Занятие 18. Основные операции с числовыми типами данных

План занятия

1. Понятие операции 2

1.1. Арифметические операции 2

1.2. Приоритет операций 2

1.3. Арифметические операции с присвоением 3

2. Логические значения 4

3. Функции преобразования чисел 4

3.1. Представление числа 6

3.2. Функция format 6

1. ПОНЯТИЕ ОПЕРАЦИИ

1.1. Арифметические операции

Python поддерживает все распространенные арифметические операции:

•+	Сложение двух чисел:	print(6 +) = 8
•-	Вычитание двух чисел:	print(6 -) = 4
•*	Умножение двух чисел:	print(6 *) = 1
•/	Деление двух чисел:	print(6 /) = 3.0
•//	Целочисленное деление двух чисел:	print(7 /) = 3.5 print(7 //) = 3 Данная операция возвращает целочисленный результат деления, отбрасывая дробную часть
•**	Возведение в степень:	print(6 **) = Возводим число 6 в степень . Результат - 36
•%	Получение остатка от деления:	print(7 % 2) = Получение остатка от деления числа 7 на 2. Результат - 1

Кроме того, в Python для операций с числами используются функции abs() (вычисление абсолютного значения - модуля, abs(−3) = 3), pow() (возведение в степень, pow(2,3) = 8), divmod() (вычисление результата целочисленного деления и остатка, divmod(17,5) = (3,2)) и round() (округление, round(100.0/6) = 17.0).

Эти функции являются «встроенными», что означает, что для их использования нет необходимости подключать дополнительные модули. Все прочие функции для работы с числами (математические), такие как вычисление квадратного корня, синуса и пр. требуют подключения модуля math:

```
import math
```

1.2. Приоритет операций

При последовательном использовании нескольких арифметических операций их выполнение производится в соответствии с их приоритетом. В

начале выполняются операции с большим приоритетом. Приоритеты операций в порядке убывания приведены в следующей таблице.

Операции	Направление
**	Слева направо
* / // %	Слева направо
+ -	Слева направо

Пусть у нас выполняется следующее выражение:

number = 3 + 4 * 5 ** + 7

print(number) = 110

Здесь начале выполняется возведение в степень (5 **) как операция с большим приоритетом, далее результат умножается на 4 (5 * 4), затем происходит сложение (3 + 100) и далее опять идет сложение (103 + 7).

Чтобы переопределить порядок операций, можно использовать скобки:

number = (3 + 4) * (5 ** + 7)

print(number) = 4

Следует отметить, что в арифметических операциях могут принимать участие как целые, так и дробные числа. Если в одной операции участвует целое число (int) и число с плавающей точкой (float), то целое число приводится к типу float.

1.3. Арифметические операции с присвоением

Ряд специальных операций позволяют использовать присвоить результат операции первому операнду:

•+=	Присвоение результата сложения
•-=	Присвоение результата вычитания
•*=	Присвоение результата умножения
•/=	Присвоение результата от деления
•//=	Присвоение результата целочисленного деления
•**=	Присвоение степени числа
•%=	Присвоение остатка от деления

В Python разрешены "цепочки" присваиваний и сравнений, однако присваивания и сравнения нельзя смешивать. Выражения a=b=c=4 и a<b<5 допустимы, а выражение a<b=4 недопустимо. Однако допустимо выражение a<b==4.

2. ЛОГИЧЕСКИЕ ЗНАЧЕНИЯ

Логические значения в Python представлены двумя величинами: логическими константами True (Истина) и False (Ложь).

Логические значения получаются в результате логических операций и вычисления логических выражений.

Операция или выражение	**Описание**
>	Условие "больше" (например, проверяем, что a > b)
<	Условие "меньше" (например, проверяем, что a < b)
==	Условие равенства (проверяем, что a равно b)
!=	Условие неравенства (проверяем, что a не равно b)
not x	Отрицание (условие x не выполняется)
x and y	Логическое "И" (умножение). Чтобы выполнилось условие x and y, необходимо, чтобы одновременно выполнялись условия x и y.
x or y	Логическое "ИЛИ" (сложение). Чтобы выполнилось условие x or y, необходимо, чтобы выполнилось одно из условий.
x in A	Проверка принадлежности элемента x множеству (структуре) A (см. "Структуры данных").
a < x < b	Эквивалентно (x > a) and (x < b)

3. ФУНКЦИИ ПРЕОБРАЗОВАНИЯ ЧИСЕЛ

Ряд встроенных функций в Python позволяют работать с числами. В частности, функции int() и float() позволяют привести значение к типу int и float соответственно.

Например, пусть у нас будет следующий код:

first_number = "2"

second_number = 3

third_number = first_number + second_number

Мы ожидаем, что "" + 3 будет равно 5. Однако этот код сгенерирует исключение, так как первое число на самом деле представляет строку. И чтобы все заработало как надо, необходимо привести строку к числу с помощью функции int():

first_number = "2"

second_number = 3

third_number = int(first_number) + second_number

print(third_number) = 5

Аналогичным образом действует функция float(), которая преобразует в число с плавающей точкой. Но вообще с дробными числами надо учитывать, что результат операций с ними может быть не совсем точным. Например:

first_number = .000second_number = 5

third_number = first_number / second_number

print(third_number) = 0.4000000000000004

В данном случае мы ожидаем получить число 0.4000, однако в конце через ряд нулей появляется еще какая-то четверка. Или еще одно выражение:

print(.0001 + 0.1) = .1001000000000003

В этом случае для округления результата мы можем использовать функцию round():

first_number = .000second_number = 0.third_number = first_number + second_number

print(round(third_number, 4)) = .100

Первый параметр функции - округляемое число, а второй - сколько знаков после запятой должно содержать получаемое число.

3.1. Представление числа

При обычном определении числовой переменной она получает значение в десятичной системе. Но кроме десятичной, в Python мы можем использовать двоичную, восьмеричную и шестнадцатеричную системы.

Для определения числа в двоичной системе перед его значением ставится 0 и префикс b:

x = 0b101 = 101: в десятичной системе равно 5

Для определения числа в восьмеричной системе перед его значением ставится 0 и префикс o:

a = 0o11 = 11: в десятичной системе равно 9

Для определения числа в шестнадцатиричной системе перед его значением ставится 0 и префикс x:

y = 0x0a = a: в десятичной системе равно 1

И с числами в других системах измерения также можно проводить арифметические операции:

```
x = 0b101  = 5
y = 0x0a  = 10
z = x + y  = 15
print("{0} in binary {0:08b}  in hex {0:0x} in octal {0:0o}".format(z))
```

3.2. Функция format

Для вывода числа в различных системах исчисления используются функция format, которая вызывается у строки. В эту строку передаются различные форматы. Для двоичной системы "{0:08b}", где число 8 указывает, сколько знаков должно быть в записи числа. Если знаков указано больше, чем требуется для числа, то ненужные позиции заполняются нулями. Для шестнадцатеричной системы применяется формат "{0:0x}". И здесь все аналогично - запись числа состоит из двух знаков, если один знак не нуден,

то вместо него вставляется ноль. А для записи в восьмеричной системе используется формат "{0:0o}".

Результат работы скрипта:

15 in binary 00001111 in hex 0f in octal 17