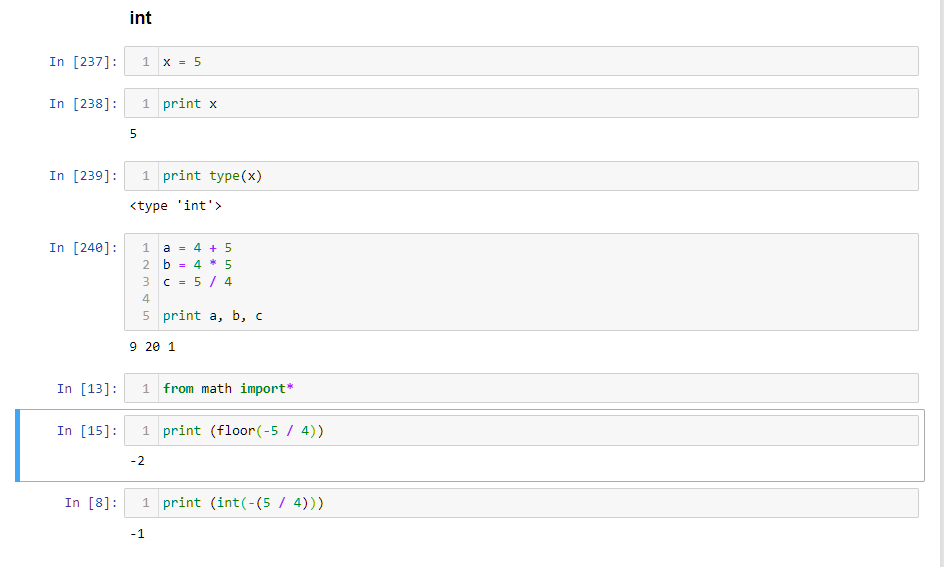
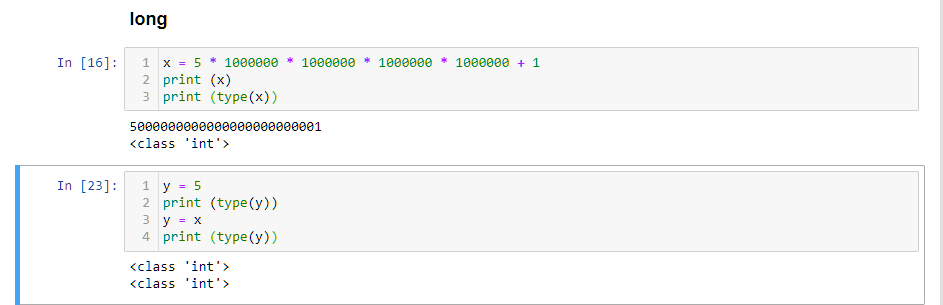
**1. Числовой тип**

**Целый(int)**

Допустим, нам нужно завести переменную, в которой будет некоторое целое число, ну, например, количество чего-нибудь. Ну это очень просто.  С int-овыми переменными можно выполнять некоторые простые операции: складывать,  
умножать, делить.

  
Пришлось немного исправить, так как в курсе используется python 2.7, а у меня 3.8. Поэтому некоторые момент 6нужно было править.  
  
**Длинный(long)**

Когда-то нам придется работать с большими числами и в этом нам поможет тип **long.**(В Python 3 этот тип переименован в [int](http://pythonz.net/references/named/int/).)

