Назначение лекции

Вы узнаете, как использовать некоторые простейшие структуры, такие как списки в Python.

Пока что мы использовали переменные, в которых могло храниться только одно значение. Однако возможно, что одно значение может содержать несколько элементов данных.

Есть несколько вариантов сохранения наборов данных в одном значении.

Три простейших варианта:

* списки;
* кортежи;
* словари.

# Списки

Содержимое **списка** может изменяться во время выполнения программы, поэтому списки стали одним из самых распространенных способов хранения наборов данных под одним именем в Python. Данные в списке не обязаны относиться к одному типу. Например, в одном списке могут храниться как строки, так и целые числа; однако позднее это может создать путаницу при обработке списка, поэтому поступать так не рекомендуется.

**Примечание:** в других языках программирования для обозначения переменной, содержащей набор данных, часто используется термин **«массив»**. В Python также существует тип данных под именем «массив», но он используется только для хранения чисел. Числовые массивы Python рассматриваются далее.

# Кортежи

После того как **кортеж** будет определен, вы уже не сможете изменить его содержимое. Это означает, что при написании программы необходимо указать, какие данные хранятся в кортеже, и они останутся неизменными во время выполнения программы. Кортежи обычно используются для команд меню, которые не будут изменяться во время выполнения.

# Словари

Содержимое **словаря** тоже может изменяться во время выполнения программы. Каждому значению присваивается индекс или ключ, по которому можно идентифицировать каждый элемент данных. Этот индекс не изменяется при добавлении или удалении других строк данных — в отличие от списков, в которых позиция элемента может измениться (что приведет к изменению его индекса).

# Примеры кода

**names\_list = ["John", "Tim", "Sam"]**

Создает список имен и сохраняет его в переменной **names\_list**. Квадратные скобки определяют эту группу данных как список; следовательно, ее содержимое может быть изменено во время выполнения программы.

**del names\_list[1]**

Удаляет элемент 1 из списка **names\_list**. Не забудьте, что нумерация начинается с 0, а не с 1. В данном случае из списка будет удалено значение **"Tim"**.

**names\_list.append(input("Add a name: "))**

Предлагает пользователю ввести имя и добавляет его в конец списка **names\_list**.

**print(sorted(names\_list))**

Выводит содержимое **names\_list** в алфавитном порядке, но не изменяет исходного списка, который по-прежнему хранится в исходном порядке. Не работает, если в списке хранятся данные разных типов (например, если один список содержит как строки, так и числовые данные).

**names\_list.sort()**

Сортирует список **name\_list** по алфавиту и сохраняет его в новом порядке. Не работает, если в списке хранятся данные разных типов (например, если один список содержит как строки, так и числовые данные).

**x = [154, 634, 892, 345, 341, 43]**

В этом примере создается список, содержащий числа. Обратите внимание: так как он содержит только числовые данные, кавычки не нужны.

**x = [154, 634, 892, 345, 341, 43]**

**print(len(x))**

Выводит длину списка (то есть количество элементов в списке).

**x = [154, 634, 892, 345, 341, 43]**

**print(x[1:4])**

Выводит данные в позициях 1, 2 и 3 (в данном случае 634, 892 и 345). Не забудьте, что в Python нумерация начинается с 0.

**x = [154, 634, 892, 345, 341, 43]**

**for i in x:**

**print(i)**

Использует элементы списка в цикле **for**; может быть полезно, если вы хотите вывести элементы списка в отдельных строках.

**x = [154, 634, 892, 345, 341, 43]**

**num = int(input("Enter number: "))**

**if num in x:**

**print(num, " Есть в списке")**

**else:**

**print("Нет в списке")**

Предлагает пользователю ввести число, проверяет, присутствует ли число в списке, и выводит соответствующее сообщение.

**x.insert(2, 420)**

Вставляет число 420 в позицию 2 и сдвигает все последующие элементы, чтобы освободить место. Вставка приводит к изменению индексов элементов списка.

**x.remove(892)**

Удаляет элемент из списка. Это может быть полезно, если индекс элемента неизвестен. Если значение входит в список в нескольких экземплярах, удаляется только первый экземпляр.

**x.append(993)**

Добавляет число 993 в конец списка.

**friends = []**

**friend = input("Введите имя друга: ")**

**friends.append(friend)**

В пустой список добавляется в конец введенное имя друга.

**friends = []**

**friend = input("Введите имя друга: ")**

**friends.append(friend)**

**friend = input("Введите имя друга: ")**

**friends.append(friend)**

**print(\*friends, sep="\n")**

В пустой список добавляется в конец введенное имя друга и выдается на экран этот список. sep – разделитель. В данном случае \n – переход на новую строку и следовательно имена будут выданы в две строки.

**relatives = ["Мать","Отец","Брат","Сестра","Дедушка","Бабушка"]**

**relationship = input("Введите родственную связь для поиска: ")**

**if relationship in relatives:**

**index = relatives.index(relationship)**

**print("Эта родственная связь существует.")**

**else:**

**print("Этой родственной связи нет.")**

Получение индекса элемента из списка: **index = relatives.index(relationship)**

**cars = ["Мерседес", "БМВ", "Ауди", "Тойота", "Форд", "Ауди"]**

**car = input("Введите марку автомобиля: ")**

**count = cars.count(car)**

**if count > 0:**

**print(f"Количество совпадений: {count}")**

**else:**

**print("Нет совпадения")**

Определение количества совпадений в списке: **count = cars.count(car)**

Конец лекции