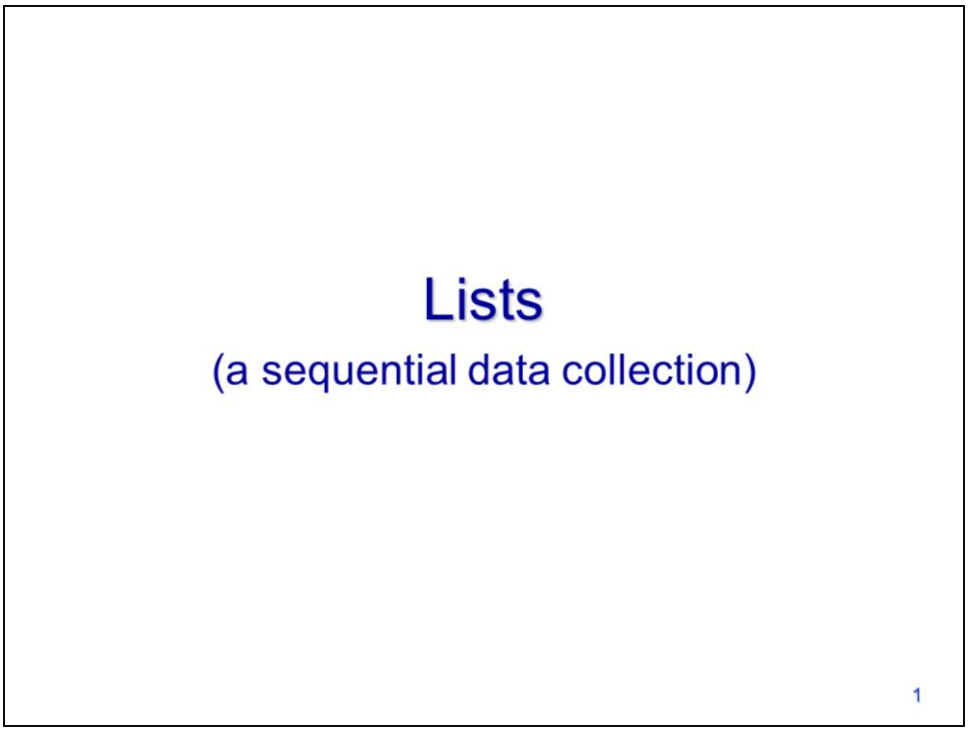
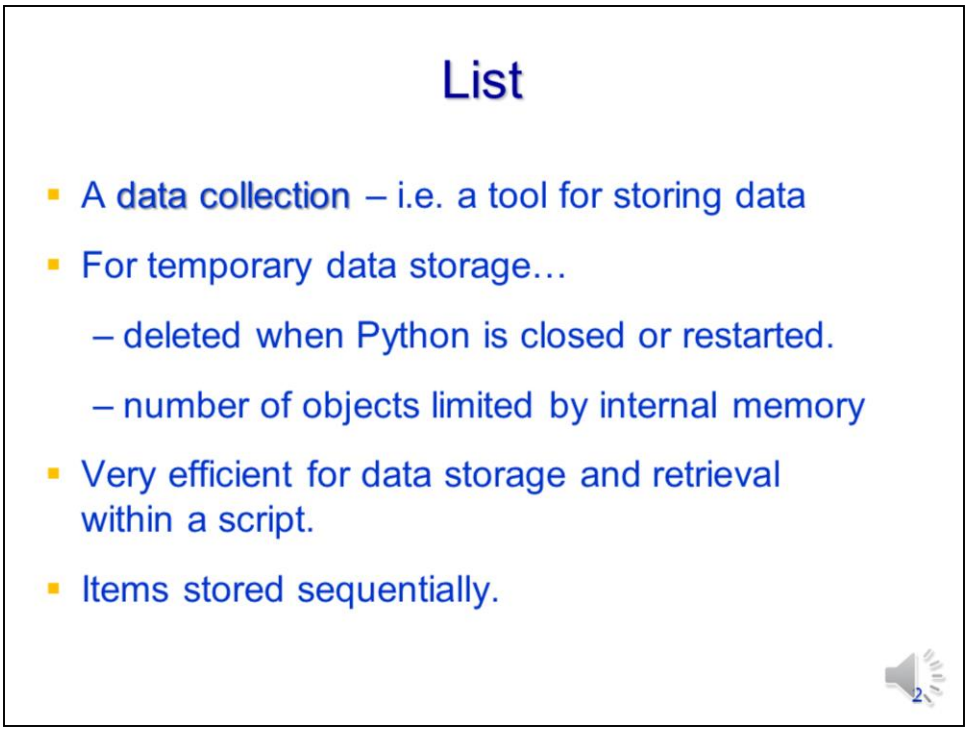


