Самое важное

Простые арифметические операции

- Сложение — (a+b)
- Вычитание — (a-b)
- Умножение — (a*b)
- Деление — (a/b)
- Возведение в степень — (a**b)

В большинстве случаев арифметические операции между разными типами данных завершатся ошибкой. Например: *‘Привет’ + 2*. В этом случае Python просто не поймёт, что именно вы хотите сделать.

Приоритет операций

Приоритет операций		
Сначала	Возведение в степень	5 ** 6
Потом	Деление, умножение	3 / 2, 6 * 7
И в самом конце	Сложение, вычитание	3 - 2, 5 + 6

Изменить приоритет так, как нужно, нам помогут скобки:

*2 + 3 * 4 = 2 + 12 = 14 —> (2 + 3) * 4 = 5 * 4 = 20.*

Функция int() поможет привести объект к числу типа int. Самое частое применение — приведение числа-str к числу-int:

- пример числа-str — ‘2’;
- пример числа int — 2.

Внимание: input() хоть и позволяет вводить числа, но любой ввод пользователя эта функция преобразует в строку! То есть если через input будет введено число, то к нему нужно будет сперва применить функцию int(), а уже потом выполнять с ним арифметические операции.

- x_string = input(‘введите число’) — так мы получим ввод пользователя;
- x_integer = int(x_string) — так превратим число в нужный нам тип;
- result = x_integer - 2 — и теперь мы можем использовать это число в расчётах.

Важно: через input() можно ввести что угодно, и всё это превратится в строку. Если мы попробуем привести строку, состоящую из цифр и букв, к int-типу, то получим ошибку. Например: *int('123аб')* — Python не поймёт, что вам нужно только число из строки, он будет до конца пытаться привести 'а' и 'б' к числам.

Деление нацело и остаток от деления: деление нацело позволяет получить целую часть от деления одного объекта на другой:

- *9/2 = 4.5* — обычное деление;
- *9//2 = 4* — целочисленное деление.

Также важной операцией является получение остатка от деления на число: *9%2 = 1.* Например: получение остатка от деления на 2. Такой остаток может быть равен либо 0, либо 1. Мы делим чётное число — остаток равен 0, мы делим нечётное число — остаток равен 1.

Например:

- *9/2 = 4.5;*
- *9%2 = 1.*

Получение остатка от деления не вернёт нам 0.5, так как эта операция возвращает ту часть числа, которую не вышло поделить, а не дробную часть результата деления. Так, при делении 9 на 2 мы можем поделить 8 на 2, а 1 останется в остатке. *14%3*: 12 мы сможем поделить, а 2 останется в остатке.
Отсюда выводится правило: остаток не может быть больше или равен числу, на которое происходит деление. Например: мы записываем *15%3* (увеличим прошлое число 14 на +1) и получаем остаток равный 0, а не 3.

Сокращённые операторы
Когда речь идёт об увеличении существующего числа, мы можем писать так:

- *a = 1,*
- *a = a + 2,*

а можем сократить вторую запись до:

- *a += 2.*

Это работает и с другими операторами!

Сокращённые операторы нельзя использовать с числами, которые ещё не были созданы. Нельзя написать *a += 2,* если переменной 'а' ещё не было создано в текущем коде. Это приведёт к заслуженной ошибке, ведь Python просто не будет знать, к чему ему добавлять +2.