Ranges are not dynamic

In AppleScript, you cannot alter a range without reassigning it to a new variable.

Capture and reassign altered ranges to new variables

The following examples delete duplicate paragraphs using a **Text Range** object. Compare the following examples:

Visual Basic for Applications (VBA)

```
Dim AmountMoved As Long
Dim myRange As Range

'start with first paragraph and extend range down to second

Set myRange = ActiveDocument.Paragraphs(1).Range
AmountMoved = myRange.MoveEnd(unit:=wdParagraph, Count:=1)

'loop until there are no more paragraphs to check

Do While AmountMoved > 0

          'if two paragraphs are identical, delete second one
          'and add the one after that to myRange so it can be checked

          If myRange.Paragraphs(1).Range.Text = _
                        myRange.Paragraphs(2).Range.Text Then
                 myRange.Paragraphs(2).Range.Delete
                AmountMoved =
                          myRange.MoveEnd(unit:=wdParagraph, Count:=1)
        Else
                  'if two paragraphs aren't identical, add the one
                  'after that to my range, so it can be checked, and
                  'drop the first one, since it is no longer of
                 'interest.
                         AmountMoved =
                                   myRange.MoveEnd(unit:=wdParagraph, Count:=1)
                          myRange.MoveStart unit:=wdParagraph, Count:=1
        End If

Loop
```

AppleScript

```
tell application "Microsoft Word"
          --start with first paragraph and extend range down to second
          set myRange to text object of paragraph 1
                  of active document
          set rangeEnd to end of content of myRange

          --in AppleScript a new range is made and returned, cannot alter
          --ranges in place, so redefine myRange to the new range
          set myRange to (move end of range myRange
                  by a paragraph item count 1)
          set newRangeEnd to end of content of myRange
          set amountMoved to newRangeEnd - rangeEnd
          set rangeEnd to newRangeEnd

          --loop until there are no more paragraphs to check
          repeat while amountMoved > 0

                   --if two paragraphs are identical, delete second one
                  --and add the one after that to myRange so it
                 --can be checked
                  if content of text object of paragraph 1 of myRange =
                          content of text object of paragraph 2
                                  of myRange then
                           delete text object of paragraph 2 of myRange
                           set myRange to text object of paragraph 1
                                  of myRange
                           set rangeEnd to end of content of myRange
                           set myRange to (move end of range myRange
                                   by a paragraph item count 1
                           set newRangeEnd to end of content of myRange
                           set amountMoved to newRangeEnd - rangeEnd
                          set rangeEnd to newRangeEnd

                 else
                           --if two paragraphs aren't identical, add
```

```
                --the one after that to my range, so it can
                --be checked, and drop  the first one, since
                --it is no longer of interest.
                set myRange to (move end of range myRange
                        by a paragraph item count 1
            try
                        set newRangeEnd to end of content
                                of myRange
                         set amountMoved to newRangeEnd - rangeEnd
                        set rangeEnd to newRangeEnd
                        set myRange to (move start
                                 of range myRange by a paragraph
                                item count 1
                on error -- errors because can't get
                         --newRangeEnd when move end of range
                         --is missing value at end of document
                        set amountMoved to 0
                end try
        end if

    end repeat

end tell
```

These examples differ because in AppleScript, you cannot alter a range and have it continue to be the same range. Ranges are not dynamic. You can change the range, but then you have to reset the variable `myRange` to the newly altered one.

To make that possible, the commands that alter ranges have to return the new range so that you can get hold of it. This is fairly straightforward. A command, such as **set range**, which in its VBA version does not need to return a result, returns the altered range in AppleScript. Simply reassign it to the same variable (for example, `myRange`) that it was assigned to before **set range**, and carry on as you do in VBA.

Working with the set range command

For example, in the following code, `myRange` is redefined to end at the end of the third paragraph of the active document:

```
Set myRange = ActiveDocument.Paragraphs(1).Range
myRange.SetRange Start:=myRange.Start, _
    End:=ActiveDocument.Paragraphs(3).Range.End
```

In VBA, `myRange` is still defined, only now it will end at the end of the third paragraph. **SetRange** does not return a result, and does not need to because `myRange` has been modified "in place."

In AppleScript, you can't set `myRange` to the **text object** (range) `of paragraph 1` and then try to extend it over more paragraphs, at least not with the **set range** command. This causes Microsoft Word to crash (this is not true for the **move end of range** command). The `myRange` variable remains a reference to the first paragraph, and it can't be something else simultaneously.

To solve this problem, create your own **range** that you set to the same start and end points, and modify that range, as shown in the following example:

```
tell application "Microsoft Word"
    set par1Range to text object of paragraph 1 of active document
    set myRange to create range active document start (start of content
        of par1Range) end
(end of content of par1Range)
    set myRange to set range myRange start (start of content of myRange)
        end (end of content of (get text object of paragraph 3
            of active document))
    content of myRange
end tell
```

That works, and there's no crashing. The `myRange` variable is set to the same dimensions as `par1Range`, but in terms of a start and an end point, not in terms of belonging to a particular paragraph. You can now use **set range** to redefine the end point, but in terms of an integer, not `paragraph 3` itself. The **set range** command returns a result that is a text range, but a new one. There is no modifying "in place."

