Subject: Re: HASH -- bug, or "feature"?
Posted by Gray on Wed, 20 Apr 2011 21:26:14 GMT
View Forum Message <> Reply to Message

On Apr 20, 4:33 pm, Mark Piper <mpi...@ittvis.com> wrote:
> Here's an example of the technique Mike suggests:
>
> IDL> names = list('red', 'green', 'blue')
> IDL> colors = hash()
> IDL> colors[names[0]] = [255,0,0]
> IDL> colors[names[1]] = [0,255,0]
> IDL> colors[names[2]] = [0,0,255]
>
> IDL> print, colors ; order is mangled
> green: 0 255 0
> blue: 0 0 255
> red: 255 0 0
>
> IDL> foreach n, names do print, n, colors[n] ; ordered
> red 255 0 0
> green 0 255 0
> blue 0 0 255
>
> This also works with an array (and a FOR loop) instead of a list.
>
> mp

This all makes perfect sense... except that it isn't really useful for me. I had been using a hash so that I could retrieve and store information (in the form of structures) about particular stars by indexing with star IDs and not having to search over arrays or lists for individual members. When I had information for a set of stars where some but not all were already in the hash, I would do something like this:

```
tmp = replicate({star},n_new)
old = where(star_hash.haskey(new_ids),n_old)
if (n_old gt 0) then tmp[old] =
(star_hash[new_ids[old]].values()).toarray()
tmp.info = new_info & tmp.id = new_ids
star_hash[new_ids] = tmp
```

But that doesn't work, because there's no guarantee that the values I retrieve from the hash are in the same order as the array elements I'm storing them in. What I end up having to do is

```
tmp = replicate({star},n_new)
old = where(star_hash.haskey(new_ids),n_old)
```

foreach i,old do tmp[i] = star_hash[new_ids[i]]

...and the rest is the same. I'm not sure whether this is more
inefficient or not... maybe it isn't, thanks to toarray() being slow.
