

Re: Magic function

Source: <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2008-01/msg01823.html>

- *From:* Ruediger <larudwer@xxxxxxxxxx>
 - *Date:* Tue, 15 Jan 2008 22:47:36 +0100
-

dg.google.groups@xxxxxxxxxxxxxxxx wrote:

Hi Rüdiger,

Thanks for your message. I liked your approach and I've been trying something along exactly these sorts of lines, but I have a few problems and queries.

The first problem is that the id of the frame object can be re-used, so for example this code (where I haven't defined InstanceTracker and getInstances, but they are very closely based on the ideas in your message):

```
class A(InstanceTracker):
    gval = 0
    def __init__(self):
        self.value = A.gval # each time you make a new object, give
        A.gval += 1 # it a value one larger
    def __repr__(self):
        return str(self.value)

def f2():
    a = A() # objects 0 and 2
    return getInstances(A)

def f3():
    a = A() # object 1
    return f2()

inst2 = f2()
inst3 = f3()
print inst2
print inst3
```

The output is:

```
[0]
[0, 2]
```

Re: Magic function

The A-variable with value 0 is not being garbage collected because it's saved in the variable inst2, but it's also being returned by the second call to getInstances because the frame of f2 is the same each time (which makes sense, but may be implementation specific?). The

Yes and No. id basically returns the memory address of an object. and yes this is implementation specific. As of my knowledge a stackframe is of constant size in cPython. Though you get always the same id for the same call level as you would always get the same number from your instance tracker.

No A-variable with value 0 is reported the second time because it had been created at the same call level __and__ it is still accessible from that call level.

If you do want such object's to be destroyed you must not create hard references to them. This may be hard for your users.

However you could still do something like:

```
def f2():
    InstanceTracker.prepare() # <-- delete previously created Entrys
    # here or calculate some magic hash value
    # or random number.
    a = A() # objects 0 and 2
    return getInstances(A)
```

or

```
@managedInstance # <-- see above
def f2():
    a = A() # objects 0 and 2
    return getInstance
```