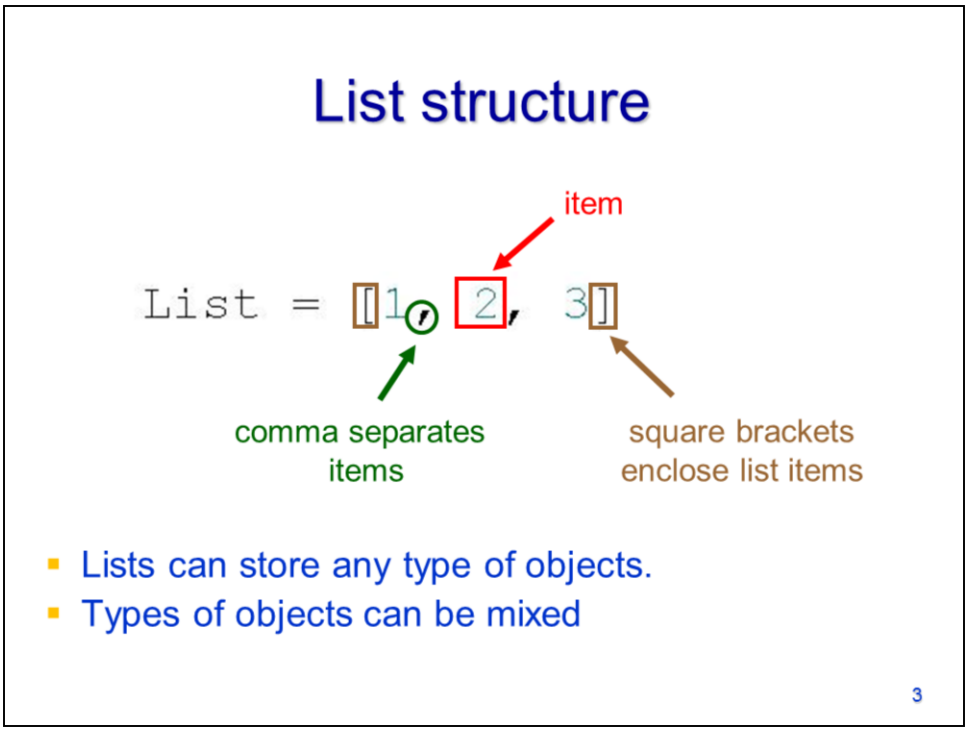
This video will show how to use Python lists.



A list is a frequently used data collections in Python.

It is used for temporary storage and retrieval of data within a script. Items in a list are stored sequentially.

Lists allow for efficient retrieval of ordered or sequential data.



Here we see an example of a Python list.

It is enclosed by square brackets…

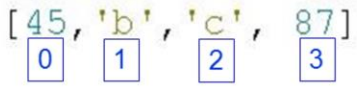
and contains items…

separated by commas. The items, in this case, are numbers.

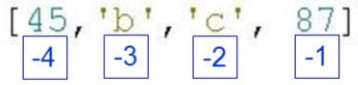
Lists can store any type of objects including numbers, strings, and even lists. A given list can store different types of objects.