To continue referring to it as `myRange`, you need to redefine the variable `myRange` to that result. Or, you could set a different variable to the result. However, if your VBA macro expects it still to be called `myRange`, your script should, too, to minimize changes.

Commands can't return the number of characters moved

Another option is to avoid using **set range** entirely, as shown in the following example:

```
tell application "Microsoft Word"
    set par1Range to text object of paragraph 1
        of active document
    set par3Range to text object of paragraph 3
        of active document
    set myRange to create range active document start
        (start of content of par1Range) end
        (end of content of par3Range)
    content of myRange
end tell
```

Although commands such as **move end of range** and **move start of range** otherwise work the same as **MoveEnd** and **MoveStart** in VBA, the fact that they have to return the modified range means that they cannot return the number of characters moved, as in VBA. However, that's not very hard to find. You can keep getting the **end of content** of the text object of the range, both before and after the move, and subtract one from the other to get the difference (`amountMoved`) as the same amount moved.

It also means that you have to keep updating the variables for `rangeEnd` and `newRangeEnd` after performing the subtraction, as in `set rangeEnd to newRangeEnd`, or you will "run out of variables." So it just takes a few more lines of code to do the same thing.

In VBA, you can get the number of characters moved by doing the following:

```
Set myRange = ActiveDocument.Paragraphs(1).Range
AmountMoved = myRange.MoveEnd(unit:=wdParagraph, Count:=1)
```

This code sets `myRange` to the **range** of the first paragraph, and then uses **MoveEnd** to move the end, another paragraph on. Again, it does so "in place." There's no need to redefine `myRange` because it's still there as a dynamic reference, and the **MoveEnd** method is able to return the integer that represents the number of characters this move has advanced.

This time, in AppleScript, you do not have to create your own range. You can use **move end of range** on a text range set to the **text object** of the first paragraph, without crashing. However, you still cannot modify the range "in place." You have to set `myRange`, or another variable, to get hold of the new range returned by **move end of range**, as shown in the following example:

```
tell application "Microsoft Word"
        set myRange to text object of paragraph 1 of active document
        set myRange to (move end of range myRange
                by a paragraph item count 1)
end tell
```

That works. But, because the command must return the new range as the result, it cannot return the amount moved. That is why so much of the script is concerned with calculating the amount moved, which isn't difficult (getting and redefining `rangeEnd` after every move) but is a bit of a bother. As you'll see shortly, there is a simpler way to do it given the necessary difference in the AppleScript command from the VBA version.

Use error trapping to close the script

Something else to bear in mind with this example is to keep alert at the end of the document. In VBA when you try to do the final **MoveEnd** on the last paragraph mark, it doesn't error, but simply returns 0. In AppleScript, **move end of range** also doesn't error, but returns `missing value` (for a nonexistent new range that can't be made). That's a sort of null value. However, the next line that attempts to **get end of content** of a non-existent range does error.

To solve this problem, you trap the error in a `try/on error` block and arbitrarily set the `amountMoved` variable to 0. There are other ways to do this, but this method keeps the same structure as the VBA macro. Also, this is an opportunity for you to see that AppleScript has a `repeat/while` loop, too, although it's not used very often.

You can also rewrite the script to keep all of the **move end**, **move start**, and **delete** occurrences, and omit all of the **get end of content** and `amountMoved` calculations. You then depend on the trapped error at the end to close the script in a simple `repeat` loop with no `while`.

The optimal code example for this task

The following example is the improved version, optimized for AppleScript:

```
tell application "Microsoft Word"
        --start with first paragraph and extend range down to second
        set myRange to text object of paragraph 1 of active document
        set myRange to (move end of range myRange by a paragraph item count 1)

        --loop until there are no more paragraphs to check
       repeat
                --if two paragraphs are identical, delete second one
                --and add the one after that to myRange so it can be checked
                if content of text object of paragraph 1 of myRange =
                        content of text object of paragraph 2 of myRange then
                        delete text object of paragraph 2 of myRange
                        set myRange to text object of paragraph 1 of myRange
                        set myRange to (move end of range myRange
                                by a paragraph item count 1)
            else
                        --if two paragraphs aren't identical, add the one
                        --after that to my range, so it can be checked,
                        --and drop the first one, since it is no longer
                       --of interest.
                       set myRange to (move end
                                of range myRange by a paragraph item count 1)
                      try
                                set myRange to (move start of range myRange
                                        by a paragraph item count 1)
                        on error -- last paragraph
                                --(missing value, so can't move start)
                               exit repeat -- finish
                       end try
                end if
        end repeat
end tell
```