IT CAREER HUB

Урок 3.
Арифметические операторы, выражения

IT CAREER HUB

Числа и арифметические операции с числами

В Python числа представлены типами:

- **целые числа (int)**
- **числа с плавающей точкой или вещественные (float)**

Python поддерживает стандартные арифметические операции, такие как сложение, вычитание, умножение и деление. Эти операции могут быть выполнены с числами (целыми или вещественными) и переменными, содержащими числовые значения.

Операция	**Название**	**Описание**
+	Сложение	Складывает два значения.
-	Вычитание	Вычитает одно значение из другого.
*	Умножение	Умножает два значения.
/	Деление	Делит одно значение на другое.
//	Целочисленное деление	Делит одно значение на другое, возвращая целую часть результата.
%	Остаток от деления	Возвращает остаток от деления двух значений.
**	Возведение в степень	Возводит значение в указанную степень.

Важно:

- При обычном делении результат всегда является дробным числом.
- Если в выражении хотя бы одна часть типа float, то результат будет тоже float.
- Результат целочисленного деления всегда округляется в меньшую сторону.

ICH IT CAREER HUB

Примеры использования

```python
# Сложение

print("Сложение:", "3 + 5 =", 3 + 5)

# Вычитание

print("Вычитание:", "7 - 2 =", 7 - 2)

# Умножение

print("Умножение:", "4 * 6 =", 4 * 6.0)

# Деление

print("Деление:", "8 / 2 =", 8 / 2)

# Целочисленное деление

print("Целочисленное деление:", "7 // 2 =", 7 // 2.0)

# Остаток от деления

print("Остаток от деления:", "7 % 3 =", 7 % 3)

# Возведение в степень

print("Возведение в степень:", "2 ** 3 =", 2 ** 3)
```

Важно:

- Если какой-то результат в последующем нужно использовать повторно, то можно заключить его в переменную.

```python
sum1 = 3 + 5
a = sum1 + 1
```

```
b = sum1 * 3
```

- Если какое-то действие нужно совершить один раз, то хранить информацию в переменной бессмысленно.

```python
print(3 + 5)
```

ZeroDivisionError

ZeroDivisionError — это исключение, которое возникает когда происходит попытка деления на ноль. Это одна из встроенных ошибок в Python, и она выбрасывается при попытке выполнить деление, где делитель равен нулю, как для целых чисел, так и для вещественных.

Когда возникает ZeroDivisionError?

- При делении целых чисел на ноль:
- При делении вещественных чисел на ноль:
- При использовании оператора остатка от деления на ноль:

Пример

```python
a = 10
b = 0
result = a / b
```

В данном примере программа выполняет деление на ноль, что приводит к исключению **ZeroDivisionError** и аварийному завершению программы.

Приоритет математических операций и скобок

В Python, как и в математике, операции выполняются в определённом порядке. Это называется приоритетом операций. Если вы не зададите порядок явно с помощью скобок, Python будет следовать установленным правилам приоритета.

Правила приоритета операций:

1. **Скобки ()**: Операции внутри скобок всегда выполняются первыми, независимо от их приоритета. Это позволяет явно задать порядок выполнения операций.

Пример

```python
result = (2 + 3) * 4  # Скобки заставляют сложить 2 и 3 перед умножением

print(result)         # Результат: 20
```

2. **Возведение в степень (**)**: После скобок, возведение в степень имеет наивысший приоритет.

Пример

```python
result = 2 * 3 ** 2   # Это интерпретируется как 2 * (3 ** 2)
print(result)         # Результат: 18
```

3. **Умножение (*), Деление (/), Целочисленное деление (//), Остаток от деления (%)**: Эти операции имеют одинаковый приоритет и выполняются слева направо.

Пример

```python
result = 10 / 2 * 3   # Интерпретируется как (10 / 2) * 3
print(result)         # Результат: 15.0
```

4. **Сложение (+) и Вычитание (-)**: Они имеют самый низкий приоритет и также выполняются слева направо.

Пример

```python
result = 10 - 2 + 5   # Интерпретируется как (10 - 2) + 5
print(result)         # Результат: 13
```

Порядок приоритета операций (от высшего к низшему):

Приоритет	Операция	Символы
1	Скобки	()
2	Возведение в степень	**
3	Умножение, Деление, Целочисленное деление, Остаток от деления	*, /, //, %
4	Сложение и Вычитание	+, -

IT CAREER HUB

Задание для закрепления 1

1. Какой результат выведет следующий код?

```python
print(5 + 3 * 2)
```

a. 16
b. 11
c. 13
d. 10

Посмотреть ответ

2. Какой результат выведет следующий код?

```python
print(11 % 4)
```

a. 1
b. 2
c. 3
d. 4

Посмотреть ответ

3. Какой результат выведет следующий код?

```python
print(2 * 3 ** 2)
```

a. 12
b. 36
c. 18
d. 9

Посмотреть ответ

4. Какой результат выведет следующий код?

```python
print(7 // 0)
```

a. 0
b. 7
c. Ошибка ZeroDivisionError
d. 1

Посмотреть ответ

IT CAREER HUB

Неизменяемые типы данных

В Python существуют неизменяемые (immutable) и изменяемые (mutable) типы данных. Неизменяемые типы данных не могут быть изменены после создания объекта, а изменяемые могут.

