Лекция № 4

Основы языка Python

Цель работы

Целью лекции является знакомство с алгоритмическим языком Python

План

- **Арифметические действия**
- **Присваивание значений**
- **Сравнение величин**
- **Проверка условий**
- **Определение приоритетов**
- **Преобразование типов данных**
- **Манипуляции с битами**

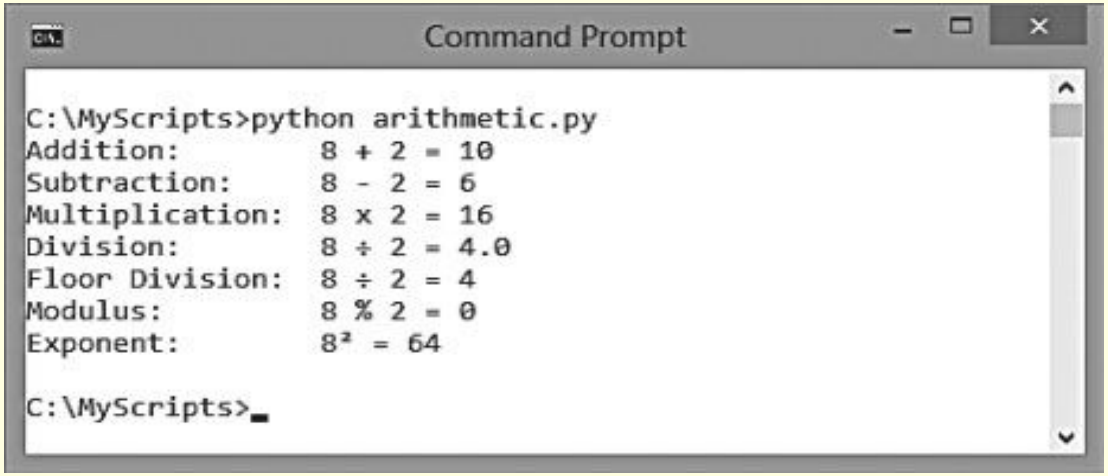
Арифметические действия

Оператор	Операция
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Деление по модулю
//	Целочисленное деление
**	Возведение в степень

```
a = ( b * c ) - ( ( d % e ) / f )
```

Арифметические действия

```
a = 8
b = 2
print( 'Addition:\t' , a , '+' , b , '=' , a + b )
print( 'Subtraction:\t' , a , '-' , b , '=' , a - b )
print( 'Multiplication:\t' , a , 'x' , b , '=' , a * b )
print( 'Division:\t' , a , '÷' , b , '=' , a / b )
print( 'Floor Division:\t' , a , '÷' , b , '=' , a // b )
print( 'Modulus:\t' , a , '%' , b , '=' , a % b )
print( 'Exponent:\t ' , a , '² = ' , a ** b , sep = '' )
```

