

Let's make Python debugging on the console not suck!

```
> /home/andreas/work/pudb/debug_me.py(37)<module>()
-> print i
(Pdb) n
(4, 3, 5)
> /home/andreas/work/pudb/debug_me.py(36)<module>()
-> for i in fermtat(2):
(Pdb) n
> /home/andreas/work/pudb/debug_me.py(36)<module>()
-> print i
(Pdb) n
(8, 6, 1)
> /home/andreas/work/pudb/debug_me.py(36)<module>()
-> for i in fermtat(2):
(Pdb) n
> /home/andreas/work/pudb/debug_me.py(37)<module>()
-> print i
(Pdb) n
(12, 4, 1)
> /home/andreas/work/pudb/debug_me.py(36)<module>()
-> for i in fermtat(2):
(Pdb) n
31         if x**n + y**n ==
32             yield x, y.
33
34     print "done func(%d)" % n
35
36     -> for i in fermtat(2):
37         print i
38
39     print "FINISHED"
[EOF]
(Pdb) █
```

Want:

- ▶ Situational Awareness
 - ▶ Source Code
 - ▶ Variables
 - ▶ Stack
- ▶ Easy Drop to Python Shell
- ▶ Single Keystrokes
- ▶ Visual Breakpoints

DEBUG FAIL

PuDB!

easy_install pudb

PuDB 0.92.15 - The Python Urwid debugger - Hit ? for help - (C) 2008

```
z = None
w = ()

y = dict((i, i**2) for i in s)

k = set(range(5, 99))

try:
    x.invalid
except AttributeError:
    pass

#import sys
#sys.exit(1)

> return 2*x

def fermat(n):
    """Returns triplets of the form x^n + y^n = z^n.
    Warning! Untested with n > 2.
    """
    from itertools import count
    for x in range(100):
        for y in range(1, x+1):
            * for z in range(1, x**n+y**n + 1):
                #from pudb import set_trace; set_trace()
                if x**n + y**n == z**n:
                    yield x, y, z

print "SF", simple_func(10)
```

Variables:

- k: set
- s: list
- w: tuple
- x: 11
- y: dict
- z: NoneType

Stack:

- <module> debug_me.py:34
- >> simple_func debug_me.py:20

Breakpoints:

- debug_me.py:29

PuDB: Debug like it's 1992!

easy_install pudb

- ▶ Perfect for debugging over SSH
- ▶ `from pudb import set_trace`
`set_trace()`
- ▶ `python -m pudb.run failscript.py`
- ▶ Step over/into `n` / `s`
- ▶ Drop to Python/IPython Shell `!`
- ▶ Module Browser `m`
- ▶ Expand variable (on variable) `<`
- ▶ View stack frame (on frame) `Enter`



Thanks for your attention!