List sequences

- Lists are sequences – retrieve item by specifying position in the list.
- Position is counted from either the left of a list…



- Or the right side of a list…

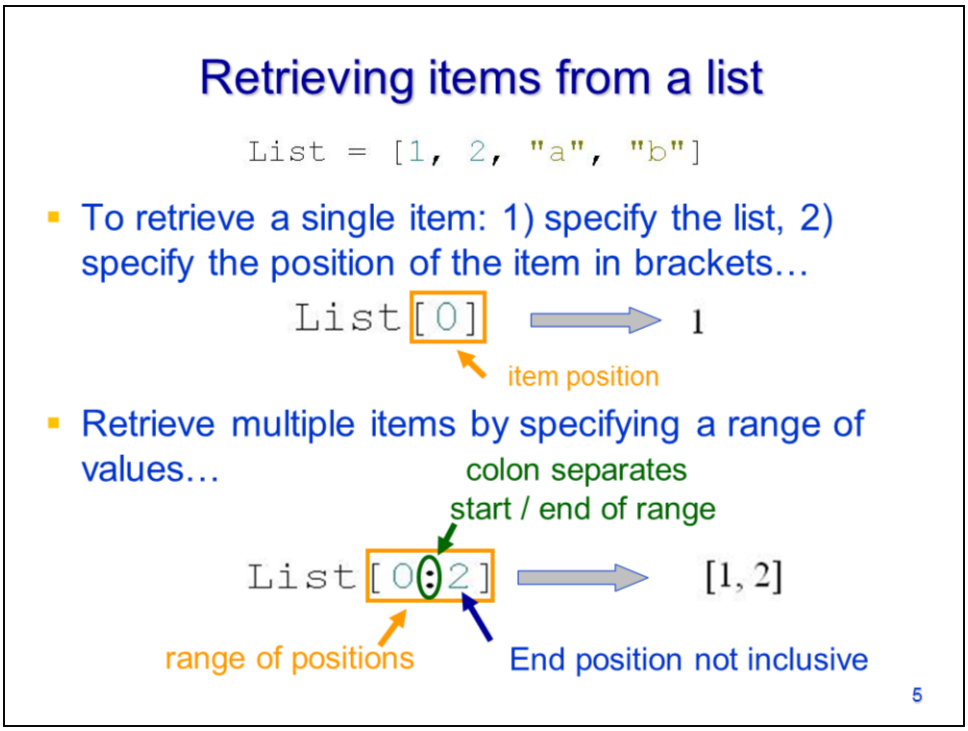


4

Lists are sequential data collections. Items are retrieved by specifying the position of the item in the list – this is analogous to retrieving characters from a string.

The position of an item can either be determined based on the first item in a list…

Or based on the last item in the list. The decision to base the reference the position from the first or last item of the list is made solely on convenience.



Recall that lists can contain different types of objects – in this case the list contains both numbers and strings.

To retrieve a single item from the list, specify the variable assigned to the list followed by the position of the item that you want to retrieve. Note that square brackets enclose the item position.

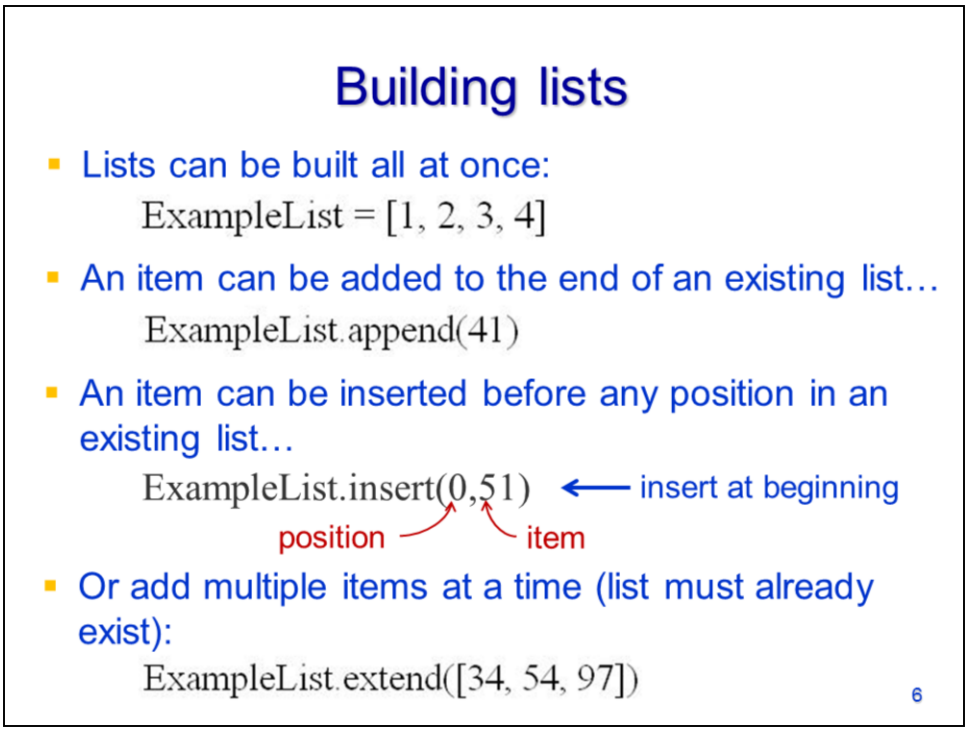
In this case, the item at position zero in the list is the number 1.