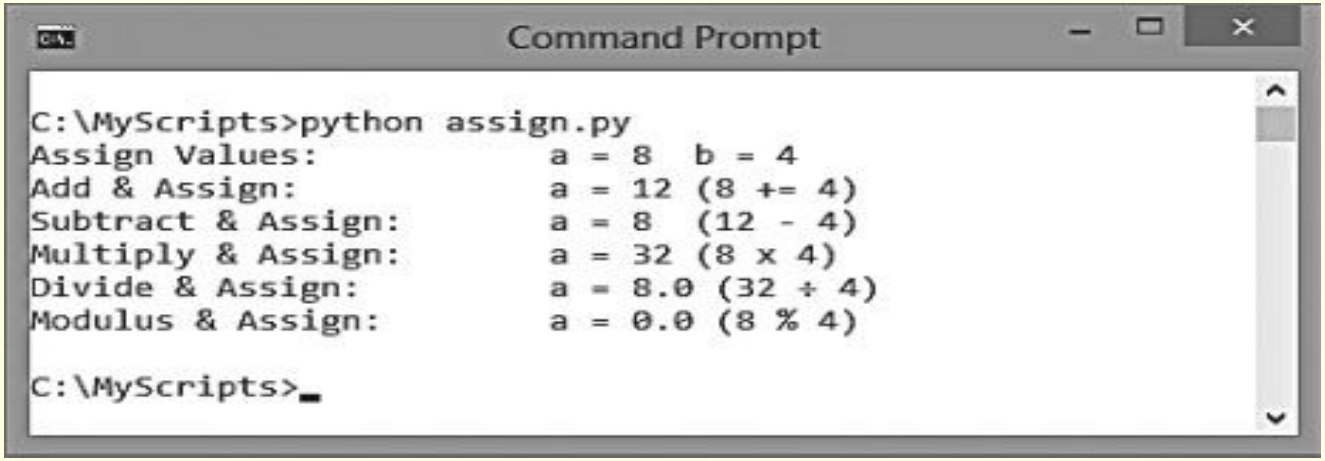


Присваивание значений

Оператор	Пример	Эквивалентная операция
=	a = b	a = b
+=	a += b	a = (a + b)
-=	a -= b	a = (a - b)
*=	a *= b	a = (a * b)
/=	a /= b	a = (a / b)
%=	a %= b	a = (a % b)
//=	a //= b	a = (a // b)
**=	a **= b	a = (a ** b)

Присваивание значений

```
a = 8
b = 4
print( 'Assign Values:\t\t' , 'a =' , a , '\tb =' , b ).
a += b
print( 'Add & Assign:\t\t' ,'a =' , a , '(8 += 4)' )
a -= b
print( 'Subtract & Assign:\t' , 'a =' , a , ' (12 - 4)' ) a *= b
print( 'Multiply & Assign:\t' , 'a =' , a , '(8 x 4)' )
a /= b
print( 'Divide & Assign:\t' , 'a =' , a , '(32 ÷ 4)' ) a %= b
print( 'Modulus & Assign:\t' , 'a =' , a , '(8 % 4)' )
```

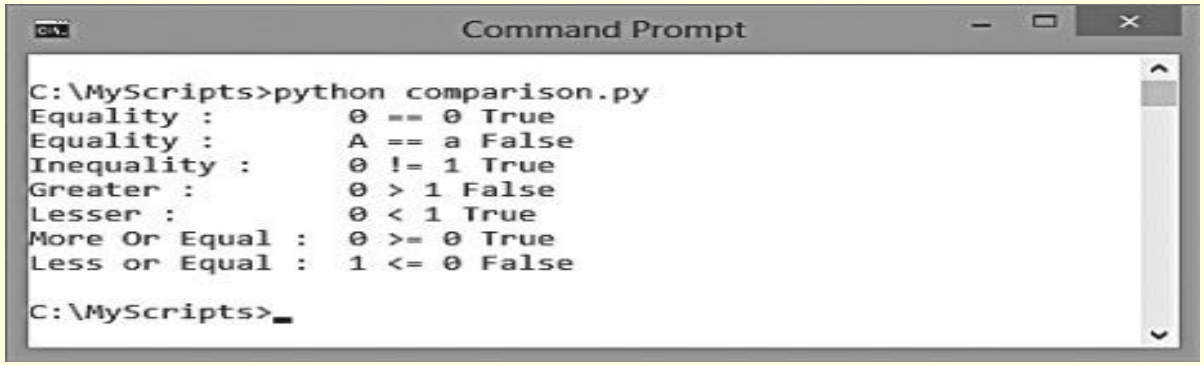


Сравнение величин

Оператор	Проверяемое условие
= =	Равенство
! =	Неравенство
>	Больше
<	Меньше
> =	Больше либо равно
< =	Меньше либо равно

Сравнение величин значений

```
nil = 0
num = 0
max = 1
cap = 'A'
low = 'a'
print( 'Equality :\t' , nil , '==' , num , nil == num )
print( 'Equality :\t' , cap , '==' , low , cap == low )
print( 'Inequality :\t' , nil , '!=' , max , nil != max )
print( 'Greater :\t' , nil , '>' , max , nil > max )
print( 'Lesser :\t' , nil , '<' , max , nil < max )
print( 'More Or Equal :\t' , nil , '>=' , num , nil >= num )
print( 'Less or Equal :\t' , max , '<=' , num , max <= num )
```

