УНИВЕРСИТЕТ СИНЕРГИЯ

ТИПЫ ПЕРЕМЕННЫХ

Модуль 2. Занятие 7.

Тема и цель

Тема:

Основы работы с переменными разных типов. Классификация переменных.

Цель занятия:

Изучение способов работы с переменными разных типов и классификация переменных.

Глоссарий

1. **Переменная (в программировании)** — это каким-либо образом проименованная и/или адресованная область физической или виртуальной памяти, предназначенная для хранения данных (значений).

Подтемы

1. Перечислите управляющие последовательности в строке.
2. Что позволяет узнать функция **type()?**
3. В типе данных **float** используется точка или запятая?
4. Как произвести преобразование типа данных?
5. Перечислите типы переменных.

Как создать переменную

Представьте, что нам нужно напечатать на экране фразу Father! два раза.
Эту задачу можно решить так:

- **print** ('Father!')
- **print** ('Father!')

Как создать переменную

А можно поступить по-другому. Чтобы не копировать выражение, достаточно создать с ним переменную:

```
# greeting — переводится как «приветствие»
greeting = 'Father!'
print (greeting)
print (greeting)
#=> 'Father!'
#=> 'Father!'
```

Как создать переменную

Количество создаваемых переменных неограниченно. Большие программы содержат десятки и сотни тысяч имен переменных. Вот как выглядят две переменные внутри одной программы:

```
greeting1 = 'Father!'
print (greeting1)
print (greeting1)

greeting2 = 'Mother!'
print (greeting2)
print (greeting2)
```

Как изменить переменную

Например:

```
# greeting - переводится как приветствие
greeting = 'Father!'
print (greeting)  # => Father!

greeting = 'Mother!'
print (greeting)  # => Mother!
```

Какие ошибки часто допускают с переменными

Порядок инструкций в коде с переменными имеет огромное значение. Поэтому переменную нужно определить до первого использования. Ниже пример ошибки, которую часто допускают новички:

```
print (greeting)
greeting = 'Father!'
```

Как работают константы

Некоторые данные никогда не меняются — например, математические постоянные. Возьмем для примера число π. Оно всегда равно 3.14 и не может измениться. Чтобы обратиться к подобным данным, в Python используют константы:

```
PI = 3.14
print(PI)  # => 3.14
```

Константа создается так же, как переменная. Разница только в том, что константы принято именовать заглавными буквами и с _ в качестве разделителя между словами. Константа, как и переменная, может использоваться в любом выражении.

Как работают константы

Предположим, что к этому значению нужно обратиться для выполнения вычислений. Вместо того, чтобы каждый раз вводить его, можно просто использовать переменную. Вот так:

```
>>> sun_to_earth = 149597970
>>> sun_to_earth = sun_to_earth + 1
>>> print(sun_to_earth)
149597971
```

Эти данные могли бы быть и куда объёмнее. В них, например, могло бы быть 100 цифр, а обращаться к данным, возможно, нужно было бы в 100 разных местах. Таким образом польза от присваивания имени значению очевидна.

Как работают константы

Разберем код примера:

1. Сначала создается переменная sun_to_earth, и ей присваивается значение 149597970.
2. Переменной sun_to_earth присваивается новое значение — 149597971.
3. Выполняется функция print, которая выводит текущее значение переменной, то есть, 149597971.

Как работают константы

```
>>> dogs_name = "Шарик"
>>> dogs_kind = "Сенбернар"
>>> print("Его зовут " , dogs_name , ". Он ", dogs_kind, ".", sep="")
Его зовут Шарик. Он Сенбернар.
```

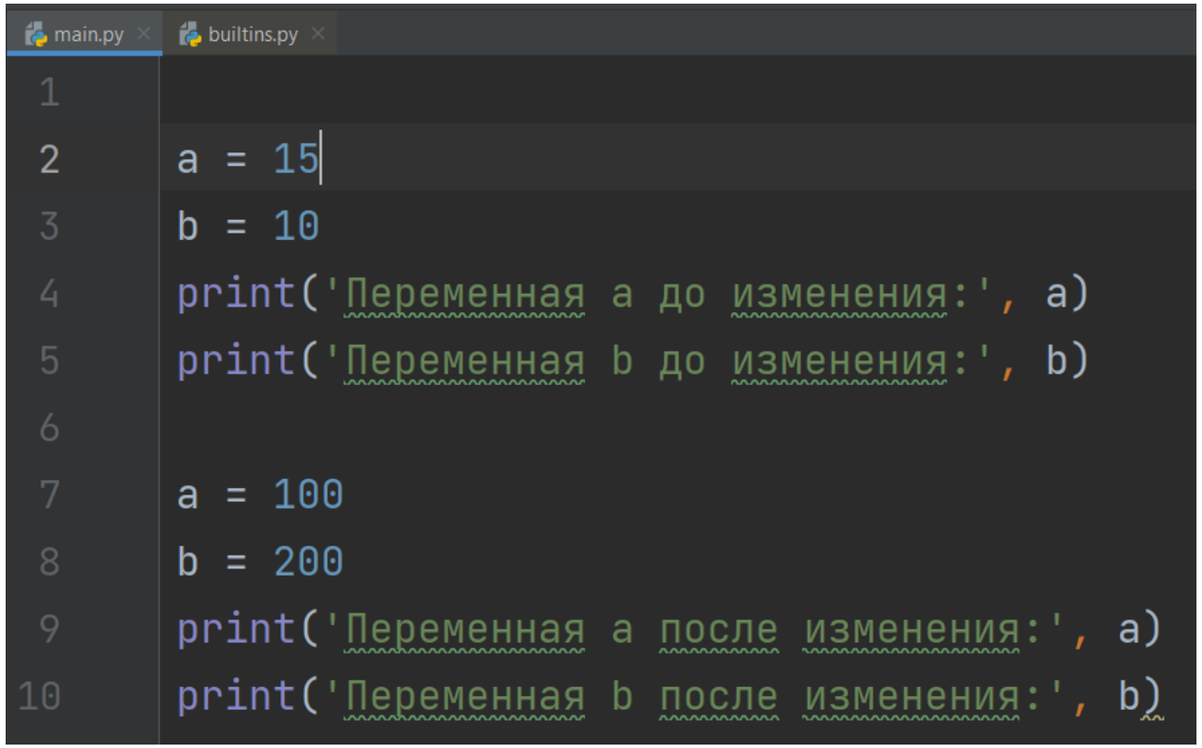
Это отличный пример применения не только переменной, но и функции print, ведь в последней используются сразу текст и значение переменной. Плюс, названия переменных выбраны очень удачно. Названия для переменных — тема субъективная, поэтому достаточно использовать то, что удобно для вас.

