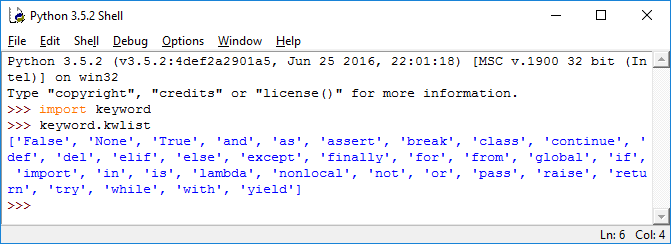
**Переменные**

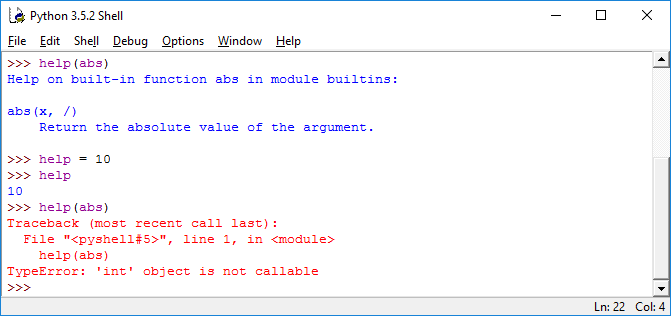
Все данные в языке **Python** представлены объектами. Каждый объект имеет тип данных и значение. Для доступа к объекту предназначены переменные. При инициализации в переменной сохраняется ссылка (адрес объекта в памяти компьютера) на объект. Благодаря этой ссылке можно в дальнейшем изменять объект из программы.

**2.1 Именование переменных**

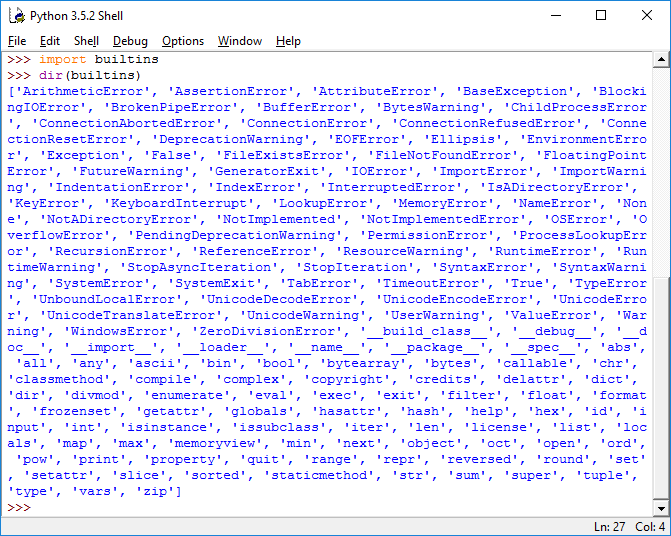
Каждая переменная должна иметь уникальное имя, состоящее из латинских букв, цифр и знаков подчеркивания, причём имя переменной не может начинаться с цифры. Кроме того, следует избегать указания символа подчеркивания в начале имени, т. к. идентификаторы с таким символом имеют специальное значение. Например, имена, начинающиеся с символа подчёркивания, не импортируются из модуля с помощью инструкции **from module import \***, а имена, имеющие по два символа подчёркивания в начале и конце, для интерпретатора имеют особый смысл.

В качестве имени переменной нельзя использовать ключевые слова. Получить список всех ключевых слов позволяет код, приведенный ниже:

 Помимо ключевых слов следует избегать совпадений со встроенными идентификаторами. В отличие от ключевых слов, встроенные идентификаторы можно переопределять, но дальнейший результат может стать для вас неожиданным.

>»

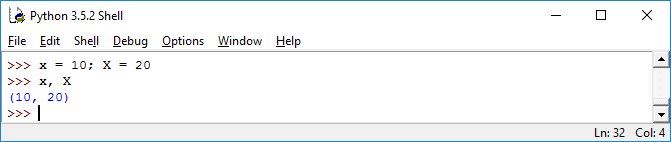
В этом примере с помощью встроенной функции **help()** получаем справку по функции **abs**(). Далее переменной **help** присваиваем число **10**. После переопределения идентификатора больше нельзя пользоваться функцией **help()**, т. к. это приведет к выводу сообщения об ошибке. По этой причине лучше избегать имен, совпадающих со встроенными идентификаторами. Очень часто подобная ошибка возникает при попытке назвать переменную, в которой предполагается хранение строки, именем **str**. Вроде бы логично, но **str** является часто используемым встроенным идентификатором и после такого переопределения поведение программы становится непредсказуемым. В редакторе **IDLE** встроенные идентификаторы подсвечиваются фиолетовым цветом. Обращайте внимание на цвет переменной, он должен быть **черным**. Если вы заметили, что переменная подсвечена, то название переменной следует обязательно изменить. Получить полный список встроенных идентификаторов позволяет следующий код:

Правильные имена переменных: **х, y1, strName, str\_name**.