You can retrieve multiple items from a list by specifying a range of positions – this is done by indicating the starting and ending position of the range.

The start and end positions are separated by a colon.

Note that the ending position is not inclusive which means that the corresponding item is not retrieved.

The multiple items retrieved from a list are stored in a new list.

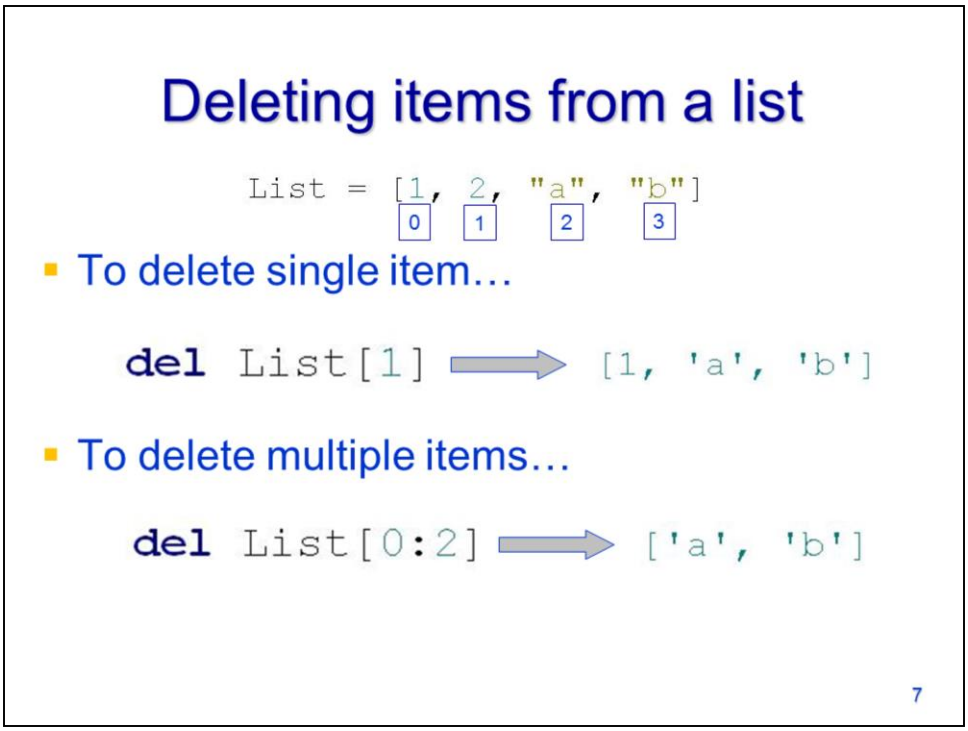


Items can be added to a list when they are first created.

Items can also be added to existing lists. The **append** method adds an item to the end of the list.

