Модуль 1. Вводный

Урок 4. Основные операторы

Изучение арифметических и условных операторов. С помощью условного оператора if мы можем выполнять разный код в зависимости от различных типов условий. Развитие логики.

В этом занятии мы вводим понятия логических и арифметических операций и составим программы, которые позволят выполнять тот или иной код в соответствии с заданными условиями. Таким образом, мы сможем решать довольно сложные логические задачи.
Начнем с базовых арифметических операторов.

Арифметические операторы в Python

Оператор	Описание	Примеры
+	Сложение - суммирует значения слева и справа от оператора.	15 + 5 в результате будет 20 20 + -3 в результате будет 17 13.4 + 7 в результате будет 20.4
-	Вычитание - вычитает правый операнд из левого.	15 - 5 в результате будет 10 20 - -3 в результате будет 23 13.4 - 7 в результате будет 6.4
*	Умножение - перемножает операнды.	5 * 5 в результате будет 25 7 * 3.2 в результате будет 22.4 -3 * 12 в результате будет -36
/	Деление - делит левый операнд на правый.	15 / 5 в результате будет 3 5 / 2 в результате будет 2 (в Python 2.x версии при делении двух целых чисел результат будет целое число) 5.0 / 2 в результате будет 2.5 (чтобы получить "правильный" результат хотя бы один операнд должен быть float)
%	Деление по модулю - делит левый операнд на правый и возвращает остаток.	6 % 2 в результате будет 0 7 % 2 в результате будет 1 13.2 % 5 в результате 3.2

**	Возведение в степень - возводит левый операнд в степень правого.	5 ** 2 в результате будет 25 2 ** 3 в результате будет 8 -3 ** 2 в результате будет -9
//	Целочисленное деление - деление, в котором возвращается только целая часть результата. Часть после запятой отбрасывается.	12 // 5 в результате будет 2 4 // 3 в результате будет 1 25 // 6 в результате будет 4

Операторы сравнения в Python

Ряд операций представляют условные выражения. Все эти операции принимают два операнда и возвращают логическое значение, которое в Python представляет тип boolean. Существует только два логических значения – **True** (выражение истинно) и **False** (выражение ложно).

Оператор	Описание	Примеры
==	Проверяет, равны ли оба операнда. Если да, то условие становится истинным.	5 == 5 в результате будет True True == False в результате будет False "hello" == "hello" в результате будет True
!=	Проверяет, равны ли оба операнда. Если нет, то условие становится истинным.	12 != 5 в результате будет True False != False в результате будет False "hi" != "Hi" в результате будет True
<>	Проверяет, равны ли оба операнда. Если нет, то условие становится истинным.	12 <> 5 в результате будет True. Похоже на оператор !=
>	Проверяет больше ли значение левого операнда, чем значение правого.	5 > 2 в результате будет True True > False в результате будет True "A" > "B" в результате будет False

	Если да, то условие становится истинным.	
<	Проверяет, меньше ли значение левого операнда, чем значение правого. Если да, то условие становится истинным.	3 < 5 в результате будет True True < False в результате будет False "A" < "B" в результате будет True
>=	Проверяет, больше или равно значение левого операнда, чем значение правого. Если да, то условие становится истинным.	1 >= 1 в результате будет True 23 >= 3.2 в результате будет True "C" >= "D" в результате будет False
<=	Проверяет меньше или равно значение левого операнда, чем значение правого. Если да, то условие становится истинным.	4 <= 5 в результате будет True 0 <= 0.0 в результате будет True -0.001 <= -36 в результате будет False

Операторы присваивания в Python

Ряд специальных операций позволяют присвоить результат операции первому операнду.

Оператор	Описание	Примеры
=	Присваивает значение правого операнда левому.	c = 23 присвоит переменной c значение 23

+=	Прибавит значение правого операнда к левому и присвоит эту сумму левому операнду.	c = 5 a = 2 c += a равносильно: c = c + a, c будет равно 7
-=	Отнимает значение правого операнда от левого и присваивает результат левому операнду.	c = 5 a = 2 c -= a равносильно: c = c - a, c будет равно 3
*=	Перемножает правый операнд с левым и присваивает результат левому операнду.	c = 5 a = 2 c *= a равносильно: c = c * a, c будет равно 10
/=	Делит левый операнд на правый и присваивает результат левому операнду.	c = 10 a = 2 c /= a равносильно: c = c / a, c будет равно 5
%=	Делит по модулю операнды и присваивает результат левому.	c = 5 a = 2 c %= a равносильно: c = c % a, c будет равно 1
**=	Возводит левый операнд в степень правого и присваивает результат левому операнду.	c = 3 a = 2 c **= a равносильно: c = c ** a, c будет равно 9
//=	Производит целочисленное деление левого операнда на правый и присваивает результат левому операнду.	c = 11 a = 2 c //= a равносильно: c = c // a, c будет равно 5

