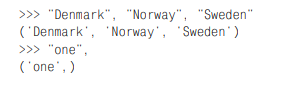
Лекция 13

Коллекции данных

Введение

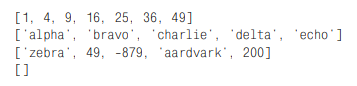
Часто бывает удобно хранить целую коллекцию элементов данных. В языке Python для этого имеется несколько типов коллекций, способных хранить элементы данных, включая ассоциативные массивы и множества. Но в этом разделе мы рассмотрим только два типа кол- лекций: tuple (кортежи) и list (списки). Кортежи и списки в языке Python могут использоваться для хранения произвольного числа элементов данных любых типов. Кортежи относятся к разряду неизменяемых объектов, поэтому после создания кортеж нельзя изменить. Списки относятся к разряду изменяемых объектов, поэтому мы легко можем вставлять и удалять элементы списка по своему желанию.

Кортежи создаются с помощью запятых (,), как показано ниже:



При выводе кортежа интерпретатор Python заключает его в круглые скобки. Многие программисты имитируют такое поведение и заключают литералы кортежей в круглые скобки. Если создается кортеж с одним элементом, то даже при наличии круглых скобок мы обязаны использовать запятую, например: (1,). Пустой кортеж создается с помощью пустых круглых скобок (). Запятая также используется для отделения аргументов при вызове функции, поэтому, если в качестве аргумента требуется передать литерал кортежа, мы должны заключать его в круглые скобки, чтобы избежать неоднозначности.

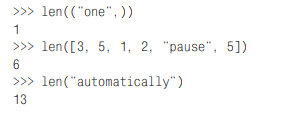
Ниже приводятся несколько примеров списков:



Как показано здесь, списки могут создаваться с помощью квадратных скобок. Четвертый список в примере – это пустой список. Списки и кортежи хранят не сами элементы данных, а ссылки на объекты. При создании списков и кортежей (а также при добавлении новых элементов в списки) создаются копии указанных ссылок на объекты. В случае значений-литералов, таких как целые числа и строки, в памяти создаются и инициализируются объекты соответствующих типов, затем создаются ссылки, указывающие на эти объекты, и эти ссылки помещаются в список или в кортеж. Как и все остальное в языке Python, коллекции данных – это объекты: благодаря этому имеется возможность вкладывать одни объекты-коллекции в другие, например, без лишних формальностей можно создать список списков. В некоторых ситуациях тот факт, что кортежи и списки, а также большинство коллекций других типов, хранят ссылки на объекты, а не сами объекты, имеет большое значение. В процедурном программировании мы вызываем функции и часто передаем им данные в виде аргументов. Например, мы уже познакомились с функцией print(). Другая часто используемая функция в языке Python – это функция len(), которая в качестве аргумента принимает единственный элемент данных и возвращает его «длину» в виде значения типа int.

Функция len()

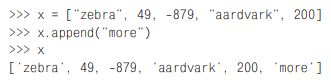
Ниже приводятся несколько примеров вызова функции len():



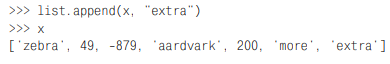
Кортежи, списки и строки имеют «размер», то есть это типы данных, которые обладают категорией размера, и элементы данных таких типов могут передаваться функции len(). (Если функции len() передать элемент, тип которого не предполагает такого понятия, как размер, будет возбуждено исключение.) Все элементы данных в языке Python являются объектами (называемых также экземплярами) определенных типов данных (называемых также классами). Мы будем использовать термины тип данных и класс как взаимозаменяемые. Одно из основных отличий между объектом и простым элементом данных в некоторых других языках программирования (например, встроенные числовые типы в C++ или Java) состоит в том, что объект может обладать методами. В сущности, метод – это обычная функция, которая вызывается в контексте конкретного объекта.

Тип list

Например, тип list имеет метод append(), с помощью которого можно добавить новый объект в список, как показано ниже:



Объект x знает, что принадлежит к типу list (все объекты в языке Python знают, к какому типу они принадлежат), поэтому нам не требуется явно указывать тип данных. Первым аргументом методу append() передается сам объект x – делается это интерпретатором Python автоматически, в порядке реализованной в нем поддержки методов. Метод append() изменяет первоначальный список. Это возможно благодаря тому, что списки относятся к категории изменяемых объектов. Потенциально это более эффективно, чем создание нового списка, включающего первоначальные элементы и дополнительный элемент с последующим связыванием ссылки на объект с новым списком, особенно для очень длинных списков. В процедурном языке то же самое могло бы быть достигнуто с использованием метода append(), как показано ниже (что является совершен- но допустимым вариантом в языке Python):



Здесь указываются тип данных и метод этого типа данных, а в качестве первого аргумента передается элемент данных того типа, метод которого вызывается, за которым следуют дополнительные параметры.

Если вы еще не знакомы с объектно-ориентированным программированием, на первый взгляд такая форма вызова функций может показаться немного странной. Пока вам достаточно будет знать, что обычные функции в языке Python вызываются так: functionName(argiments), а методы вызываются так: objectName.methodName(arguments). Оператор точки («оператор доступа к атрибуту») используется для доступа к атрибутам объектов. До сих пор мы видели только одну разновидность атрибутов – методы, но атрибут может быть объектом любого типа. Поскольку атрибут может быть объектом, имеющим свои атрибуты, которые также могут быть объектами, обладающими атрибутами и т. д., мы можем использовать столько операторов точки, сколько потребуется для доступа к необходимому атрибуту. Тип list имеет множество других методов, включая insert(), который используется для вставки элемента в позицию с определенным индексом, и remove(), который удаляет элемент с указанным индексом. Как уже упоминалось ранее, счет индексов в языке Python начинается с 0. Ранее мы уже видели, что существует возможность извлечь из строки символ, используя оператор квадратных скобок, и отметили, что этот оператор может использоваться с любыми последовательностями.

Списки – это последовательности, поэтому мы можем выполнять следующие операции:



Кортежи также являются последовательностями, поэтому, если бы ссылка x указывала на кортеж, мы могли бы извлекать элементы с помощью квадратных скобок точно так же, как это было сделано со ссылкой x, представляющей список. Но так как списки являются изменяемыми объектами (в отличие от строк и кортежей, которые являются неизменяемыми объектами), мы можем использовать оператор квадратных скобок еще и для изменения элементов списка.

Например:

