

Re: Use "range," not "for"?

Source: <http://coding.derkeiler.com/Archive/Tcl/comp.lang.tcl/2004-02/0645.html>

From: Glenn Jackman (xx087_at_freenet.carleton.ca)

Date: 02/13/04

Date: 13 Feb 2004 01:10:24 GMT

David McClamrock <mcclamrock@locl.net> wrote:

> I've found that a simple "foreach" loop fairly often won't do the jobs I
> want done,

Really? Why?

[...]

> Instead of writing this:

> for {set i 1} {\$i <= 10} {incr i} {puts "\$i. \"for\" is ugly!"}

> you can write this:

> range i 1 to 10 {puts "\$i. \"range\" is beautiful!"}

>

> Or, if you have a list called "lum," instead of writing this:

> for {set i 0} {\$i < [llength \$lum]} {incr i} {puts [lindex \$lum \$i]}

> you can write this:

> range i 0 no [llength \$lum] {puts [lindex \$lum \$i]}

>

> To go backward, skipping every other number, instead of this:

> for {set i 10} {\$i >= 0} {incr i -2} {puts \$i}

> you can write this:

> range i 10 to 0 -2 {puts \$i}

[...]

Woof, it looks needlessly complicated. You don't have to muck about with uplevel at all. Have a look at <http://wiki.tcl.tk/10795>

It demonstrates an "integer range generator" proc called [..]

Using it would change your examples to:

```
1 foreach i [.. 1 10] {puts "$i. '.' is beautiful"}
```

```
2 foreach i [.. 0 [expr {[llength $lum] - 1}]] {puts [lindex $lum $i]}
   # OK, that's not as pretty as your [range], but it's a crummy
   # example anyway
```

```
3 foreach i [.. 10 0 -2] {puts $i}
```

Here's a wiki page showing a "do ... until ..." loop, which demonstrates how to neatly handle executing a script body that's passed as an

comp.lang.tcl: Re: Use "range," not "for"?

argument to a proc:

<http://wiki.tcl.tk/917>

--

Glenn Jackman
NCF Sysadmin
glennj@ncf.ca