Побитовые операторы в Python

Побитовые операторы предназначены для работы с данными в битовом (двоичном) формате. Предположим, что у нас есть два числа a = 60; и b = 13. В двоичном формате они будут иметь следующий вид:

a = 0011 1100

b = 0000 1101

Оператор	**Описание**	**Примеры**

&	Бинарный "И" оператор, копирует бит в результат, только если бит присутствует в обоих операндах.	(a & b) даст нам 12, которое в двоичном формате выглядит так 0000 1100
|	Бинарный "ИЛИ" оператор копирует бит, если тот присутствует в хотя бы в одном операнде.	(a | b) даст нам 61, в двоичном формате 0011 1101
^	Бинарный "Исключительное ИЛИ" оператор копирует бит, только если бит присутствует в одном из операндов, но не в обоих сразу.	(a ^ b) даст нам 49, в двоичном формате 0011 0001
~	Бинарный комплиментарный оператор. Является унарным (то есть ему нужен только один операнд), меняет биты на обратные, там где была единица становиться ноль и наоборот.	(~a) даст в результате -61, в двоичном формате выглядит 1100 0011.
<<	Побитовый сдвиг влево. Значение левого операнда "сдвигается" влево на количество бит, указанных в правом операнде.	a << 2 в результате даст 240, в двоичном формате 1111 0000
>>	Побитовый сдвиг вправо. Значение левого операнда "сдвигается" вправо на количество бит, указанных в правом операнде.	a >> 2 даст 15, в двоичном формате 0000 1111

Логические операторы в Python

Оператор	**Описание**	**Примеры**
and	Логический оператор "И". Условие будет истинным, если оба операнда истинны.	True and True равно True True and False равно False False and True равно False False and False равно False

or	Логический оператор "ИЛИ". Если хотя бы один из операндов истинный, то и все выражение будет истинным.	True or True равно True True or False равно True False or True равно True False or False равно False
not	Логический оператор "НЕ". Изменяет логическое значение операнда на противоположное.	not True равно False not False равно True

Операторы тождественности в Python

Операторы тождественности сравнивают размещение двух объектов в памяти компьютера.

Оператор	Описание	Примеры
is	Возвращает истину, если оба операнда указывают на один объект.	x is y вернет истину, если id(x) будет равно id(y)
is not	Возвращает ложь, если оба операнда указывают на один объект.	x is not y, вернет истину, если id(x) не равно id(y)

Приоритет операторов в Python

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.

При последовательном использовании нескольких арифметических операций их выполнение производится в соответствии с их приоритетом. В начале, как и в алгебре, выполняются операции с большим приоритетом.

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 двоичной системе равно 5
```

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

```
a = 0o11    # 11 в восьмеричной системе равно 9
```

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

```
y = 0x0a        # a в шестнадцатеричной системе равно 10
```

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 {0:02x} in octal {0:02o}".format(z))
```

Для вывода числа в различных системах исчисления используется 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 "{0:02x}". И здесь все аналогично: запись числа состоит из двух знаков, если один знак не нужен, то вместо него вставляется ноль. А для записи в восьмеричной системе используется формат "{0:02o}".

Простейшие условные выражения представляют операции сравнения, которые сравнивают два значения.

Операции сравнения могут сравнивать различные объекты – строки, числа, логические значения, однако оба операнда операции должны представлять один и тот же тип.

Если один из операндов оператора and возвращает False, то другой операнд уже не оценивается, так как оператор в любом случае вернет значение False. Подобное поведение позволяет немного увеличить производительность, так как не приходится тратить ресурсы на оценку второго операнда.

Аналогично, если один из операндов оператора or возвращает True, то второй операнд не оценивается, так как оператор в любом случае возвратит True.

Что такое условные операторы?

Они отвечают за изменение поведения программы в зависимости от входных параметров, определённых в проверке. Проще говоря, если будет число 1, то программа запустит скрипт one, а если число 2 – скрипт two. Внутри условных операторов могут быть другие такие же условия для уточнения полученных данных. В рамках одного оператора можно сразу проверить пару условий. Для того, чтобы проверить несколько условий, нужно их разделить элементом and (логическое “и”).

Условные конструкции используют условные выражения, и, в зависимости от их значения, направляют выполнение программы по одному из путей. Одна из таких конструкций – это конструкция **if**. Она имеет следующее формальное определение:

```
if логическое_выражение:
    инструкции
[elif логическое выражение:
    инструкции]
[else:
    инструкции]
```

Пример создания условия:

```
a = 2
if a != 0 and a != 1:
        print ("Проверка сработала")
```

На экране будет показана запись лишь в том случае, когда переменная a не будет равна значению 0 и значению 1. То есть, обе проверки в операторе должны выдать результат – **true**.

Есть возможность произвести проверку с помощью or – логическое “или”. При использовании данного оператора достаточным поводом для

запуска сообщения «Заработало» станет соответствие хотя бы одного из условий.

Пример:

```
a = 1.1
if a != 1.1 or a > 0:
        print ("Проверка сработала")
```

В процессе разработки может возникнуть ситуация, в которой после одной истинной проверки следует сделать еще несколько. В таком случае необходимо использовать вложенные условные конструкции, то есть одну if...elif...else конструкцию внутри другой.

Логика выполнения вложенных условных конструкций та же, что и у обычных. Главное – не запутаться с отступами и порядком выполнения сравнений.

Исходный код

Операции на условие

```
num = input ("Введите число: ")


if int (num) > 0:
if int (num) > 10:
        print ("Вы ввели число больше 10")
        if int (num) >= 50:
                print ("Вы ввели число больше 50")
else:
        print ("Вы ввели число меньше 10 и больше 0")
elif int (num) < -10:
        print ("Вы ввели число меньше -10")
else:
```

```
            print ("Вы ввели число меньше 0 и больше -10")
        print ("Все отлично!")
        name = input ()
        A = 'Yes' if name != "Test" else 'No'
print (A)
```

В этом занятии мы разобрали базовые операции и условные операторы, для закрепления материала выполните задания.

Найти максимальное число из трех

Вводятся три целых числа. Определить какое из них наибольшее.

Определить существование треугольника и его тип

По длинам трех отрезков, введенных пользователем, определить возможность существования треугольника, составленного из этих отрезков. Если такой треугольник существует, то определить, является ли он прямоугольным, равнобедренным или равносторонним.