Если у нас есть переменная num равная 5, то мы не можем изменить значение этой переменной, например с помощью арифметической операции:

```python
num = 5

num + 2             # В переменной num останется значение 5

print(num)
```

Для того чтобы изменить значение в num нужно присвоить переменной новое значение.

```python
num = 5
num = num + 2  # В переменную num будет присвоен результат вычисления 5 + 2
print(num)
```

IT CAREER HUB

Операторы приращения

Операторы приращения — это операторы, которые изменяют значение переменной на основе её текущего значения. Для этого используются операторы присваивания в сочетании с арифметическими операциями, такими как сложение, вычитание, умножение и другие.

Оператор	Описание	Пример	Эквивалент
+=	Прибавление и присваивание	a += 1	a = a + 1
-=	Вычитание и присваивание	a -= 1	a = a - 1
*=	Умножение и присваивание	a *= 2	a = a * 2
/=	Деление и присваивание	a /= 2	a = a / 2
//=	Целочисленное деление и присваивание	a //= 2	a = a // 2
%=	Остаток от деления и присваивание	a %= 3	a = a % 3
**=	Возведение в степень и присваивание	a **= 2	a = a ** 2

Операторы приращения — это более краткая и удобная форма записи операций присваивания (a += 1). Они могут быть заменены на более полные выражения вида a = a + 1, но их использование делает код более компактным и понятным.

Примеры

1. **Прибавление (+=)**

```python
a = 5
a += 1  # Эквивалентно: a = a + 1
print(a)  # Результат: 6
```

2. **Умножение (*=)**

```python
a = 5
a *= 2 # Эквивалентно: a = a * 2
print(a)  # Результат: 10
```

Множественное присваивание

Множественное присваивание — это особенность Python, которая позволяет присваивать значения нескольким переменным одновременно в одной строке. Это делает код более компактным и читабельным, особенно когда требуется инициализировать сразу несколько переменных.

Примеры

1. **Присваивание нескольких значений:**

```python
a, b, c = 1, 2, 3  # Инициализируем переменные a, b, c значениями 1, 2, 3 соответственно
print(a, b, c)     # Вывод: 1 2 3
```

2. **Присваивание одинакового значения нескольким переменным:**

```python
a = b = c = 0      # Инициализируем каждую из переменных a, b, c значением 0
print(a, b, c)     # Вывод: 0 0 0
```

3. **Обмен значениями между переменными:** Множественное присваивание можно использовать для обмена значениями без необходимости использования временной переменной.

```python
a, b = 5, 10
a, b = b, a  # Меняем местами значения a и b
```

```
print(a, b)  # Вывод: 10 5
```

Задание для закрепления 2

1. Какое значение будет у переменной num после выполнения следующего кода?

```python
num = 5
num + 3
print(num)
```

a. 8
b. 5
c. Ошибка
d. 3

Посмотреть ответ

2. Какое значение будет у переменных a и b после выполнения следующего кода?

```python
a, b = 1, 2
a, b = b, a
print(a, b)
```

a. 1, 2
b. 2, 1
c. 3, 3
d. Произойдет ошибка

Посмотреть ответ

IT CAREER HUB

Преобразование типов

Преобразование (приведение) типов — это процесс преобразования значения одного типа данных в значение другого типа. Это может быть необходимо, когда нужно выполнить определенные операции со значением или значениями разных типов. Например, невозможно произвести математические операции со строковыми данными, но можно с числовыми.

Пример

```python
num1 = "10"
num2 = "5.0"

print(num1 / num2)  # Попытка выполнить деление строк
```

При попытке выполнить деление num1 на num2, Python вызывает ошибку TypeError, поскольку оператор / не поддерживается для строк. Для успешного выполнения операции необходимо привести строки к числовому типу (например, к int или float):

Исправленный пример

```python
num1 = "10"
num2 = "5.0"

result = int(num1) / float(num2)     # Приведение строк к числовым типам и выполнение деления
print(result)                        # Вывод: 2.0
```

При **неявном преобразовании** Python может автоматически преобразовывать некоторые типы данных в других контекстах, например, при выполнении арифметических операций между числами разных типов.

```python
result = 5
print(type(result))   # <class 'int'>
result += 3.14        # Неявное преобразование int в float
print(type(result))   # <class 'float'>
```

Явное преобразование происходит, когда разработчик вручную преобразует один тип данных в другой с помощью встроенных функций. Это необходимо, если Python не может выполнить преобразование автоматически или если нужно строго контролировать тип данных.

