

Re: converting a sed / grep / awk / . . . bash pipe line into python

Source: <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2008-09/msg00160.html>

- *From:* Roy Smith <roy@xxxxxxxx>
 - *Date:* Wed, 03 Sep 2008 16:54:12 -0400
-

In article <g9m6at\$a71\$01\$1@xxxxxxxxxxxxxxxxxxxx>, Peter Otten <__peter__@xxxxxx> wrote:

Roy Smith wrote:

In article <g9lvc5\$8qq\$03\$1@xxxxxxxxxxxxxxxxxxxx>, Peter Otten <__peter__@xxxxxx> wrote:

I might take it one step further, however, and do:

```
fields = line.split()[:2]
a, b = map(int, fields)
```

in fact, I might even get rid of the very generic, but conceptually overkill, use of map() and just write:

```
a, b = line.split()[:2]
a = int(a)
b = int(b)
```

If you go that route your next step is to introduce another try...except, one for the unpacking and another for the integer conversion...

Why another try/except? The potential unpack and conversion errors exist in both versions, and the existing try block catches them all. Splitting the one line up into three with some intermediate variables doesn't change

Re: converting a sed / grep / awk / . . . bash pipe line into python

that.

As I understood it you didn't just split a line of code into three, but wanted two processing steps. These logical steps are then somewhat remixed by the shared error handling. You lose the information which step failed. In the general case you may even mask a bug.

Peter

Well, what I really wanted was two conceptual steps, to make it easier for a reader of the code to follow what it's doing. My standard for code being adequately comprehensible is not that the reader *can* figure it out, but that the reader doesn't have to exert any effort to figure it out. Or even be aware that there's any figuring-out going on. He or she just reads it.

.