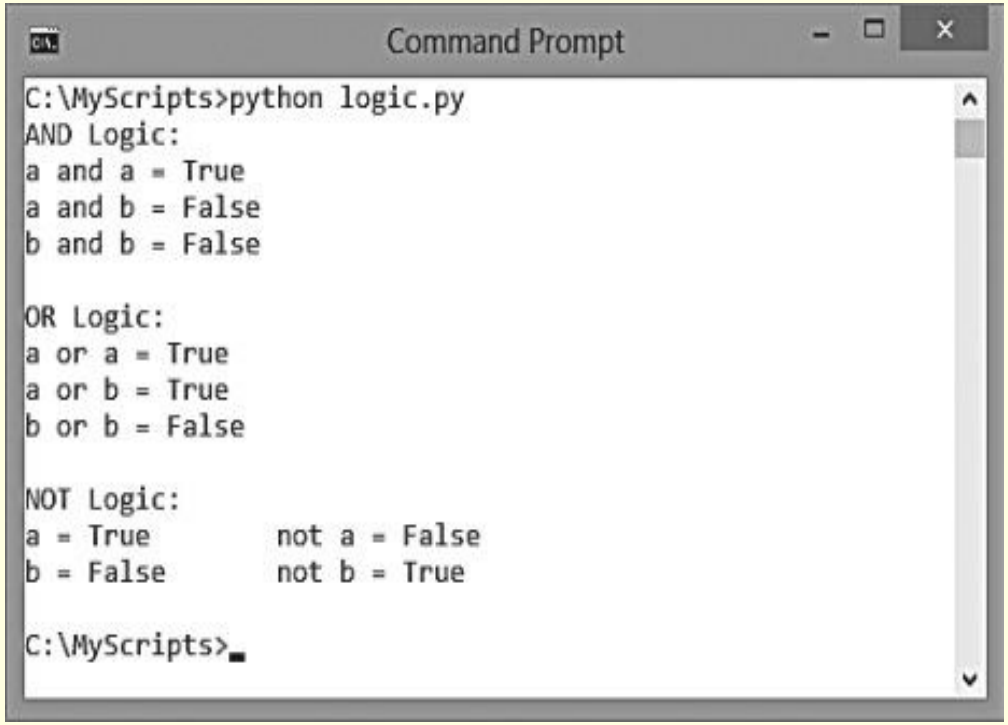


Оценочная логика

Оператор	Операция
and	Логическое «И»
or	Логическое «ИЛИ»
not	Логическое «НЕ»

Оценочная логика

```
a = True
b = False
print( 'AND Logic:' )
print( 'a and a =' , a and a )
print( 'a and b =' , a and b )
print( 'b and b =' , b and b )
print( '\nOR Logic:' )
print( 'a or a =' , a or a )
print( 'a or b =' , a or b )
print( 'b or b =' , b or b )
print( '\nNOT Logic:' )
print( 'a =' , a , '\tnot a =' , not a )
print( 'b =' , b , '\tnot b =' , not b )
```

