References to objects are by index

In AppleScript, all object references are by index and get reevaluated every time you delete an item.

Iterate backwards using a repeat loop when deleting items

As a result, if you accidentally skip items, you can receive script errors when you get to final items whose indices are greater than any existing item. The solution is to iterate backwards through items using a `repeat` loop that moves backwards "by -1."

The examples below delete all rows from a table that contain a particular text string in the first column.

VBA

```
Sub DeleteRows()

Dim TargetText As String
Dim oRow As Row

If Selection.Information(wdWithInTable) = False Then Exit Sub

TargetText = InputBox$("Enter target text:", "Delete Rows")

For Each oRow In Selection.Tables(1).Rows
          If oRow.Cells(1).Range.Text = TargetText & vbCr
                  & Chr(7) Then oRow.Delete
Next oRow

End Sub
```

AppleScript

```
set cellEndChar to ASCII character 7
tell application "Microsoft Word"
          if (get selection information selection information type
                   (with in table)) = "false" then return
         display dialog
                   "Delete rows with this text in cell 1:"
                   default answer "Enter target text" with icon 1
          set targetText to text returned of result

          set theRows to every row of table 1 of selection
          repeat with i from (count theRows) to 1 by -1
                   set theRow to item i of theRows
                   if content of text object of cell 1 of theRow =
                            (targetText & return & cellEndChar) then
                           delete theRow
                  end if
         end repeat
end tell
```

Avoid using commands for characters in loops

In AppleScript, **ASCII character 7** is the same thing as **Chr(7)** in Visual Basic for Applications (VBA): an invisible character used by Microsoft Word as a table-cell-end character following a carriage return. The carriage return is known as `vbCr` or **Chr(13)** in VBA, and `return` or **ASCII character 13** in AppleScript. **ASCII character** is a command in the Standard Additions dictionary, which contains the built-in Macintosh collection of scripting additions. You can see this character by doing the following:

1. On the **Word** menu, click **Preferences**.
2. In the **Word Preferences** dialog box, under **Authoring and Proofing Tools**, click **View**.
3. In the **View** dialog box, under **Nonprinting characters**, click **Paragraph marks**.

There is overhead that slows scripts when you call scripting additions, so you don't want to call them repeatedly in a `repeat` loop. The solution is to set a variable to the scripting addition at the top of the script, and then call it just once.

Similarly, you typically want to use the `return`, `space`, and `tab` AppleScript constants rather than their equivalent **ASCII character** commands (**13**, **31**, **9**).

Prior to Word 2004, you could not use `tab` in Word`tell` blocks, because it was interpreted as the **tab** class. That class is now called **tab stop**, and `tab` on its own is the usual text character.

Also, AppleScript doesn't confuse the `return` character with the **return** command. You can see the character in the following snippet from the example above: `(targetText & return & cellEndChar)`.

How repeat loops handle deletions of items

There is also a difference between the AppleScript `repeat` loop and the VBA `For Each` loop. In AppleScript, if you use the `repeat with theRow in the Rows` syntax, or, more literally, `repeat with theRow in (every row of table 1 of selection)`, AppleScript keeps track of the count by indexing the list exactly as in a `repeat with i from 1 to (count theRows)` loop. It constantly checks, or refreshes, the list of items on every iteration, but it doesn't check the index.

Therefore, any time you use a `delete` loop to delete items, you must iterate backwards, as in `(count theRows) to 1 by -1`. Otherwise, every time you delete an item, the next iteration skips one.

For example, if you have three items [a, b, c], and you delete `item 1` (a), `item 2` becomes the original `item 3` (c), and `item 1` becomes the original `item 2` (b), as in [b, c]. `Item 1` is then skipped. Every second item gets skipped this way, and eventually you receive an error when the index becomes larger than the last remaining item.

For this reason, you must always iterate backwards when deleting. That way, all items that have not yet been inspected keep their original index to the end.

Finally, note that the `with in table` information type (remember that `with` and `in` are separate words) does not return the booleans *true* and *false*. Rather, it returns the strings `true` and `false`.