ICH IT CAREER HUB

Функции для явного преобразования в примитивные типы данных

Функция	Описание	Пример использования
`int()`	Преобразует в целое число	`int("10")`
`float()`	Преобразует в вещественное число	`float("10.5")`
`str()`	Преобразует в строку	`str(10)`
`bool()`	Преобразует в логический тип	`bool(1)`

Приведение типов осуществляется с помощью встроенных функций, таких как `int()`, `float()`, `str()`, и т. д. Названия функций соответствует названию типов данных.

Примеры

1. **Целое число в строку:**

```python
a = 5
b = str(a)      # Преобразуем целое число в строку
print(b)        # Вывод: '5'
print(type(b))  # Вывод: <class 'str'>
```

2. **Строка в число:**

```python
a = "10"
b = int(a)      # Преобразуем строку в целое число
print(b)        # Вывод: 10
```

```
print(type(b))  # Вывод: <class 'int'>
```

3. **Вещественное число в целое:**

```python
a = 10.7
b = int(a)  # Преобразуем вещественное число в целое
print(b)  # Вывод: 10 (дробная часть отбрасывается)
```

Особенности и ошибки преобразования:

1. **Не все типы могут быть преобразованы друг в друга.** Например, строка "abc" не может быть преобразована в число.

```python
a = "abc"


b = int(a)  # Ошибка: ValueError
```

2. **При преобразовании вещественных чисел в целые дробная часть отбрасывается.**

```python
a = 5.99
b = int(a)
print(b)  # Вывод: 5
```

IT CAREER HUB

ValueError

ValueError — это встроенное исключение в Python, которое возникает, когда функция получает аргумент правильного типа, но с некорректным значением. Это ошибка, которая указывает на то, что переданное значение не подходит для выполнения операции, хотя тип данных сам по себе допустим.

Когда возникает ValueError? Например, если вы попытаетесь преобразовать строку, которая не может быть интерпретирована как число, в целое или вещественное число, будет вызвано исключение **ValueError**.

Пример

```python
a = "abc"
b = float(a)  # Ошибка: ValueError: invalid literal for float() with base 10: 'abc'
```

IT CAREER HUB

Задание для закрепления 3

1. Что произойдёт при выполнении следующего кода?

```python
result = 5
result += 3.14
print(type(result))
```

a. <class 'int'>
b. <class 'float'>
c. Ошибка
d. <class 'str'>

Посмотреть ответ

2. Соотнесите функцию для преобразования типа с её результатом:

1. int("10.0")	a. "10"
2. float("3.14")	b. 3.14
3. str(10)	c. False
4. bool(0)	d. Ошибка

Посмотреть ответ

IT CAREER HUB

Ответы на задания

Задания на закрепление 1	Вернуться к заданиям
1. Результат выполнения кода	Ответ: b
2. Результат выполнения кода	Ответ: c
3. Результат выполнения кода	Ответ: c
4. Результат выполнения кода	Ответ: c
Задания на закрепление 2	Вернуться к заданиям
1. Переменная `num`	Ответ: b
2. Переменные a и b	Ответ: b
Задания на закрепление 3	Вернуться к заданиям
1. Результат выполнения кода	Ответ: b
2. Функция и ее результат	Ответ: 1 - d, 2 - b, 3 - a, 4 - c

IT CAREER HUB

Практическая работа

1. Определите математические операции

Напишите программу, которая имеет две переменные a = 7, b = 3. Выполните математические операции, которые в результате дадут числа 4, 1, 2 (по одной операции для значения). Выведите на экран результат.

Пример вывода:

Результат операции <название>: 4

Результат операции <название>: 1

Результат операции <название>: 2

Решение:

```python
a = 7
b = 3

print("Результат операции 'Разность':", a - b)
print("Результат операции 'Остаток от деления':", a % b)
print("Результат операции 'Целочисленное деление':", a // b)
```

2. Расчёт сдачи

Напишите программу, которая принимает от пользователя стоимость товара, количество и сумму, которую он заплатил. Программа должна рассчитать и вывести сдачу.

Пример вывода:

Введите стоимость товара: 60
Введите количество товара: 3
Введите сумму оплаты: 500
Сдача: 320 рублей

Решение:

```python
price = int(input("Введите стоимость товара: "))
quantity = int(input("Введите количество товара: "))
payment = int(input("Введите сумму оплаты: "))

total_cost = price * quantity
change = payment - total_cost

print("Сдача:", change, "рублей")
```