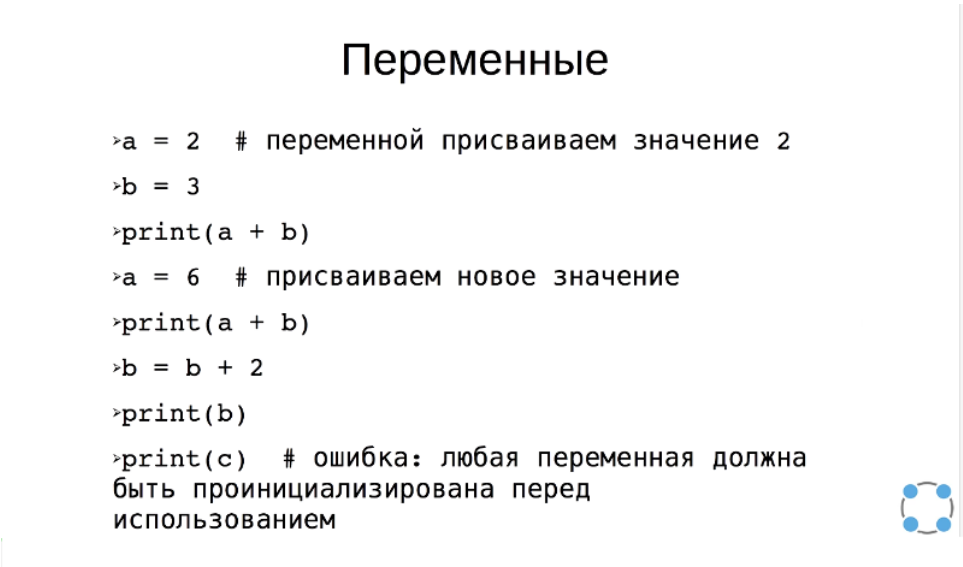
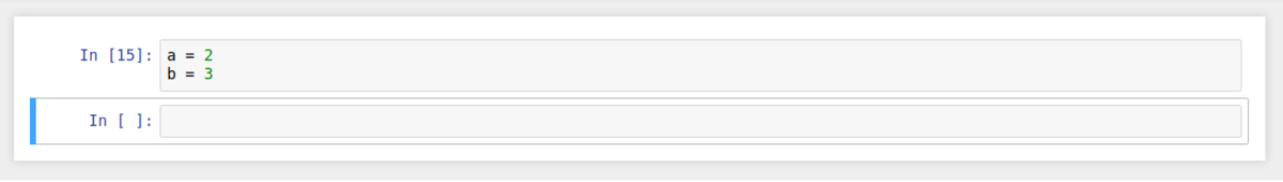
Step 1

Step on Stepik: https://stepik.org/lesson/2232/step/1

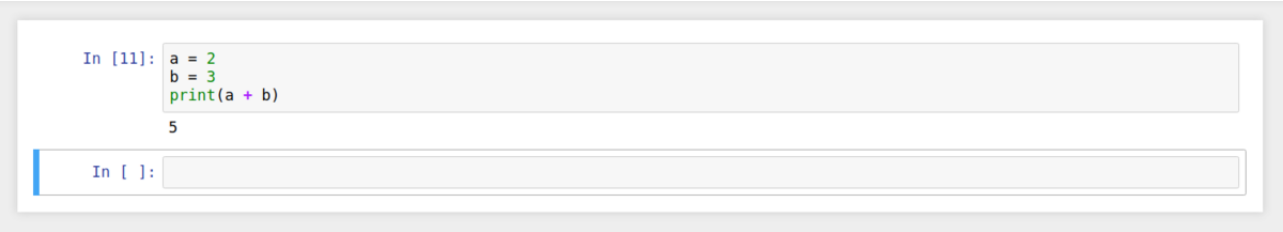
Мы изучили, что все объекты в Python характеризуется своим типом данных. Часто бывает удобно, а иногда просто необходимо для объекта создать имя и далее обращаться к этому объекту с помощью имени. Для этой цели в языках программирования придуманы переменные. Давайте посмотрим на примере, как используются переменные языке Python.



В первой строчке мы объявляем переменную **a** и записываем в нее значение 2. Мы можем создать еще одну переменную **b** и записать в нее произвольное значение. Например: 3. Далее мы можем эти переменные использовать в произвольных выражениях.



В третьей строчке мы используем функцию **print**, которой мы передаем выражение **(a + b)**. Что происходит при этом? Функция **print** обращается по имени к переменной **a**, смотрит ее значение - 2, далее смотрит на значение переменной **b**, равное 3 и складывает их. Таким образом, результатом этой операции будет выведено число 5.



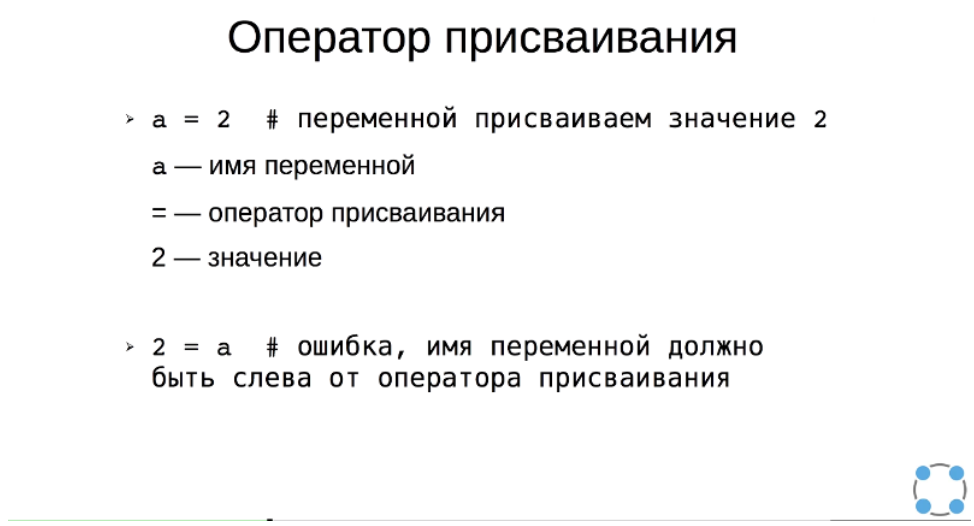
Мы можем изменить значение переменной, фактически связать переменную **a** с новым значением - 6. Если мы выведем после этого сумму **(a + b)**, то результат вычисления выражения соответственно изменится - (6 + 3) и будет равен 9.

Мы можем при изменении значения переменной использовать уже существующие переменные. Например, можем в переменную **b** записать значение **b + 2**. Таким образом в значение **b** будет записано значение (3 + 2), чтобы проверить это выведем значение переменой **b** с помощью функции **print** и убедимся в этом.

Обязательное условие - перед тем как использовать переменную мы должны ее проинициализировать - записать в нее некоторое значение. Если мы попробуем выполнить операцию **print(c)**, то у нас произойдет ошибка, потому что переменная **c** еще не была объявлена и в нее не было записано какое-либо значение.

```
In [11]: a = 2
         b = 3
         print(a + b)
5

In [12]: a = 6
         print(a + b)
9

In [13]: b = b + 2
         print(b)
5

In [14]: print(c)
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-14-5315f3e3adca> in <module>()
----> 1 print(c)

NameError: name 'c' is not defined

In [ ]:
```

