

Using debug print routine inside assert

Source: <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2003-11/0713.html>

From: Edvard Majakari (edvard+news_at_majakari.net)

Date: 11/04/03

Date: Tue, 04 Nov 2003 14:15:30 +0200

Hello, fellow Pythonistas!

Today I shortly discussed the problems in using print statements for debugging problems in Python. Often in coding, in addition to asserts it is nice to have a debug routine like

```
import time
global DEBUG_LVL = 0
...
def debug(msg, threshold=1):
    if threshold > DEBUG_LVL:
        return

    print "%s %s\n" %(time.strftime("%d%m%y %H:%M:%S"), msg)

# somewhere in the program
debug("subfroblicate(%s, %s) returned %s" %(p1, p2, subfroblicate(p1, p2)))
```

The idea here is to have consistent debug print messages. However, parameters or expressions inside the debug statement might be very time-consuming to run. Assume for a moment that calling `subfroblicate()` takes 0.02 seconds, and it is called 10 000 times in your test program. During the debug stage, over three minutes of waiting is not necessarily bad, but if you have lots of that kind of debug statements, it really pays to take them away from the final product. But this could be tedious if you have lots of code.

The problem is, if `DEBUG_LVL` is less than `threshold` (like in the example above), parameters to method call are evaluated but results are thrown away, rendering parameter evaluation (and potential calculations) useless. You spent extra time in evaluating debug parameter just to throw results away. If you could skip the evaluation of `debug()`, given low enough `DEBUG_LVL`, you would be fine.

One solution – though not elegant and rather ugly – is to use Python's ability to remove assert statements from byte-compiled programs, and modify debug routine so that it prints given argument and returns true:

comp.lang.python: Using debug print routine inside assert

```
def debug(msg, threshold=1):
    if threshold > DEBUG_LVL:
        return

    print "%s %s\n" %(time.strftime("%d%m%y %H:%M:%S"), msg)
    return True # return true so that assert always succeeds
```

use in the program:

```
assert(debug("subfroblicate(%s, %s) returned %s" \
            %(p1, p2, subfroblicate(p1, p2))))
```

In C, you could use macros to implement assert-wrapped debug function, but due to their hard-to-detect nature I wouldn't resort to macros, and nevertheless, in Python, you don't have such thing as macros(?).

What do you think, what is the best solution for this kind of problem? I for one wouldn't like to see myself writing `assert(debug(...))` every now and then :)

```
--
# Edvard Majakari           Software Engineer
# PGP PUBLIC KEY available  Soli Deo Gloria!
$_ = '456476617264204d616a616b6172692c20612043687269737469616e20'; print
join(' ',map{chr hex}{split/(\w{2})/}),uc substr(crypt(60281449,'es'),2,4),"\n";
```