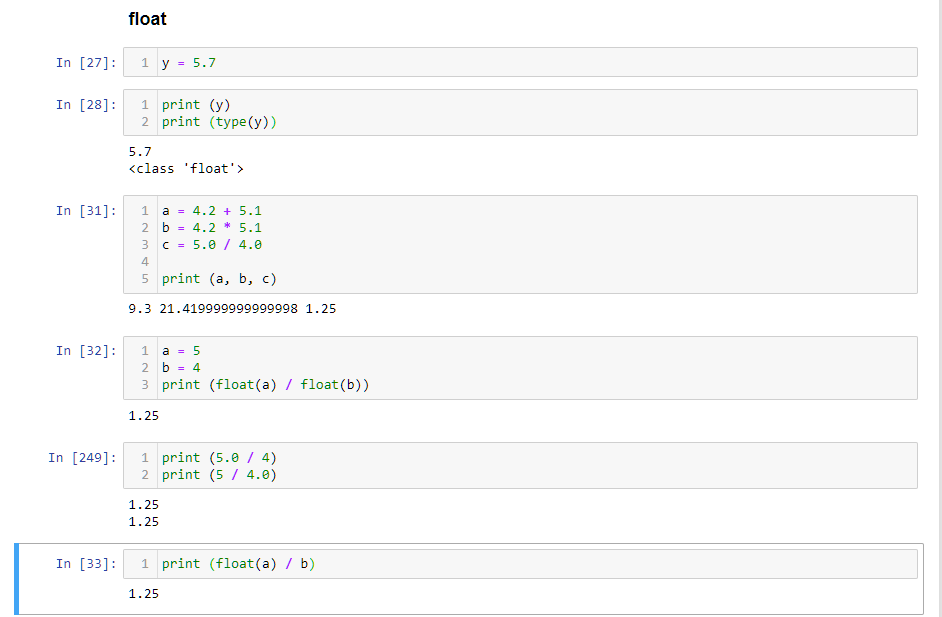


**Дробный(float)**

Ну и, конечно, в «Питоне» есть дробные числа.

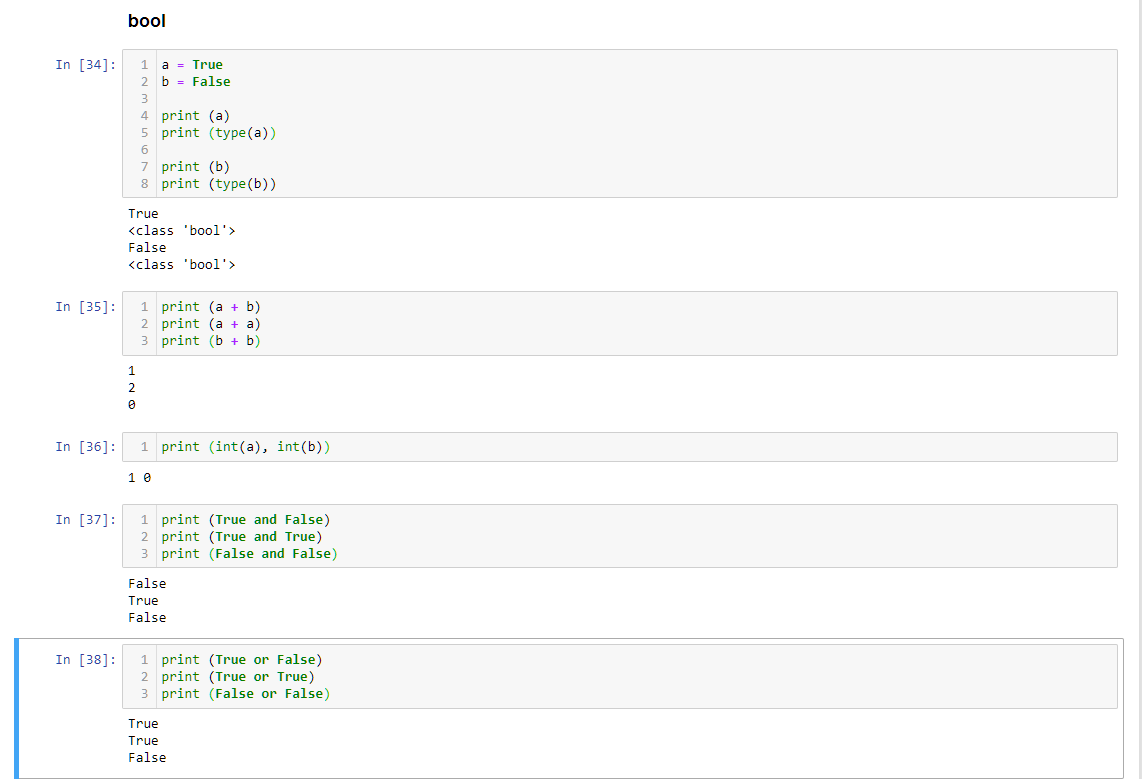
Они представлены типом float. С ними тоже можно делать простые операции: сложение, умножение, деление.

Ну и, конечно, здесь уже деление не целочисленное.