Как работают константы

В одной строке можно присвоить сразу несколько переменных. Вот пример:

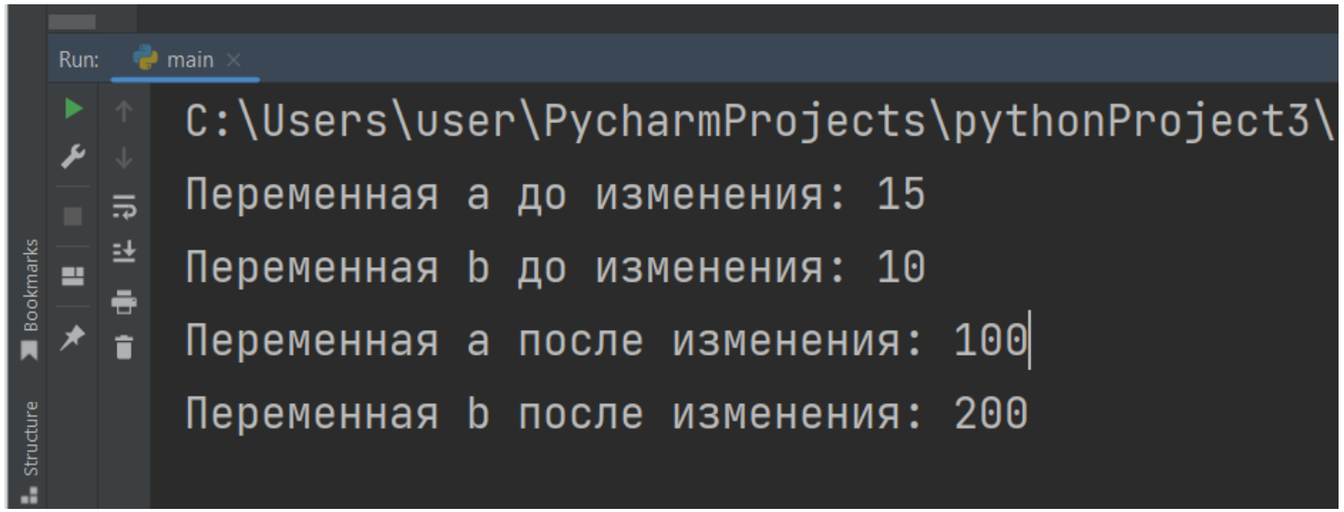
```
>>> i, j, k = "Hello", 55, 21.0765
>>> print(i, j, k)
Hello 55 21.0765

>>> dogs_name = "Шарик"
>>> dogs_kind = "Сенбернар"
>>> print("Его зовут " , dogs_name , ". Он ", dogs_kind, ".", sep="")
Его зовут Шарик. Он Сенбернар.
```

Как работают константы



Отличительной особенностью языка Python является то, что в течении работы программы можно менять значение переменной:

