

## Re: Semi-newbie, rolling my own \_\_deepcopy\_\_

---

*Source:* <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2005-04/msg01143.html>

---

- *From:* Michael Spencer <[mahs@xxxxxxxxxxxxxxxxxxxxx](mailto:mahs@xxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Wed, 06 Apr 2005 11:11:41 -0700
- 

ladasky@xxxxxxxxxxxxx wrote:

....  
I see Steve Bethard has answered most of the points in your last eMail

On line 11 we create a dictionary item in memo, [id(self):type(self)]...So now I'm confused as to the purpose of memo. Why should it contain the ID of the \*original\* object?

No, you create memo[id(self):type(self)()] i.e., a mapping of the old object to a new object of the same type. This new object does not yet contain the data of the old one, because you called its `__init__` method with no arguments.

BTW, as I mentioned in a previous comment, I believe this would be more plainly written as `type(self).__new__()`, to emphasize that you are constructing the object without initializing it. (There is a explanation of `__new__`'s behaviour at [http://www.python.org/2.2/desclntro.html#\\_new\\_](http://www.python.org/2.2/desclntro.html#_new_)). This works only for new-style classes (also explained in the same reference). For old-style 'classic' classes you would use `new types.InstanceType`

....

Is there another section of the Python docs that will clarify all this for me? I got hung up on all the "static", "public", etc. declarations in Java early on. Python has taken me an awful lot farther, but perhaps I'm hitting the conceptual wall again.

I assume you have seen <http://docs.python.org/lib/module-copy.html> which explains this topic and, in particular, why the memo dict is passed around.

Hang in there. It's not an easy topic, and this example contains several subtleties as you have discovered.

If you want further help, why not post the actual class you are working

Re: Semi-newbie, rolling my own \_\_deepcopy\_\_

with, and your \_\_deepcopy\_\_ attempt. It would be much easier to react to something concrete than a general howto.

Michael