**Логический(bool)**

«Питоне» есть логический тип данных, bool, почему он находится рядом с числовым типом, ну хотя бы потому что его легко можно привести к типу int, ну и потому, что если мы попробуем поскладывать разные значения, то мы получим вот такой вот интересный результат. То есть, оказывается, true + true будет равно двум. Ну, в самом деле, если мы приведем к int, то мы убедимся, что true это у нас единичка, а false это у нас 0. true и false в «Питоне» пишутся с большой буквы. Ну и, конечно, есть обычные привычные операции с логическим типом. Это операция «и», «или».



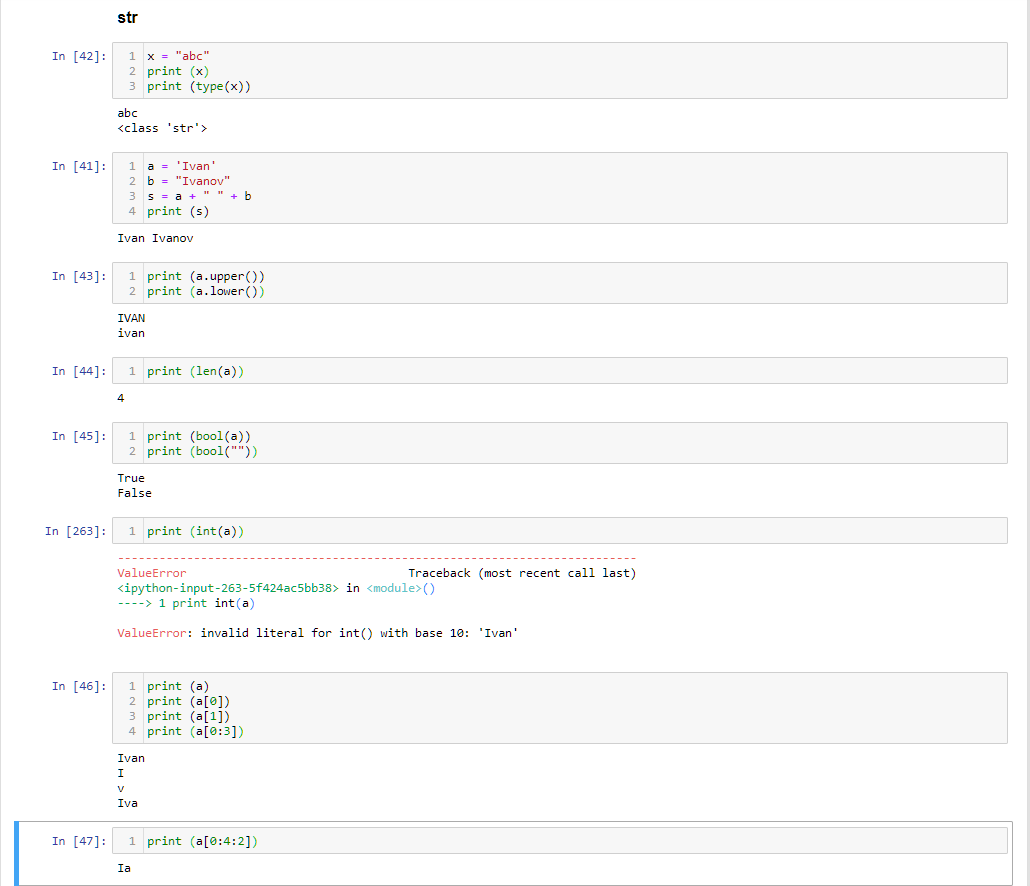
**2. None**

В «Питоне» есть тип none. Это не 0 и не false. Это специальный тип, который означает отсутствие значения, у него даже свое название типа: NoneType. И его, конечно же, не получится привести к какому-то другому типу. Но как один из примеров того, откуда может взяться none, можно присвоить какой-нибудь переменный результат вычисления функций, которая ничего не возвращает.



**3. Строковые**

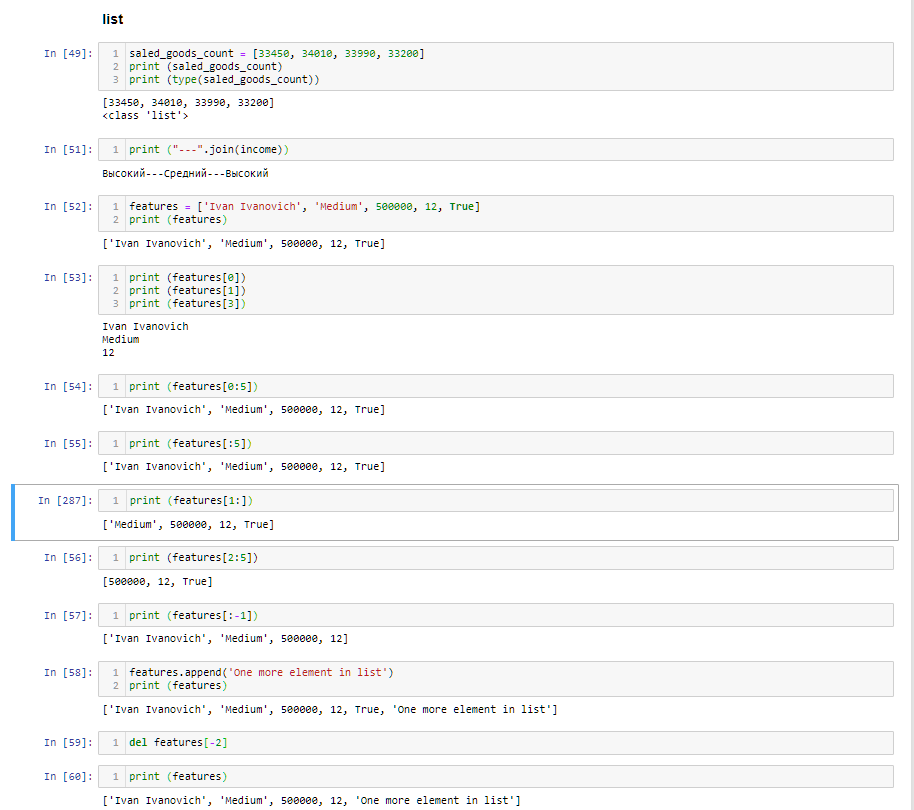
Ну и, конечно, вы можете записывать переменные строки. Для этого есть тип str. С ним тоже есть какие-то простые операции, тоже можно складывать, но для строк сложение означает уже дописывание к одной строке другой строки. Можно приводить к нижнему регистру или к верхнему регистру. Можно получить длину строки. Можно привести к типу bool, если строка имеет не нулевую длину, то есть не пустая строка, то результат будет true. В противном случае, то есть для пустой строки, результат будет false. Ну и, конечно, приводить к числовому типу уже не получается. Ну и можно делать срезы строк.



**4. Массивы**

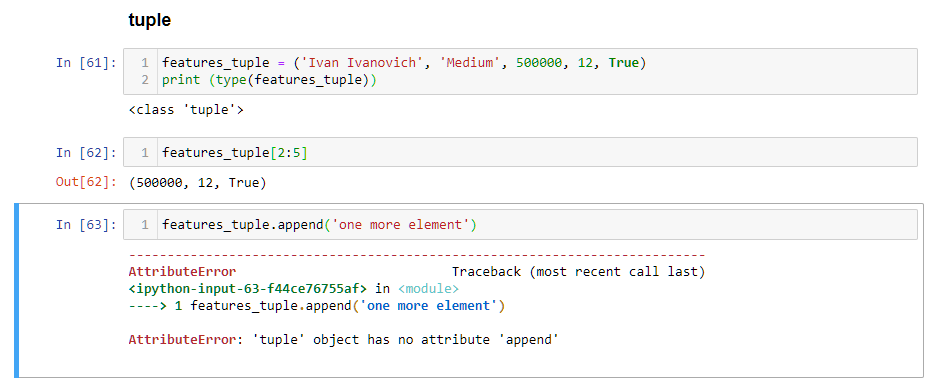
**Изменяемое множество(list)**

Одним из типов для массивов — это list. Ну что мы можем хранить в массивах? Мы можем хранить какие-то числа, ну, например, количество проданных товаров в разные дни, можем хранить какие-то строчки, например, уровень дохода разных людей, которые хотят взять кредит или брали когда-то кредит, или их имена. Индексация работает также как в строчках, мы действительно получаем элемент какой нужно по индексу, действительно можем делать срез также, как в строчках, можем делать срез до какого-то элемента или от какого то элемента, и можем использовать индексацию с конца массива. Также можем добавлять элемент массива с помощью метода append и удалять элементы.