Как работают константы

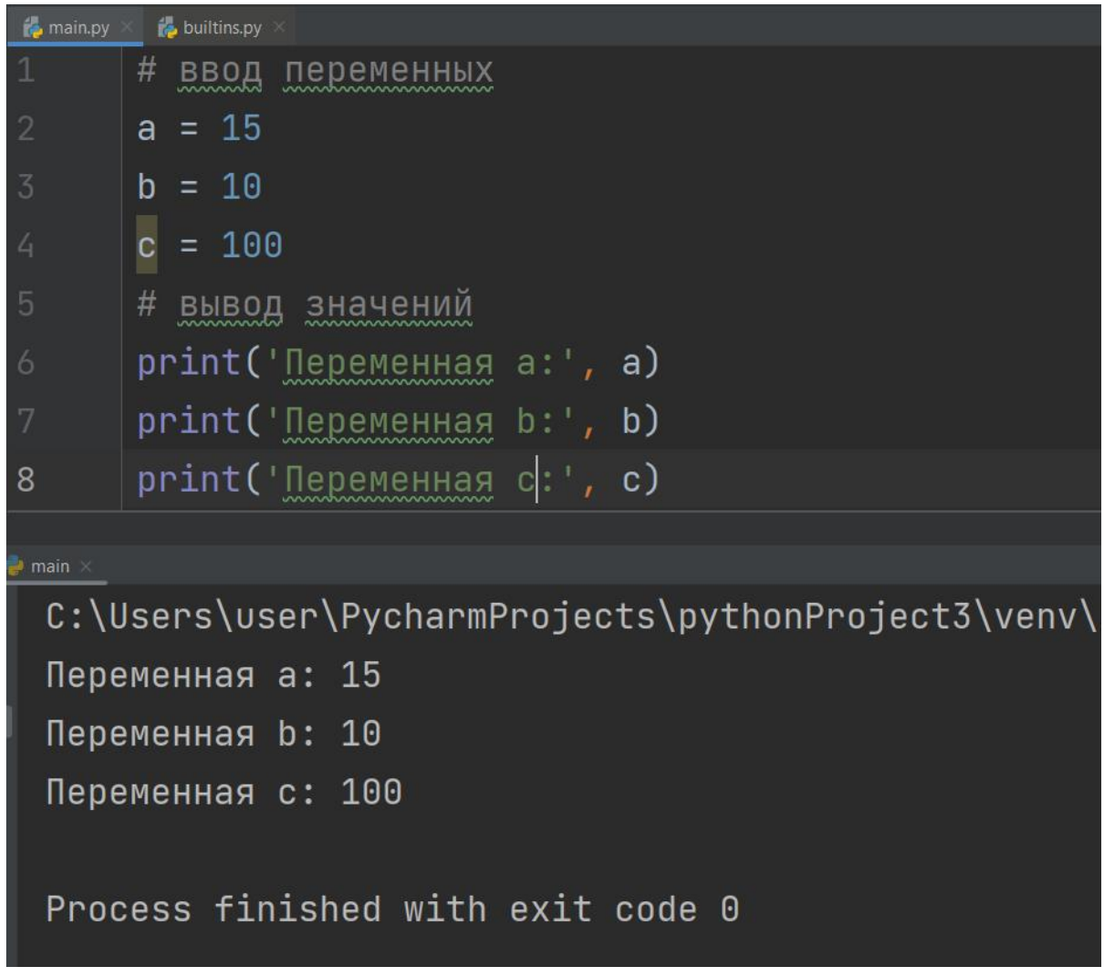


Как работают константы

Самыми базовыми типами данных являются следующие:

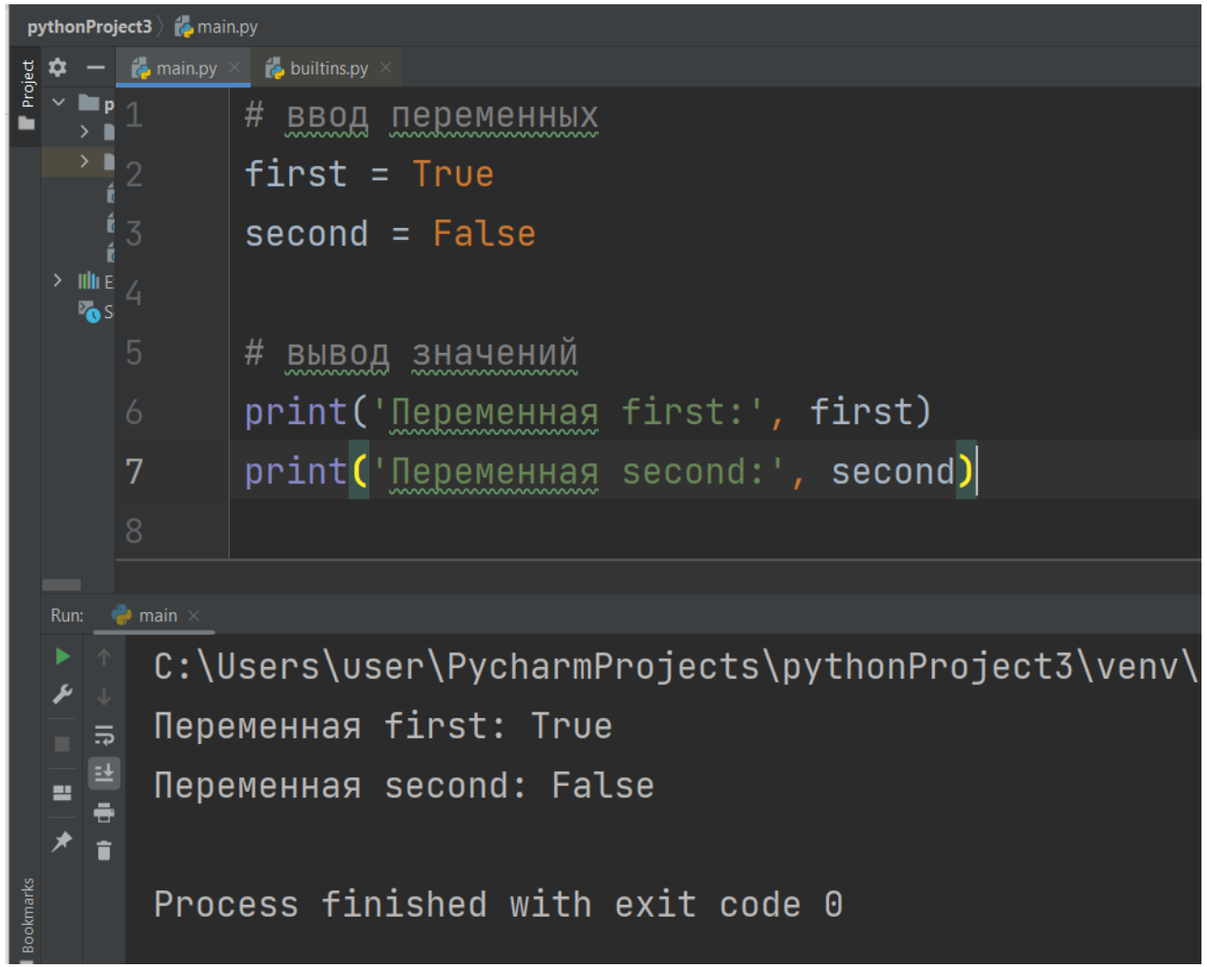
- **int** (целые числа)
- **float** (числа с плавающей запятой)
- **str** (строковой тип данных)
- **complex** (комплексные числа)
- **bool** (логический тип данных)

