождественно	is not	Возвращает True, если операнды указывают на разные объекты

```
my_list = [1, 2, 3, 4, 5]
x = 2

in_result = x in my_list  # True
not_in_result = x not in my_list  # False

# Пример с оператором is
a = [1, 2, 3]
b = a
c = [1, 2, 3]

print(a is b)  # True (один объект)
print(a is c)  # False (разные объекты)
print(a == c)  # True (одинаковые значения)
```

Побитовые операторы Python

Побитовые операторы работают с двоичным представлением чисел.

Оператор	Знак	Описание
Побитовое И	&	Побитовая операция И
Побитовое ИЛИ	|	Побитовая операция ИЛИ
Побитовое исключающее ИЛИ	^	Побитовая операция исключающее ИЛИ
Побитовое отрицание	~	Побитовая инверсия
Сдвиг влево	<<	Сдвигает биты влево
Сдвиг вправо	>>	Сдвигает биты вправо

```
a = 5   # 101 в двоичной системе
b = 3   # 011 в двоичной системе

print(a & b)  # 1 (001)
print(a | b)  # 7 (111)
print(a ^ b)  # 6 (110)
print(~a)     # -6
print(a << 1) # 10 (1010)
print(a >> 1) # 2 (10)
```

Выражения в Python: примеры и объяснения

Выражения в Python представляют собой комбинации операторов и операндов, которые вычисляются интерпретатором для получения значения.

1. Арифметические выражения

```
a = 2 + 3 * 4  # Результат: 14 (сначала умножение, затем сложение)
b = (2 + 3) * 4  # Результат: 20 (сначала скобки)
```

2. Строковые выражения

```
greeting = 'Hello' + ' ' + 'world'  # "Hello world"
name = "Python"
message = f"Изучаем {name}!"  # "Изучаем Python!"
```

3. Логические выражения

```
x = 6
is_valid = x > 5 and x < 10  # True
is_even = x % 2 == 0  # True
```

4. Выражения сравнения

```
a = 3
b = 3
are_equal = a == b  # True
is_greater = a > b  # False
```

5. Выражения доступа к элементам

```
my_list = [1, 2, 3, 4, 5]
first_element = my_list[0]  # 1
last_element = my_list[-1]  # 5

my_dict = {'name': 'Alice', 'age': 30}
name = my_dict['name']  # 'Alice'
```

6. Выражения вызова функций

```
numbers = [1, 2, 3, 4, 5]
length = len(numbers)  # 5
maximum = max(numbers)  # 5
total = sum(numbers)  # 15
```

7. Выражения-генераторы списков

```
squares = [x**2 for x in range(5)]  # [0, 1, 4, 9, 16]
even_numbers = [x for x in range(10) if x % 2 == 0]  # [0, 2, 4, 6, 8]
```

8. Условные выражения (тернарный оператор)

```
x = -5
result = x if x > 0 else -x  # 5 (абсолютное значение)
status = "adult" if age >= 18 else "minor"
```

9. Выражения генерации кортежей

```
coordinates = [(x, y) for x in range(3) for y in range(3)]
# [(0, 0), (0, 1), (0, 2), (1, 0), (1, 1), (1, 2), (2, 0), (2, 1), (2, 2)]
```

10. Выражения атрибутов

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

person = Person("Alice", 30)
name = person.name  # "Alice"
age = person.age  # 30
```

11. Выражения срезов

```
my_list = [10, 20, 30, 40, 50]
sub_list = my_list[1:4]  # [20, 30, 40]
reversed_list = my_list[::-1]  # [50, 40, 30, 20, 10]
```

12. Выражения с лямбда-функциями

```
square = lambda x: x**2
result = square(5)  # 25

numbers = [1, 2, 3, 4, 5]
squared_numbers = list(map(lambda x: x**2, numbers))  # [1, 4, 9, 16, 25]
```

Приоритет операторов в Python

При работе с выражениями важно понимать приоритет операторов:

1. Скобки ()
2. Возведение в степень **
3. Унарные операторы +x, -x, ~x
4. Умножение, деление *, /, //, %
5. Сложение, вычитание +, -
6. Побитовые сдвиги <<, >>
7. Побитовое И &
8. Побитовое исключающее ИЛИ ^
9. Побитовое ИЛИ |
10. Сравнения ==, !=, <, <=, >, >=, is, is not, in, not in
11. Логическое НЕ not
12. Логическое И and
13. Логическое ИЛИ or