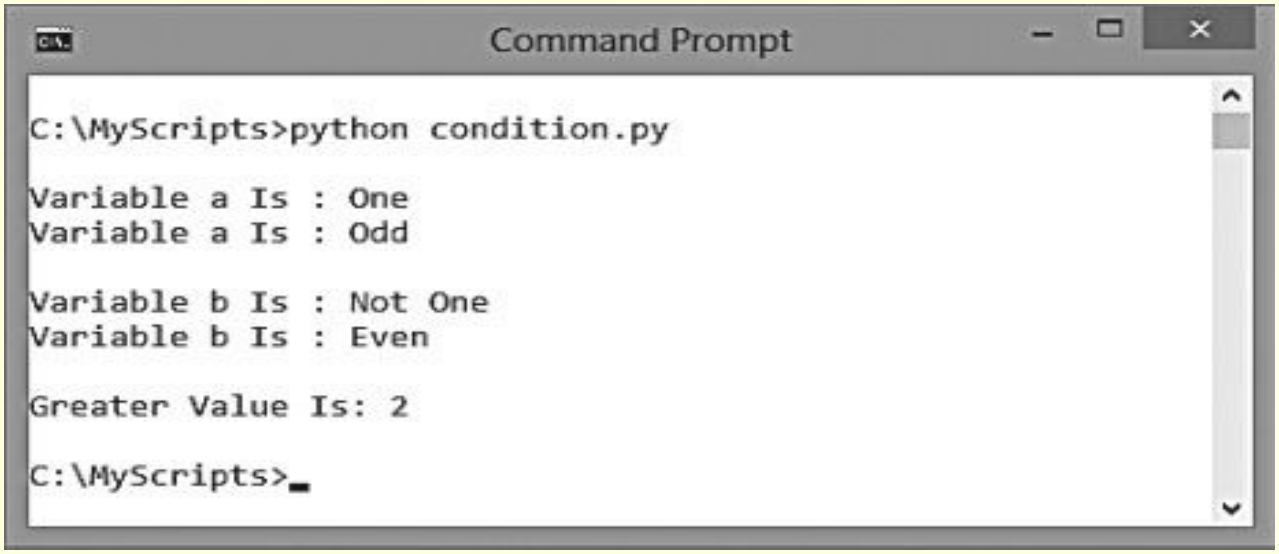


Проверка условий

- **если-истина-возвращаем-это if (проверочное-выражение) else если-ложь-возвращаем-это**
- ***c = a if (a < b) else b***
- ***если-истина(нечетное)-выполняем-это if (var % 2 != 0) else если-ложь(четное)-выполняем-это***

Проверка условий

```
a = 1
b = 2
print( '\nVariable a Is :' , 'One' if ( a == 1 ) else 'Not One' )
print( 'Variable a Is :' , 'Even' if ( a % 2 == 0 ) else 'Odd' )
print( '\nVariable b Is :' , 'One' if ( b == 1 ) else 'Not One' )
print( 'Variable b Is :' , 'Even' if ( b % 2 == 0 ) else 'Odd' )
max = a if ( a > b ) else b
print( '\nGreater Value Is:' , max )
```



Определение приоритетов

Приоритет операторов определяет порядок, которому интерпретатор Python следует при оценке выражений. Например, в выражении

3 * 8 + 4 **порядок действий по умолчанию определяет, что умножение выполняется первым, так что результат**

равен 28 (24 + 4).

В таблице ниже перечисляются операторы в порядке убывания приоритета.

Те из них, что находятся в строках выше, имеют более высокий приоритет.

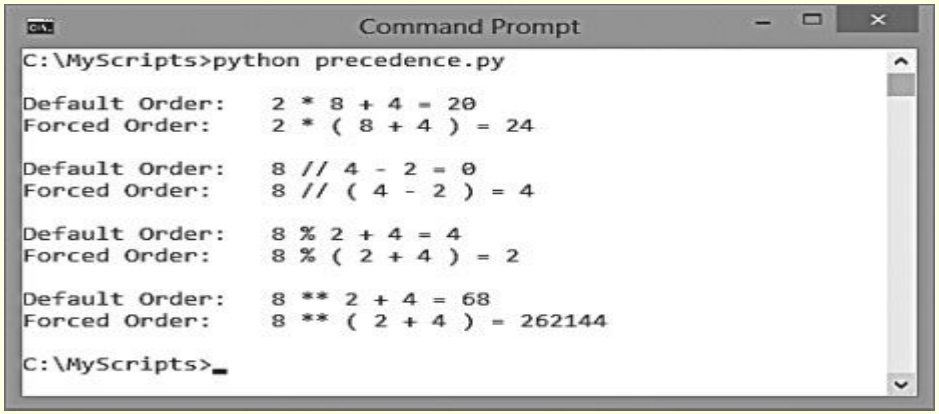
Приоритет операторов, находящихся на одной строке в таблице, определяется правилом «слева направо».

Определение приоритетов

Оператор	Описание
**	Возведение в степень
+ - ~	Положительное значение Отрицательное значение Побитовое отрицание
* / // %	Умножение Деление Целочисленное деление Деление по модулю
+ -	Сложение Вычитание
|	Побитовое ИЛИ
^	Побитовое исключающее ИЛИ
&	Побитовое И
>> <<	Побитовый сдвиг вправо Побитовый сдвиг влево
>, < =, <, < =, = =, !=	Сравнение
=, %=, /=, //=, -=, +=, *=, **=	Присваивание
is, is not	Идентичность
in, not in	Вхождение
not	Логическое отрицание
and	Логическое И
or	Логическое ИЛИ

