

Re: Regular expressions: recursive patterns and callouts

Source: <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2006-01/msg03581.html>

- *From:* "Paul McGuire" <ptmcg@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Sun, 22 Jan 2006 03:56:39 GMT
-

"Carlos" <carlosjosepita@xxxxxxxx> wrote in message
news:1137876890.715521.292280@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

> Hi!
> I think two extensions to re could be really powerful in aiding to
> match non regular strings –for example those containing parens nested
> up to an arbitrary depth– but without a significative performance loss,
> although I'm not at all sure at this last point. One of these features
> is re nesting; I don't mean nesting some strings and then compiling the
> composed re, but to include –possibly recursive– references to other
> re's. The second feature is the ability to add callout points in a re,
> so external functions would be called when the matcher reaches those
> points to validate the current matched prefix or even to do some
> arbitrary further matching of its own, advancing the pattern position.
> pcre implements similar features with its (?CN) callout and (?N)
> recursive patterns mechanisms.
> Are the above or equivalent features planned for future python
> releases?
> Do you know of some extension package which provide them or at least
> pcre compatible res?
> Thank you in advance,
> Carlos
>

Pyparsing supports both parse–time callouts (called "parse actions") and recursive grammar definitions.

Download pyparsing at <http://pyparsing.sourceforge.net>.

— Paul

-
- *References:*
 - ◆ [*Regular expressions: recursive patterns and callouts*](#)

Re: Regular expressions: recursive patterns and callouts

◇ *From:* Carlos

- Prev by Date: ***Re: how to run python scripts on a website***
- Next by Date: ***new.instancemethod as a form of partial()***
- Previous by thread: ***Regular expressions: recursive patterns and callouts***
- Next by thread: ***Compiling cx Oracle and LD LIBRARY PATH***
- Index(es):
 - ◆ ***Date***
 - ◆ ***Thread***