**Операторы**

Операторы позволяют произвести определенные действия с данными. Например, операторы присваивания служат для сохранения данных в переменной, математические операторы позволяют выполнить арифметические вычисления, а оператор конкатенации строк используется для соединения двух строк в одну. Рассмотрим операторы, доступные в **Python 3**.

**3.1 Математические операторы**

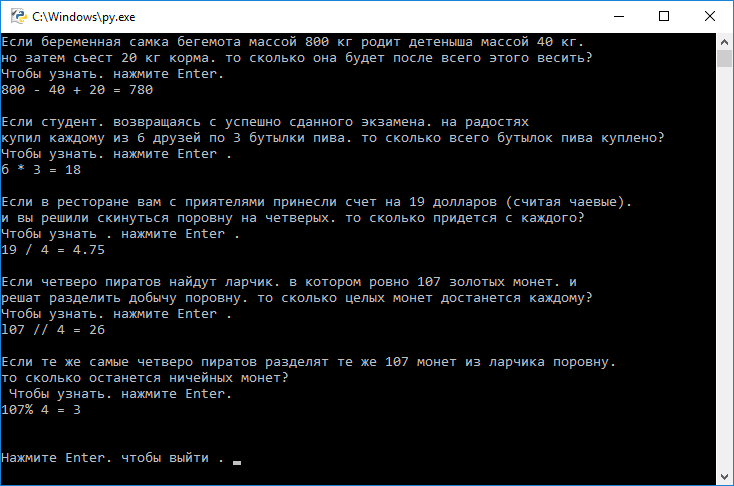
Производить операции над числами позволяют следующие операторы:

* + сложение;
* - вычитание;
* \* умножение;
* */* деление. Результатом деления всегда является вещественное число, даже если производится деление целых чисел.
* //деление с округлением вниз. Вне зависимости от типа чисел остаток отбрасывается.
* % остаток от деления;
* \*\* возведение в степень;
* унарный - (минус) и унарный + (плюс).

Если в операции участвуют целое и вещественное числа, то целое число будет автоматически преобразовано в вещественное число, а затем произведена операция над вещественными числами. Результатом этой операции будет вещественное число.

При выполнении операций над вещественными числами следует учитывать ограничения точности вычислений.

Примеры использования математических операторов:



**3.2 Двоичные операторы**

Побитовые операторы предназначены для манипуляции отдельными битами. Язык **Python** поддерживает следующие побитовые операторы: ·

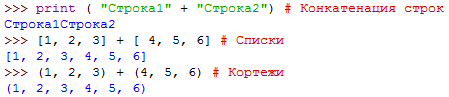
* ~ двоичная инверсия. Значение каждого бита заменяется на противоположное:  
  ~01100100 = 10011011
* & двоичное И: 01100100 & 01001011 = 01000000
* | двоичное ИЛИ: 01100100 | 01001011 = 01101111
* ^ двоичное исключающее ИЛИ: 01100100 ^ 11111010 = 10011110
* << сдвиг влево - сдвигает двоичное представление числа влево на один или более разрядов и заполняет разряды справа нулями: 01100100 <<2 = 10010000
* >> сдвиг вправо - сдвигает двоичное представление числа вправо на один или более разрядов и заполняет разряды слева нулями, если число положительное:  
  01100100 >> 2 = 00011001

Если число отрицательное, то разряды слева заполняются единицами:  
10000001 >> 2 = 11100000

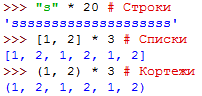
**3.3 Операторы для работы с последовательностями**

Для работы с последовательностями предназначены следующие операторы:

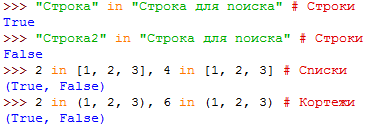
* **+** конкатенация:



* \* повторение:



* **in** проверка на вхождение. Если элемент входит в последовательность, то возвращается логическое значение **True**:



**3.4 Операторы присваивания**

Операторы присваивания предназначены для сохранения значения в переменной. Перечислим операторы присваивания, доступные в языке **Python**:

* **=** присваивает переменной значение:



* **+=** увеличивает значение переменной на указанную величину:



Для последовательностей оператор **+=** производит конкатенацию:



* **-=** уменьшает значение переменной на указанную величину:



* **\*=** умножает значение переменной на указанную величину:



Для последовательностей оператор **\*=** производит повторение:



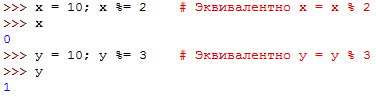
* /= делит значение переменной на указанную величину:



* ***//=***деление с округлением вниз и присваиванием:



* **%=** деление по модулю и присваивание:



* \*\*= возведение в степень и присваивание:



**3.5 Приоритет выполнения операторов**

В какой последовательности будет вычисляться приведённое ниже выражение?

**х = 5 + 10 \* 3 /2**

Это зависит от приоритета выполнения операторов. В данном случае последовательность вычисления выражения будет такой:

1. Число 10 будет умножено на 3, т. к. приоритет оператора умножения выше приоритета оператора сложения.
2. Полученное значение будет поделено на 2, т. к. приоритет оператора деления равен приоритету оператора умножения (а операторы с равными приоритетами выполняются слева направо), но выше чем у оператора сложения.
3. К полученному значению будет прибавлено число *5,* т. к. оператор присваивания **=** имеет наименьший приоритет.
4. Значение будет присвоено переменной х.



С помощью скобок можно изменить последовательность вычисления выражения:

**х = (5 + 10) \* 3 /2**

Теперь порядок вычислений будет другим:

1. К числу *5* будет прибавлено 10.
2. Полученное значение будет умножено на 3.
3. Полученное значение будет поделено на 2.
4. Значение будет присвоено переменной х.



Перечислим операторы в порядке убывания приоритета:

1. **-х, +х, ~х, \*\*** : унарный минус, унарный плюс, двоичная инверсия, возведение в степень. Если унарные операторы расположены слева от оператора \*\*, то возведение в степень имеет больший приоритет, а если справа- то меньший. Например, выражение **-10 \*\* -2** эквивалентно следующей расстановке скобок:   
   -(10 \*\* (-2))
2. **\*, %, /, //** : умножение (повторение), остаток от деления, деление, деление с округлением вниз.
3. **+,-** : сложение (конкатенация), вычитание.
4. **<<, >>** : двоичные сдвиги.
5. **&** : двоичное И.
6. **^** : двоичное исключающее ИЛИ.
7. | : двоичное ИЛИ.
8. **=, +=, -=, \*=, /=, //=, %=, \*\*=** : присваивание.