Определение приоритетов

```
a = 2
b = 4
c = 8
print( '\nDefault Order:\t', a, '*', c,'+', b, '=', a * c + b )
print( 'Forced Order:\t', a, '* (', c,'+', b, ') =', a * ( c + b ) )
print( '\nDefault Order:\t', c, '//', b, '-', a, '=', c // b - a )
print( 'Forced Order:\t', c, '// (', b,'-', a, ') =', c // ( b - a ) )
print( '\nDefault Order:\t', c, '%', a, '+', b, '=', c % a + b )
print( 'Forced Order:\t', c, '% (', a, '+', b, ') =', c % ( a + b ) )
print( '\nDefault Order:\t', c, '**', a, '+', b, '=', c ** a + b )
print( 'Forced Order:\t', c, '** (', a, '+', b, ') =', c ** ( a + b ) )
```



Преобразование типов данных

- Переменные в языке Python могут хранить данные любого типа, и очень важно различать эти типы для того, чтобы избежать ошибок при обработке данных в программе.
- Типов данных несколько, рассмотрим основные из них:
- строковые (str (string)),
- целочисленные (int (integer)),
- с плавающей точкой (float (floating-point)).

Преобразование типов данных

- **Очень важно различать типы данных, особенно при присваивании переменным значений, используя пользовательский ввод:**
- **по умолчанию в нем хранится строковый тип данных.**
- **Строковые величины не могут быть использованы для арифметических выражений, и попытка сложить два строковых значения просто объединяет эти строки, а не использует операции над числами.**
- **Например, *'8' + '4' = '84'.***

Преобразование типов данных

- **Важно:** любые типы данных, хранящиеся в переменных в Python, могут быть легко преобразованы (приведены) к другим типам с помощью функций преобразования.
- Встроенные функции преобразования типов данных в Python возвращают новый объект, представляющий преобразованную величину.

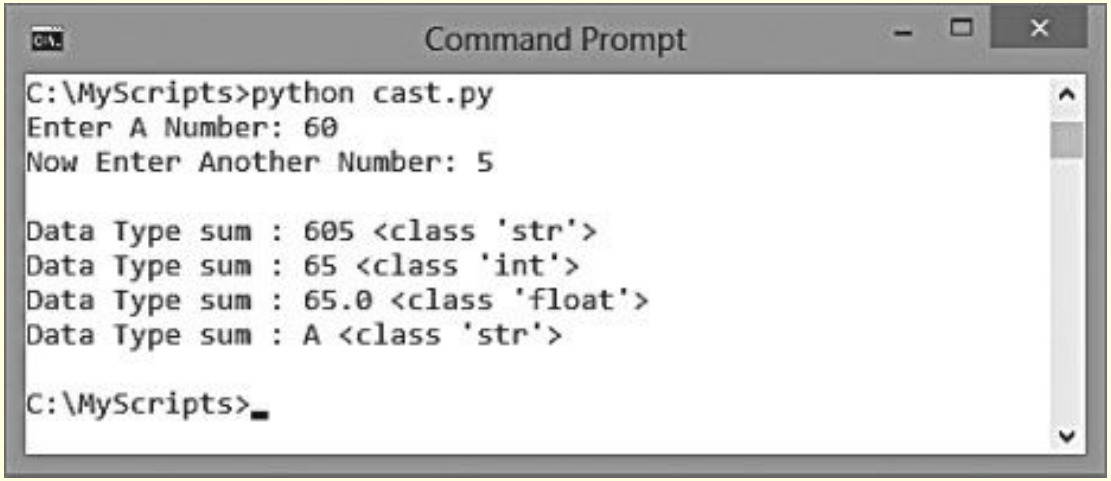
Преобразование типов данных

Функция	Описание
int(x)	Преобразует x в целое число
float(x)	Преобразует x в число с плавающей точкой
str(x)	Преобразует x в строковое представление
chr(x)	Преобразует целое x в символ
unichr(x)	Преобразует целое x в символ Юникода (Unicode)
ord(x)	Преобразует символ x в соответствующее ему целое число
hex(x)	Преобразует целое x в шестнадцатеричную строку
oct(x)	Преобразует целое x в восьмеричную строку

Преобразование типа данных с плавающей точкой (float) в целочисленный тип данных (int) отбрасывает десятичную часть числа.