Неправильные имена переменных: **1y, ИмяПеременной**.

Последнее имя неправильное, т. к. в нем используются русские буквы. Хотя на самом деле такой вариант также будет работать, но лучше русские буквы все же не применять.

При указании имени переменной важно учитывать регистр букв: х и X - разные переменные:

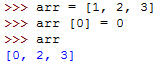


**2.2 Типы данных**

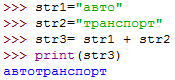
В **Python 3** объекты могут иметь следующие типы данных:

* **bool** - логический тип данных. Может содержать значения **True** или **False**, которые ведут себя как числа **1** и **0** соответственно.
* **NoneType**- объект со значением **None** (обозначает отсутствие значения). В логическом контексте значение **None** интерпретируется как **False**.
* **int** - целые числа. Размер числа ограничен лишь объемом оперативной памяти:
* **float**- вещественные числа.
* **complex** - комплексные числа.
* **str** - **Unicode**-cтроки.
* **bytes** - неизменяемая последовательность байтов.
* **bytearray** - изменяемая nпоследовательность байтов.
* **list** - списки. Тип данных **list** аналогичен массивам в других языках программирования.
* **tuple** – кортежи.
* **dict** - словари. Тип данных **dict** аналогичен ассоциативным· массивам в других языках программирования.
* **set** - множества (коллекции уникальных объектов).
* **frozenset** - не изменяемые множества.
* **ellipsis** - обозначается в виде трех точек или слова **Ellipsis**. Тип **ellipsis** используется в расширенном синтаксисе получения среза.
* **function** – функции.
* **modu1e** – модули.
* **type** - классы и типы данных. Не удивляйтесь! Все данные в языке **Python** являются объектами, даже сами типы данных!

Основные типы данных делятся на **изменяемые** и **неизменяемые**. К изменяемым типам относятся **списки**, **словари** и тип **bytearray**. Пример изменения первого элемента списка:



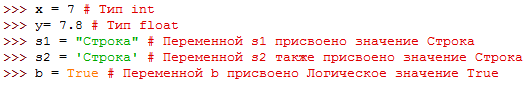
К неизменяемым типам относятся **числа**, **строки**, **кортежи** и тип **bytes**. Например, чтобы получить строку из двух других строк, необходимо использовать операцию конкатенации, а ссылку на новый объект присвоить переменной:



Кроме того, типы данных делятся на последовательности и отображения. К последовательностям относятся **строки**, **списки**, **кортежи**, типы **bytes** и **bytearray**, а к отображениям - **словари**.

**2.3. Присваивание значения переменным**

В языке **Python** используется динамическая типизация. Это означает, что при присваивании переменной значения интерпретатор автоматически относит переменную к одному из типов данных. Значение переменной присваивается с помощью оператора **=** таким образом:



В одной строке можно присвоить значение сразу нескольким переменным:



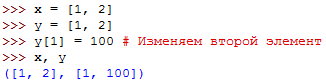
После присваивания значения в переменной сохраняется ссылка на объект, а не сам объект. Это обязательно следует учитывать при групповом присваивании. Групповое присваивание можно использовать для **чисел**, **строк** и **кортежей**, но для **изменяемых** объектов этого делать **нельзя**. Пример:



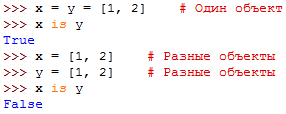
В этом примере мы создали список из двух элементов и присвоили значение переменным x и у. Теперь попробуем изменить значение в переменной у:



Как видно из примера, изменение значения в переменной у привело также к изменению значения в переменной х. Таким образом, обе переменные ссылаются на один и тот же объект, а не на два разных объекта. Чтобы получить два объекта, необходимо производить раздельное присваивание:



Проверить, ссылаются ли две переменные на один и тот же объект, позволяет оператор **is**. Если переменные ссылаются на один и тот же объект, то оператор **is** возвращает значение **True**:

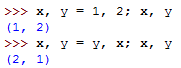


Помимо группового присваивания язык **Python** поддерживает позиционное присваивание. В этом случае переменные указываются через запятую слева от оператора =, а значения через запятую справа. Пример позиционного присваивания:

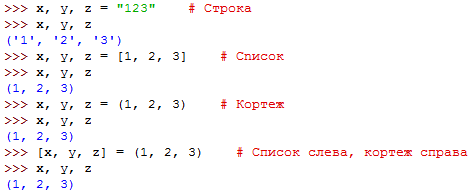


С помощью позиционного присваивания можно поменять значения переменных местами.