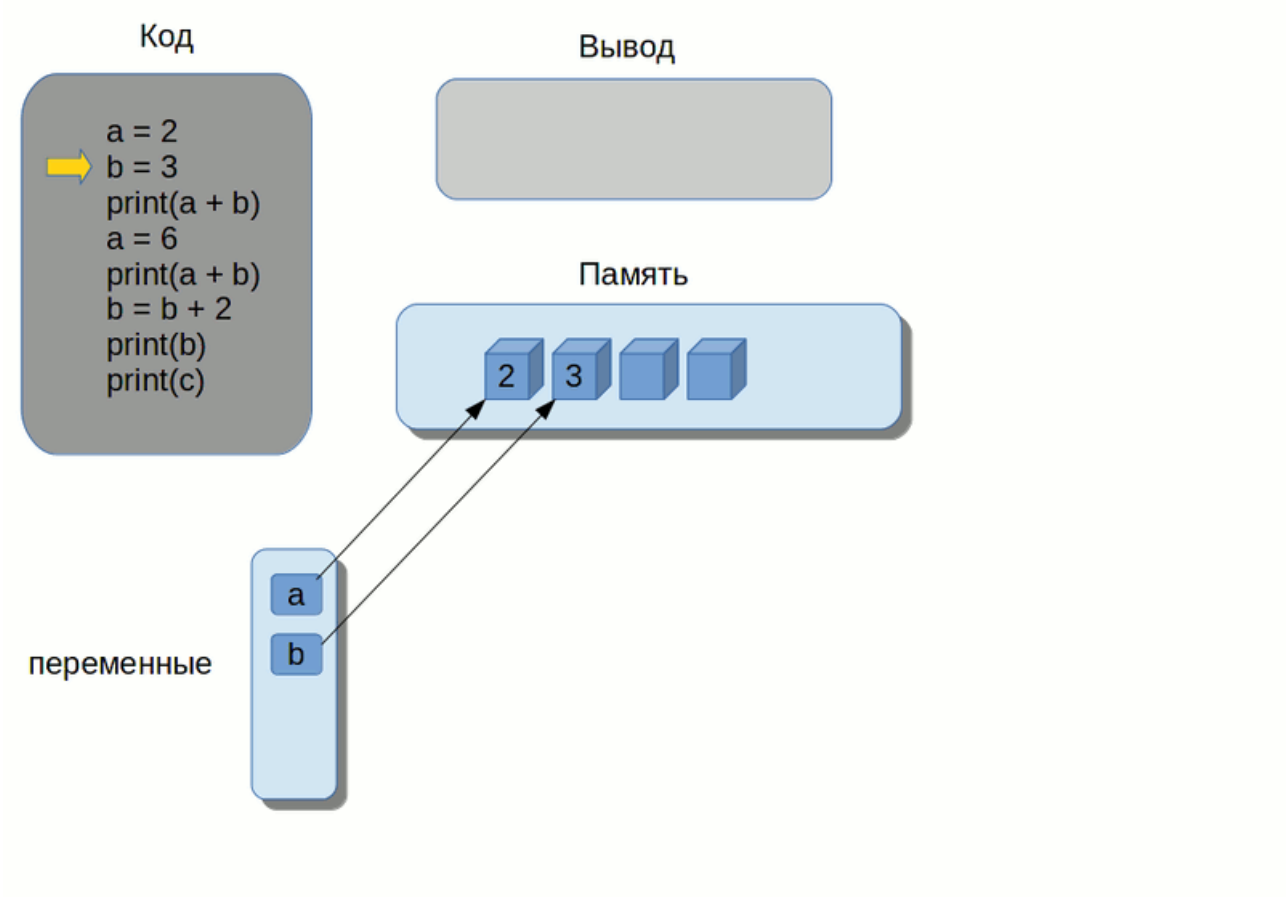
Переменные инициализируется с помощью оператора присваивания. Операция присваивания выглядит следующим образом **a = 2**.



Оператор присваивания - это просто значок равенства. Слева от оператора записывается имя переменной, а справа - значение, которое мы хотим связать с этой переменной. Во время выполнении программы существует специальный список переменных. Когда мы объявляем новую переменную, она добавляется в этот список и связывается с некоторым значением (каким-либо объектом), хранящемся в памяти. В данном случае этим объектом является число 2. Если мы захотим изменить значение переменной, то у нам необходимо имя переменной **a** связать с новым значением. Для этого напишем **a = 5**, после этого где-то в памяти выделится место

под новый объект (число 5) и вместо связи с числом 2 появится новая связь с этим объектом (числом 5). Оператор присваивания является не симметричным оператором, мы не можем написать **2 = a**. Имя переменой обязательно должно находиться слева от оператора присваивания.

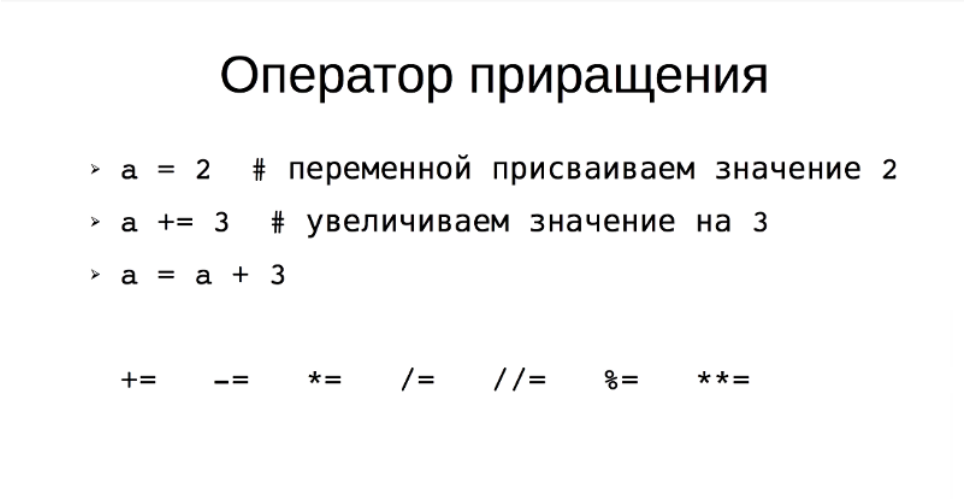
Давайте еще раз внимательно посмотрим на то что происходит в нашем примере.