Преобразование типов данных

```
a = input( 'Enter A Number: ' )
b = input( 'Now Enter Another Number: ' )
sum = a + b
print( '\nData Type sum :' , sum , type( sum ) )
sum = int( a ) + int( b )
print( 'Data Type sum :' , sum , type( sum ) )
sum = float( sum )
print( 'Data Type sum :' , sum , type( sum ) )
sum = chr( int( sum ) )
print( 'Data Type sum :' , sum , type( sum ) )
```



Манипуляции с битами

- В компьютерной терминологии каждый байт состоит из восьми битов, каждый бит может содержать либо 1, либо 0 для хранения двоичного представления десятичных чисел от 0 до 255 (всего 256 комбинаций 0 и 1).
- Каждый бит является компонентом десятичного числа, если он содержит единицу.
- Компоненты распределены справа налево от наименее (LeastSignificant Bit, LSB) до наиболее значащих битов (Most Significant Bit, MSB).

Манипуляции с битами

Двоичное число 00110010 представляет десятичное число 50 (2+16+32):

№ бита	8 MSB	7	6	5	4	3	2	1LSB
Десятичное	128	64	32	16	8	4	2	1
Двоичное	0	0	1	1	0	0	1	0

Манипуляции с битами

В языке Python можно работать с отдельными частями байта, используя побитовые операторы:

Оператор	Название	Операция с двоичными числами
|	ИЛИ	Возвращает 1 в каждый бит, где один из сравниваемых битов имел значение 1 Пример: 1010 | 0101 = 1111
&	И	Возвращает 1 в каждый бит, где оба сравниваемых бита имели значение 1 Пример: 1010 & 1100 = 1000
~	НЕ	Возвращает 1 в каждый бит, где ни один из двух сравниваемых битов не имел значение 1 Пример: 1010 ~ 0011 = 0100
^	Исключающее ИЛИ	Возвращает 1 в каждый бит, где только один из двух сравниваемых битов имел значение 1 Пример: 1010 ^ 0100 = 1110
<<	Сдвиг влево	Сдвигает влево биты на указанное количество позиций Пример: 0010 << 2 = 1000
>>	Сдвиг вправо	Сдвигает вправо биты на указанное количество позиций Пример: 1000 >> 2 = 0010

Манипуляции с битами

Побитовые операторы редко используются программистами, но они могут быть полезны.
Например,
оператор «исключающее ИЛИ» позволяет вам поменять значение двух переменных между собой без использования третьей.

Манипуляции с битами

```
a = 10
b = 5
print( 'a =' , a , '\tb = ' , b )
# 1010 ^ 0101 = 1111 (десятичное 15)
a = a ^ b
# 1111 ^ 0101 = 1010 (десятичное 10)
b = a ^ b
# 1111 ^ 1010 = 0101 (десятичное 5)
a = a ^ b
print( 'a =' , a , '\tb = ' , b )
```

Заключение

- Арифметические операторы могут формировать выражения с двумя операндами, организуя операции сложения +, вычитания -, умножения *, деления /, целочисленного деления //, деления по модулю % и возведения в степень **.
- Оператор присваивания = можно комбинировать с арифметическими операторами, чтобы производить арифметические вычисления и сразу же присваивать их результаты.
- • Операторы сравнения формируют выражения, сравнивающие два операнда.
- Например, равенство ==, неравенство !=, а также больше >, меньше <, больше либо равно >=, меньше либо равно <=.

Заключение

- «Логическое И», and, и «логическое ИЛИ», or, являются операторами, формирующими выражение для оценки двух операндов и возвращающими булево значение True (Истина) или False (Ложь).
- • Оператор «логическое НЕ», not, возвращает обратное булево значение единственного операнда.
- • Условное выражение if-else оценивает заданное условие и возвращает один из двух операндов в зависимости от результата оценки.

Заключение

- Операции в выражениях выполняются в соответствии с правилами приоритета, если явно не используются дополнительные скобки.
- • Тип данных переменной может быть преобразован к другому типу с помощью встроенных в Python функций int(), float() и str().
- • Встроенная функция type() определяет, к какому типу данных принадлежит указанная переменная.

Заключение

- Побитовые операторы «ИЛИ», |, «И», &, «НЕ», ~, и «исключающее ИЛИ», ^, сравнивают два бита и возвращают соответствующее значение,
- Операторы «сдвиг влево», <<, и «сдвиг вправо»,
- >>, производят сдвиг указанного количества значащих битов в соответствующем направлении.

Благодарю за внимание!