Типы данных
int и bool



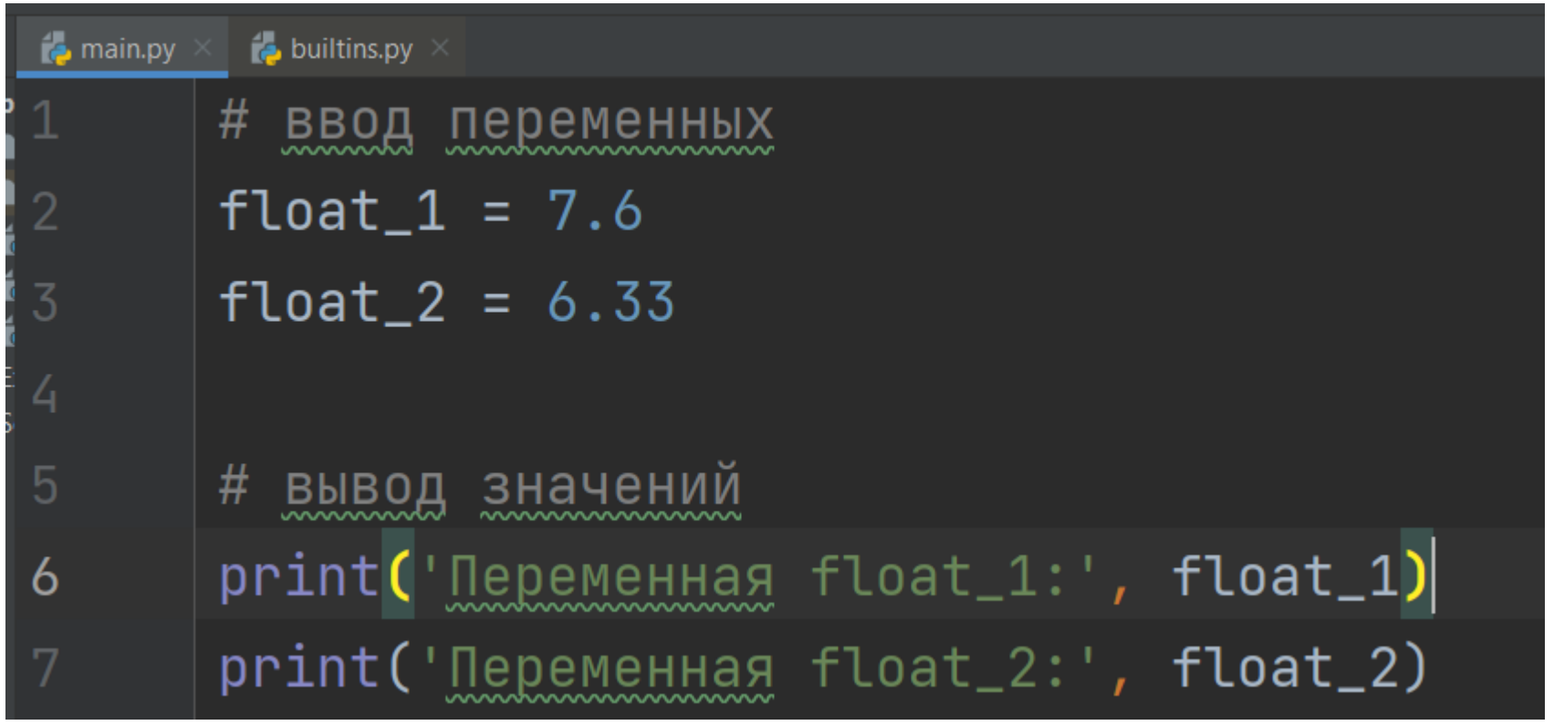
Тип данных **int** представляет целые числа (например, 1, 5, 10, 100). Приведем пример:

Типы данных
int и bool

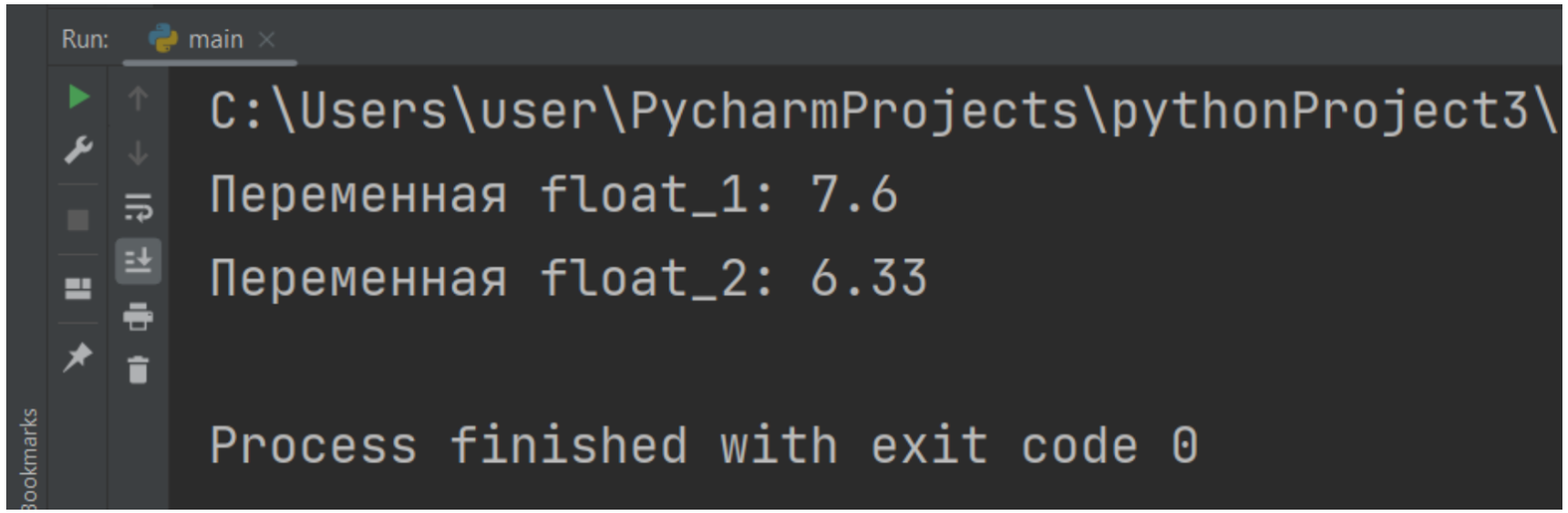


В тоже время, **False,** наоборот, показывает, что переменная или группа переменных являются ложными:

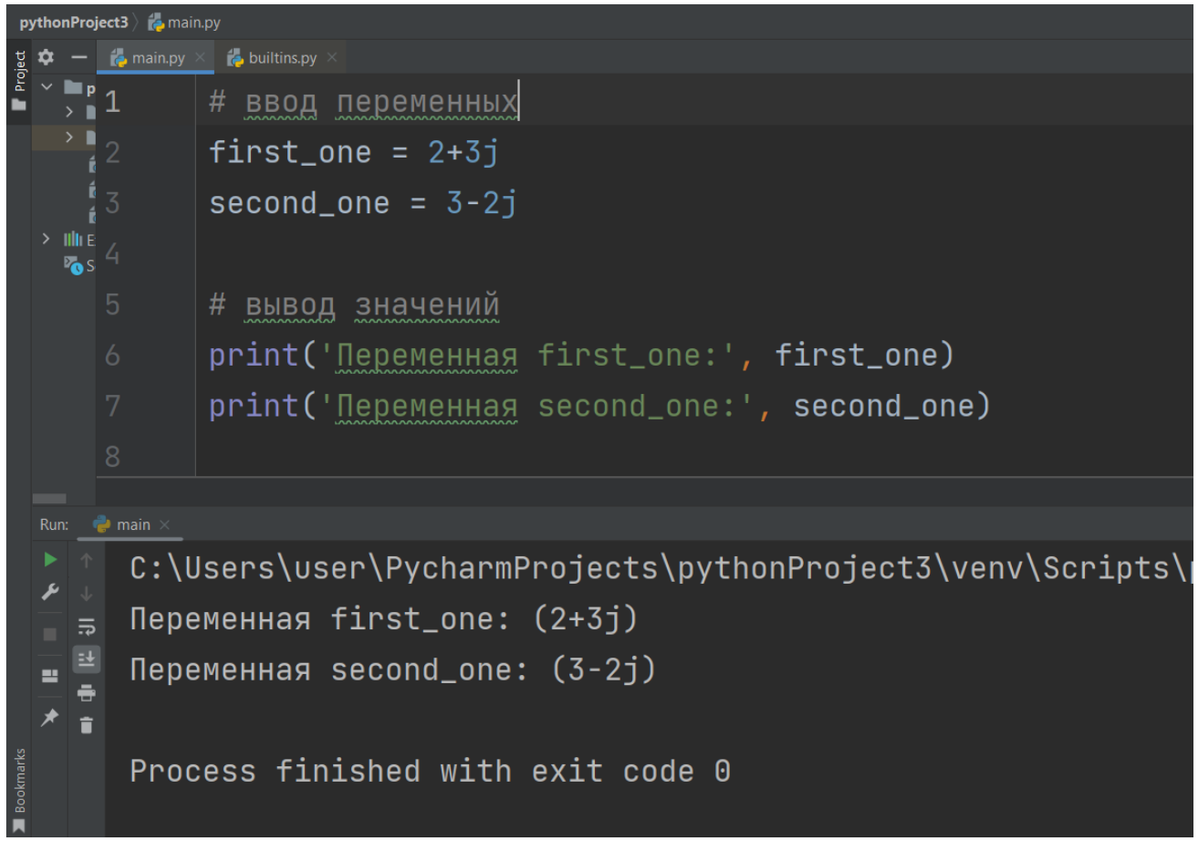
Дробный (float) и комплексный (complex) типы данных