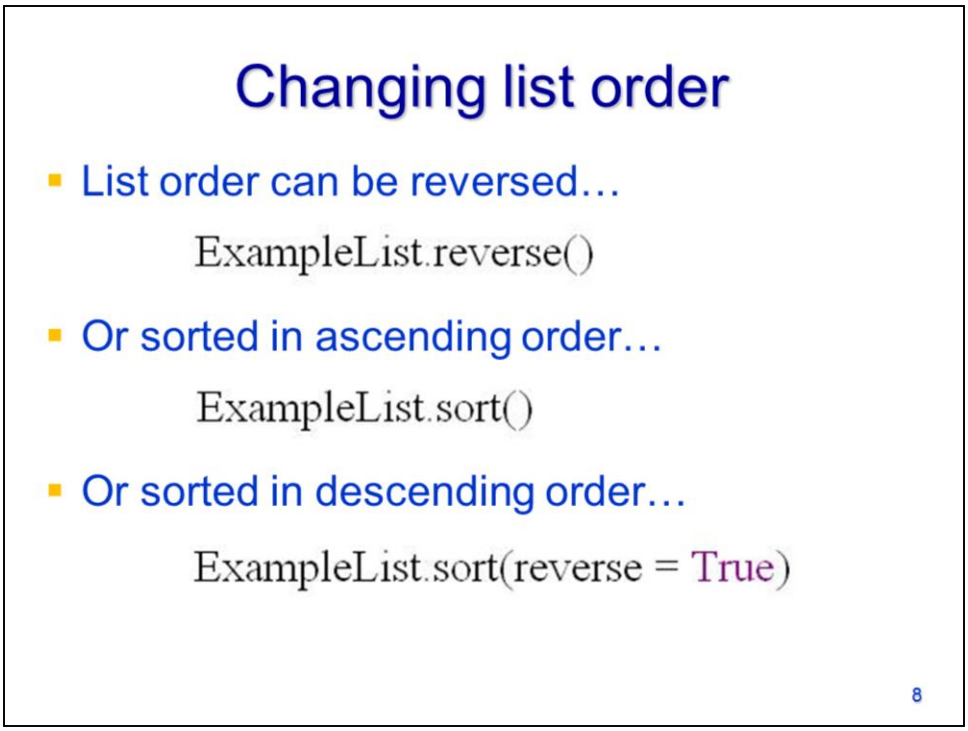
The **insert** method adds an item to any location in the list. To add an item to the beginning of a list, specify position zero.

The **extend** method allows you to append multiple items to the end of a list. The parameter for this method requires a list of items - each of these items will be appended to the existing list.



Items can be removed from a list by specifying the position of the item to remove…

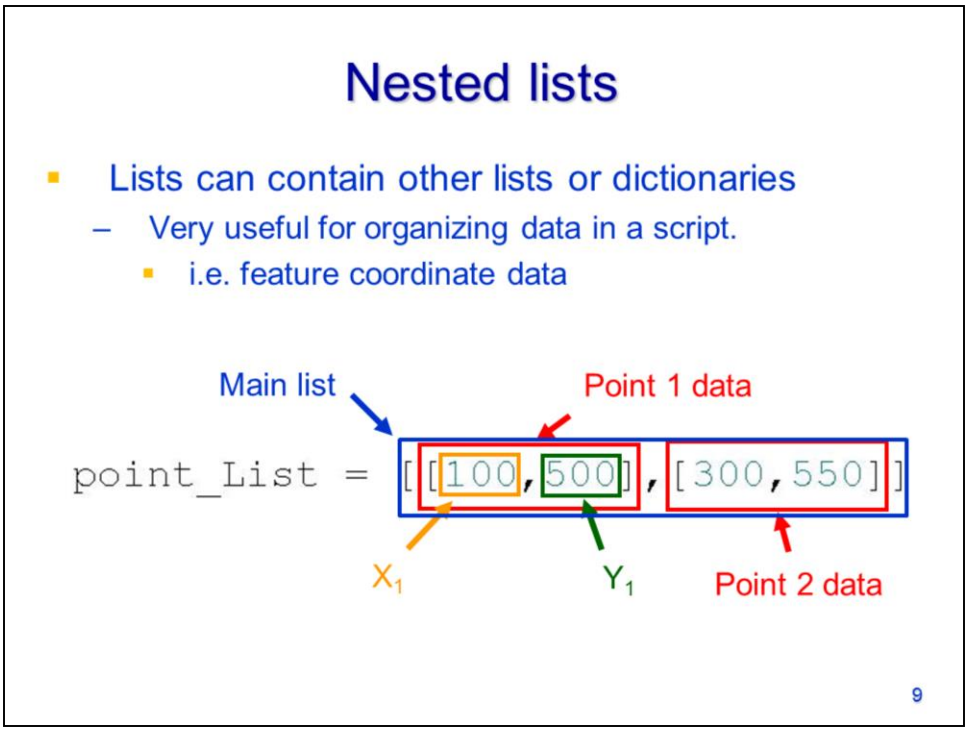
Or by specifying a range of positions to remove. Note that removing items from a list may change the positions of the remaining list items.