**Неизменяемое множество(tuple)**

Кроме того, есть еще один тип данных для массивов — это тип данных tuple.(мы не можем пользоваться append, так как это неизменяемое множество)

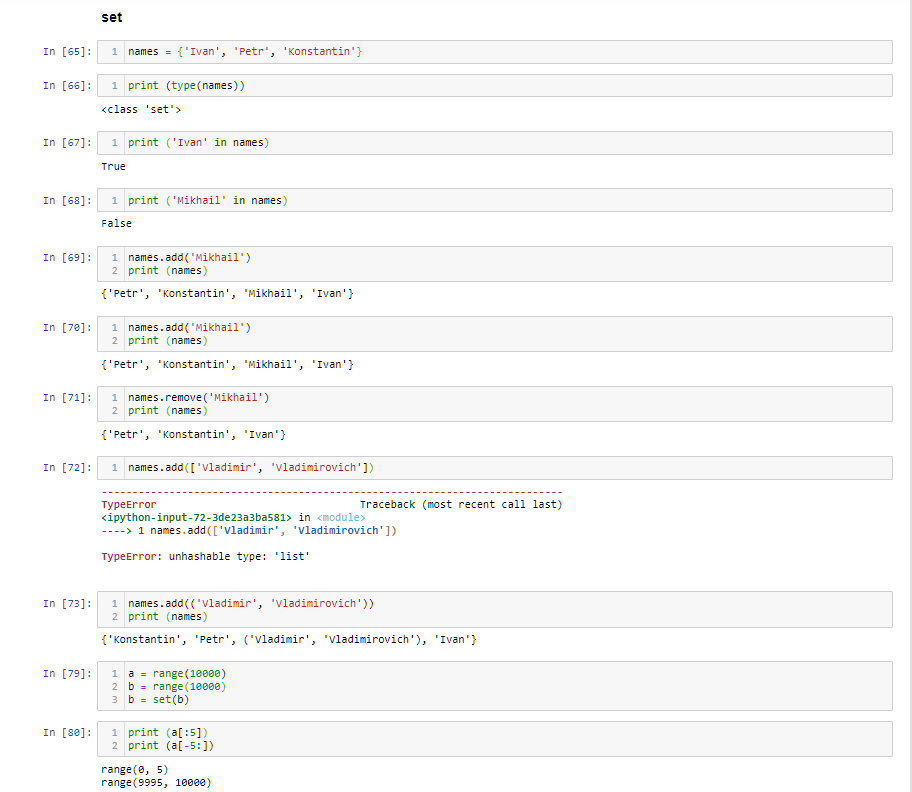


**5. Множества и словари**

**Множество(set)**

Допустим, вы хотите создать переменную в которой будут храниться разные

имена людей. Давайте убедимся что это set. Вот как вы видите в Python можно использовать не только квадратные скобки и круглые для объявления разных типов данных, но еще и фигурные, но и давайте посмотрим что получится, если мы проверим наличие какого-то элемента множеств. Вот мы получаем булевский ответ в зависимости от того, есть элемент или его нет. Можно элемент попробовать добавить и видеть, что все успешно, можно попробовать добавить тот элемент, который уже есть в множестве и увидеть, что конечно новый элемент не появился, потому что элемент уже есть, то есть дубликата множеств нет, можно удалять элементы. Можно попробовать добавить элемент какого-то другого типа по сравнению с теми, которые уже представлены. Это можно сделать с множеством точно также как с листом, но в примере, который мы сейчас запустим, ничего не получится, и мы получим сообщение об ошибке unhashable type list. Ну за такие алгоритмы и структур данных уже могли догадаться, что это связано с тем, что множество реализовано с помощью структур данных хеш-таблицы, и поэтому нужно, чтобы от добавляемого элемента можно было взять hash, то есть, чтобы элемент был неизменяемым.



**Словари(dict)**

Python тип данных dict — это словарь. Он пригодится нам, если нужно хранить какое-то отображение из одних элементов в другие, ну, например, для каждого слова уметь быстро получать его частоту в каком-нибудь тексте. При этом можно получать элемент по какому-то ключу и как выводить его на печать, так и обновлять значения. Ну и можно создавать множество, опять же, с помощью фигурных скобок, и можно хранить в множестве разные элементы и ключом, опять-таки, должен быть неизменяемый тип, ну то есть использовать

tuple в качестве ключа можно, использовать list в качестве ключа — неудачная затея.