Дробный (float) и комплексный (complex)
типы данных

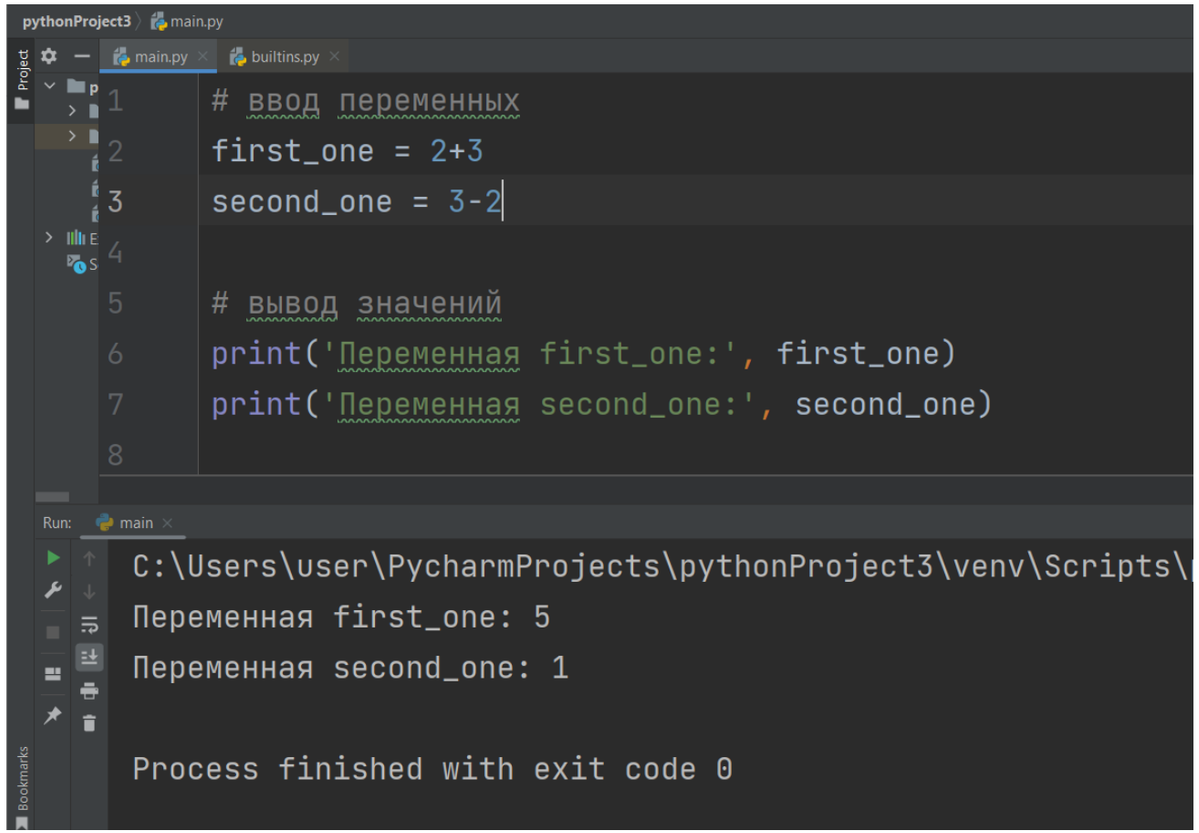


Дробный (float) и комплексный (complex) типы данных



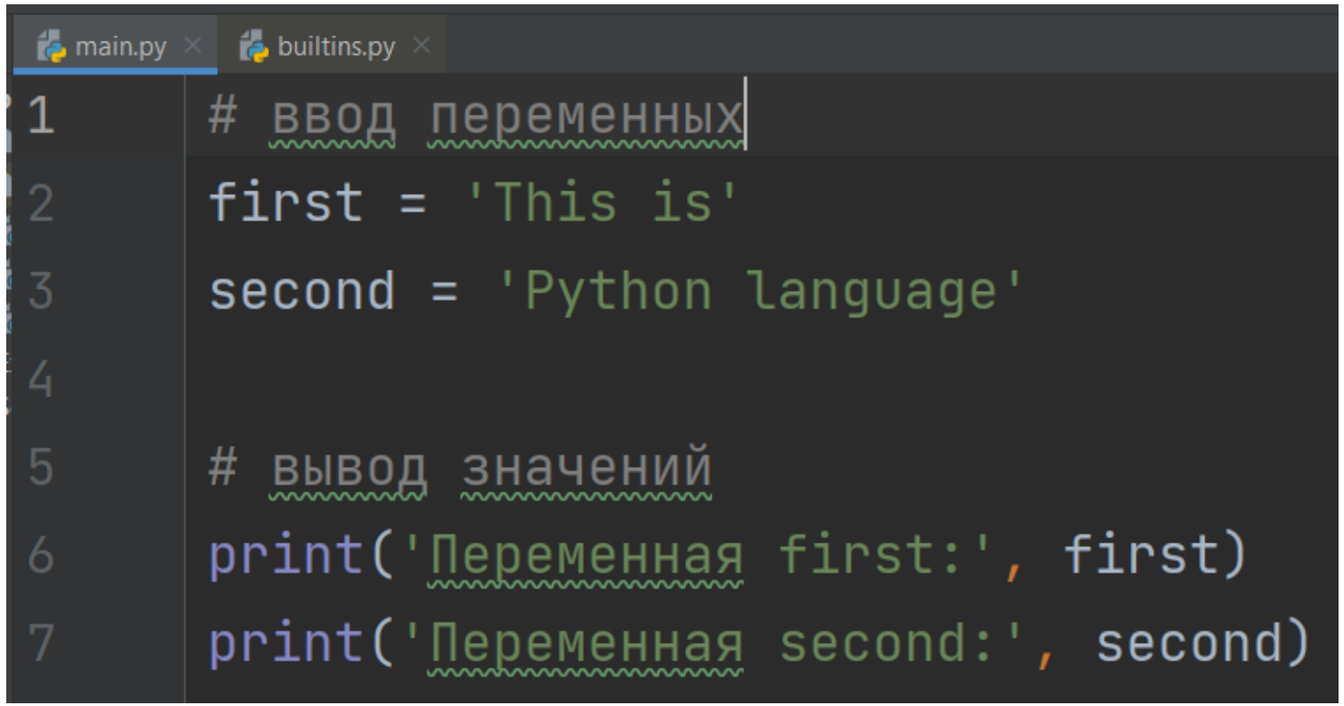
Как правило, числа с типом данных **float** могут иметь до 18 значимых символов.
Тип **complex** представляет комплексные числа в следующем формате: *вещественная часть + мнимая часть * j (j - оператор комплексных чисел):*