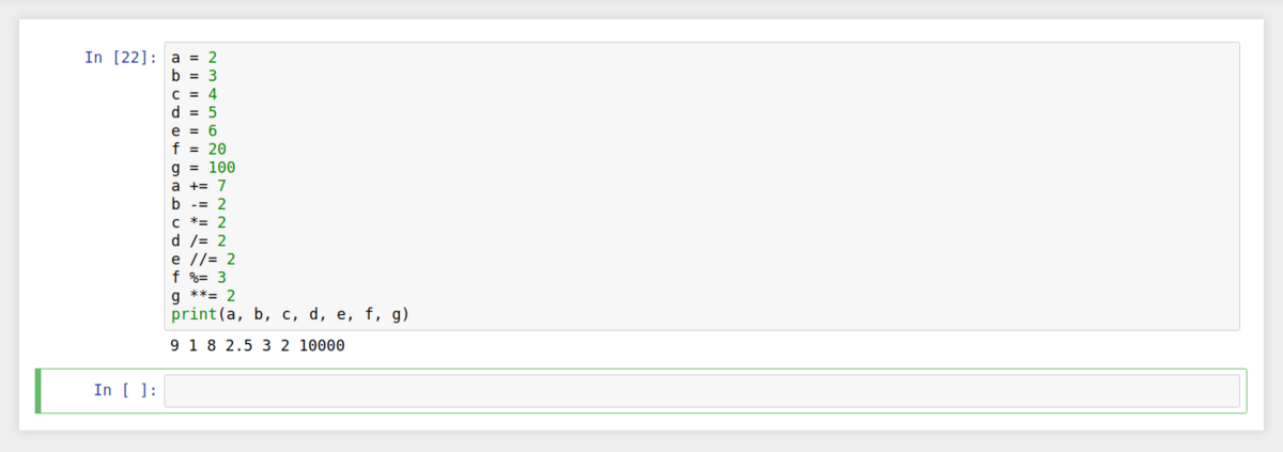
Изначально мы завели переменную **a** и связали ее со значением 2, далее мы добавили переменную **b** и связали ее со значением 3. Когда в выражениях встречаются эти переменные **a** и **b**, то на их место подставляются соответствующее значения: 2 и 3. После того как мы изменили значение переменной **a**, связав ее с числом 6, связь переменной **a** с числом 2 теряется и при последующем вычислении выражения **(a + b)** используются текущие значения переменных, соответственно 6 и 3.

Существует способ изменить переменную с помощью операторов приращения. Например, выражение **a += 3** позволяет увеличить

значение переменной **a** на 3. Фактически это тоже самое, что происходит с переменной **a** в присвоении **a = a + 3**.



Кроме оператора **"+="**, который увеличивает значение переменой на величину, стоящую справа, существует аналогичные операторы для вычитания, умножения, деления, целочисленного деления, взятия остатка и возведения в степень.



Имя переменной записывается, как мы уже говорили раньше, слева от оператора присваивания. Выбор имени может быть практически произвольным, но существует ряд ограничений.

Имя переменной

```
a = 2  # переменной присваиваем значение 2
```

a — имя переменной

- Имя переменной
  - может состоять из букв (строчных и прописных), цифр, подчеркивания _
  - должно начинаться с буквы или подчеркивания
  - не должно являться ключевым словом
  - регистр букв имеет значение

Имя переменной может содержать буквы (строчные и прописные), цифры и знак подчеркивания. Должно начинаться с буквы или подчеркивания. Не должно являться ключевым словом. Например, мы не можем завести переменную и назвать ее **True**, потому что **True** - это ключевое слово, обозначающее истину для логических тиаов данных. Регистр букв имеет значение. Таким образом, переменная **a** и переменная **A**, это две разных переменных. Если вы завели переменную **a** и далее в программе вызовете **print(A)**, возникнет ошибка. Интерпретатор сообщит, что ничего не знает про переменную **A**.

В языке Python одна и та же переменная может связываться с объектами разных типов.

Динамическая типизация

```
a = 2
a = 'abacaba'
a = foo()
```

`type(a)` - ?

Мы можем в переменную **a** записать число 2, после этого мы можем изменить значение **a** на какую нибудь строку или записать в **a** результат выполнения функции, который будет еще каким-то третьим типом данных. Все это может происходить в одной программе. И ничего страшного не случится. Такой тип типизации называется динамической типизацией, когда у нас переменная может менять тип по ходу выполнения программы. Иными словами, мы не можем с уверенностью сказать какой тип будет у переменной **a** перед тем как мы запустим программу.

Вообще, не совсем корректно говорить, что переменная имеет такой-то тип. Более правильным будет утверждение, что в конкретный момент выполнения программы конкретная переменная имеет такой-то тип, то есть связана с конкретным объектом этого типа. Этим Python отличается от многих других языков программирования с так называемой статической типизации, где тип переменной жестко зафиксирован.