The order of a list can be reversed…

Or sorted in ascending order…

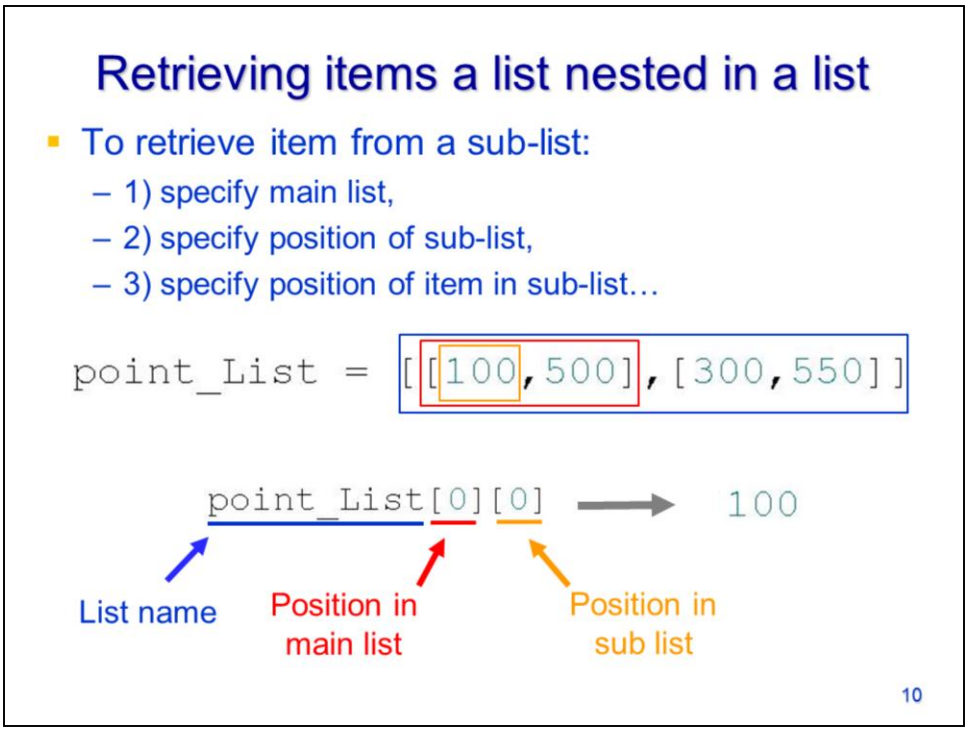
Or sorted in descending order.



Lists can contain other lists – "nesting" lists is often done to organize the data. In this example, we'll use nested lists to organize a set of point coordinates.

The main list contains the full set of points. Within the main list, there is one **sub-list** for each point.

Each sub-list contains the X and Y coordinates for a single point. The nested list structure, in this example, ensures that the X and Y coordinate for each point are paired.



This example will show how to retrieve items from a nested list.

Specify the name of the list…

Followed by the position of the item in the main list…

Followed by the position of the item in the sub-list.

In this example, the value retrieved would be the integer 100.