Дробный (float) и комплексный (complex) типы данных

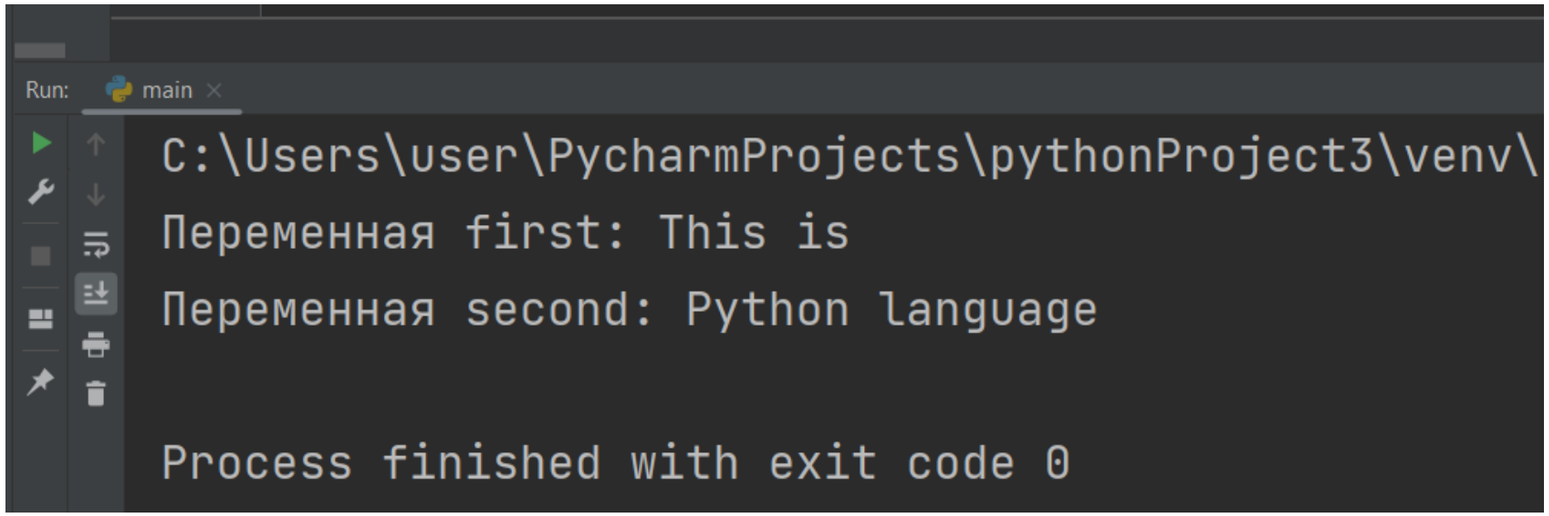


Стоит отметить, что если не задавать оператор комплексных чисел (j), то числа просто будут складываться/вычитаться друг из друга:

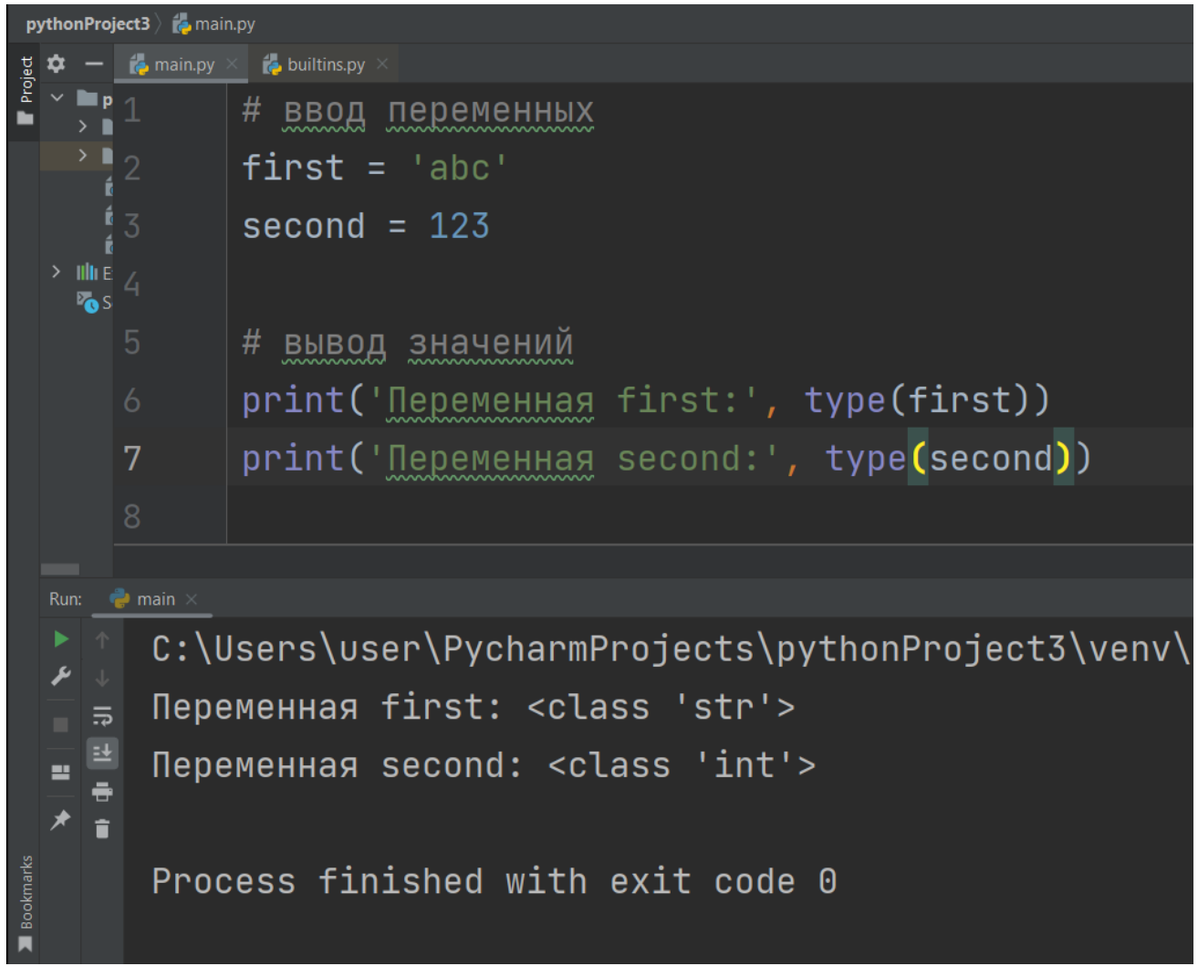
Строковый
тип данных (str)



Строковый
тип данных (str)

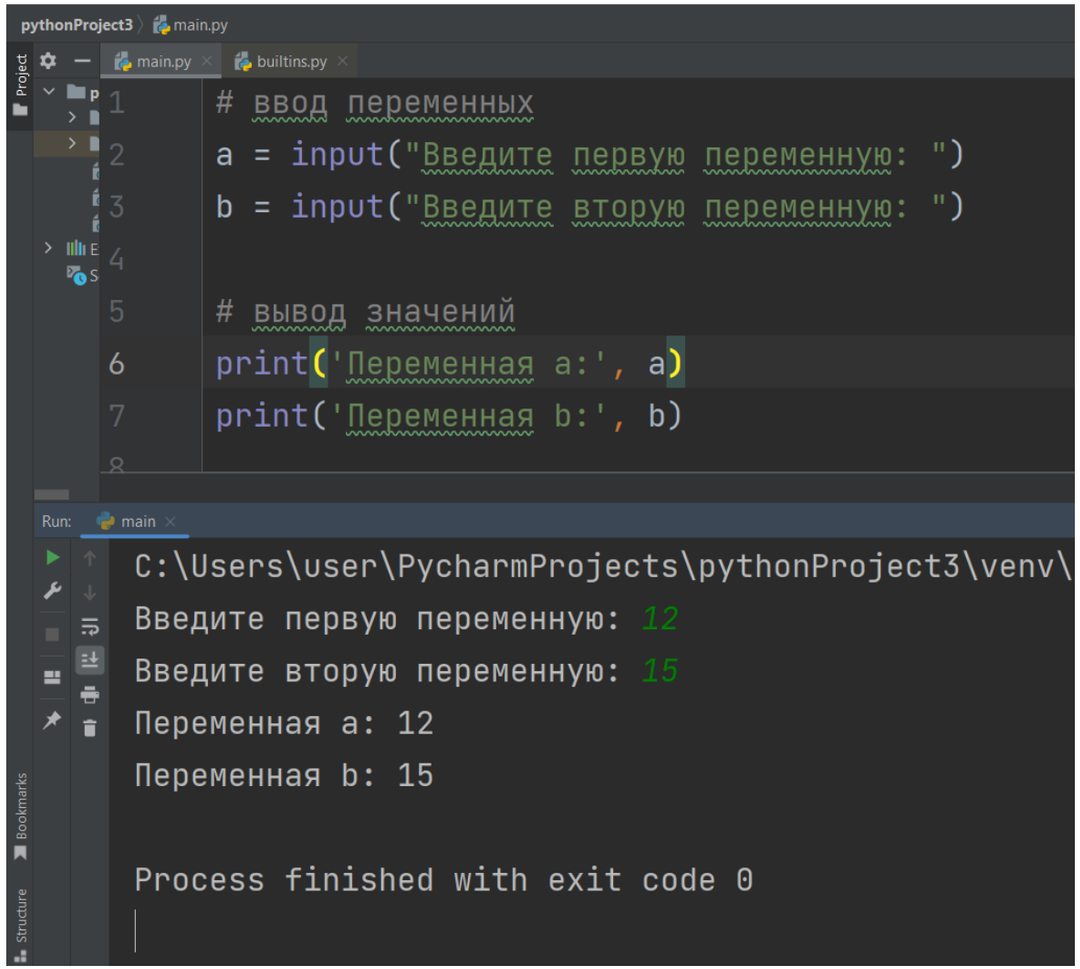


Динамическая типизация



Python является языком с динамической типизацией. Следовательно, любая переменная не привязывается к определенному типу. Тип переменной всегда определяется исходя из значения, которое ей было присвоено. Таким образом, при присвоении числа с плавающей точкой определяется тип переменной **float.** Тип данных можно просмотреть при помощи команды **type():**

Консольный ввод



Как мы уже разбирали, консольный ввод можно реализовать при помощи простой команды **input():**

Практические задачи

Задача 1

У паши дома банки для круп в форме кубов разного размера (с разными ребрами). Вчера к нему зашел сосед и попросил одолжить риса. Паша пересыпал немного из одной банки с ребром **a** в маленькую с ребром **b** и отдал соседу.

Даны **a** и **b.** Определи, какой объем риса остался у Паши.

Пример:

Ввод: 5 Вывод: 98
3

Решение

Для начала необходимо определить объем банки с ребром **a:**
Va = a * a * a = a ** 3

```
a = int(input())
b = int(input())
print(a**3 - b**3)
```

Объём банки с ребром **b:**
Vb = b * b * b = b ** 3

Итоговый ответ = Va - Vb

```
5
3
98

Process finished with exit code 0
```

Задача 2

Напишите программу-эхо, которая выводит последнюю (крайнюю правую) цифру заданного числа.

Пример:

Ввод: 23214 Вывод: 4

Решение

Чтобы решить данную задачу стоит вспомнить операторы в Python. Разберем задачу на конкретном примере (a = 23214, x - неизвестное число). Если решение сразу не приходит, можем применить последовательно все, которые знаем:

сложение: 23214 + x — получаем большее, не приближает нас к ответу
вычитание: 23214 - x — получаем меньшее, не приближает нас к ответу
умножение: 23214 * x — получаем число сильно большее
деление: 23214 / x — число сократится в x раз, уже ближе

Поскольку мы работаем с десятичными числами, представим, что делим на 10:
23214 / 10 = 2321,4
Деление целочисленное: 23214 // 10 = 2321

Нам удалось удалить правую цифру из числа, приближено к ответу, однако не в ту сторону
деление с остатком: 23214 % 10 = 4

Деление с остатком на 10 в 10сс даёт нам крайнюю правую цифру, ответ найден.

Решение

```
1    a = int(input())
2    print(a % 10)
```

```
23214
4

Process finished with exit code 0
```

Вопросы

1. Что такое **строка?**
2. Для чего используется тип переменных **str** в языке Python?
3. Что такое команда **input()** и как ее можно применять в языке Python?
4. Как можно работать с несколькими списками одновременно в языке Python?
5. Для чего служит ключевое слово **local** в языке Python?