Пример:

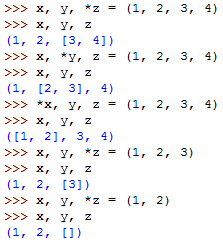


По обе стороны оператора = могут быть указаны последовательности. Напомним, что к последовательностям относятся **строки**, **списки**, **кортежи**, типы **bytes** и **bytearray**. Пример:



Обратите внимание на то, что количество элементов справа и слева от оператора = должно совпадать, иначе будет выведено сообщение об ошибке.

В **Python 3** существует возможность сохранения в переменной списка, состоящего из лишних элементов, при несоответствии количества элементов справа и слева от оператора =. Для этого перед именем переменной указывается звездочка (\*). Пример:



Как видно из примера, переменная, перед которой указана звездочка, всегда содержит список. Если для этой переменной не хватило значений, то ей присваивается пустой список. Следует помнить, что звездочку можно указать только перед одной переменной. В противном случае возникнет неоднозначность и интерпретатор выведет сообщение об ошибке.

**2.4. Проверка типа данных**

**Python** в любой момент времени изменяет тип переменной в соответствии с данными, хранящимися в ней. Пример:

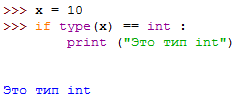


Определить, на какой тип данных ссылается переменная, позволяет функция **type(<Имя переменной>)** :

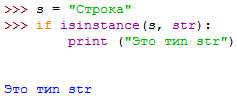


Проверить тип данных переменной можно следующими способами:

* сравнить значение, возвращаемое функцией type(), с названием типа данных:



* проверить тип с помощью функции **isinstance()** :



**2.5. Преобразование типов данных**

Как вы уже знаете, в языке **Python** используется динамическая типизация. После присваивания значения в переменной сохраняется ссылка на объект определенного типа, а не сам объект. Если затем переменной присвоить значение другого типа, то переменная будет ссылаться на другой объект, и тип данных соответственно изменится. Таким образом, тип данных в языке **Python**- это характеристика объекта, а не переменной. Переменная всегда содержит только ссылку на объект.

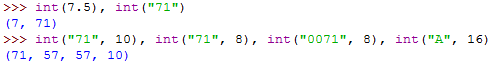
После присваивания переменной значения над объектом можно производить операции, предназначенные для этого типа данных. Например, строку нельзя сложить с числом, т. к. это приведет к выводу сообщения об ошибке.

Для преобразования типов данных предназначены следующие функции:

* **bool( [ < Объект> ] )** - преобразует объект в логический тип данных. Примеры:

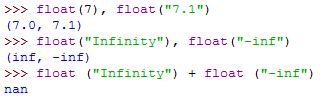


* **int ( [<Объект> [, <Система счисления>] ] )** - преобразует объект в число. Во втором параметре можно указать систему счисления (значение по умолчанию - 10). Примеры:

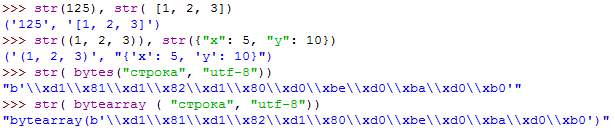


Если преобразование невозможно, то возбуждается исключение.

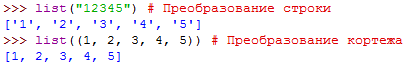
* **float ( [<Число или строка> ] )** - преобразует целое число или строку в вещественное число. Примеры:



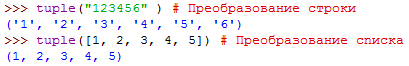
* **str ( [<Объект> ] )** - преобразует объект в строку. Примеры:

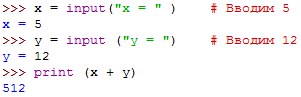


* **list (<Последовательность>)** - преобразует элементы последовательности в список. Примеры:

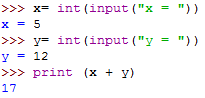


* **tuple (<Последовательность>)** - преобразует элементы последовательности в кортеж:



В качестве примера рассмотрим возможность сложения двух чисел, введенных пользователем. Как вы уже знаете, вводить данные позволяет функция input(). 

Результатом выполнения этого· скрипта будет не число, а строка "**512**". Таким образом, следует запомнить, что функция **input()** возвращает результат в виде строки. Чтобы просуммировать два числа, необходимо преобразовать строку в число



В этом случае мы получим число **17**, как и должно быть. Однако если пользователь вместо числа введет строку, то программа завершится с фатальной ошибкой. Как обработать ошибку, мы будем рассматривать по мере изучения языка.

**2.6. Удаление переменной**

Удалить переменную можно с помощью инструкции **del**:

**del <Переменная1>[, ... , <ПеременнаяN>]**

Пример удаления одной переменной:



Пример удаления нескольких переменных:

