Subject: Re: HASH -- bug, or "feature"?
Posted by Gray on Thu, 21 Apr 2011 11:20:27 GMT
View Forum Message <> Reply to Message

On Apr 20, 6:09 pm, Paulo Penteado <pp.pente...@gmail.com> wrote:
> On Apr 20, 6:26 pm, Gray <grayliketheco...@gmail.com> wrote:
>
>> 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
```

>
> I have a subclass for ordered hashes, which I could clean up and make
> public if there is interest. However, I do not see why it is needed
> above. If I understand it right, you want to put the new elements in
> the hash, without overwriting any preexisting elements. Would it not
> be the same as just
>

```
> tmp=replicate({star},n_new)
> tmp.info=new_info
> tmp.id=new_ids
> new=where(~star_hash.haskey(new_ids),/null)
> star_hash[new_ids[new]]=tmp[new]
```

>
> ?
>
> The work being just to avoid overwriting the preexisting elements. If
> they could be overwritten, it would be just
>

```
> tmp=replicate({star},n_new)
> tmp.info=new_info
> tmp.id=new_ids
> star_hash[new_ids]=tmp
```

It's worse than that. tmp.info = new_info was shorthand for updating
the relevant structure tags with the new information; however, there
are other tags with old information that I don't want to overwrite.
So I need to preserve some stuff and overwrite others, which is why I

had to do the complicated jig.