Grammar for simple S-expressions:

```
<P>  ::= <E>
<E>  ::= atom | ' <E> | ( <E> <Es> )
<Es> ::= <E> <Es> | ε

FIRST sets:
<P>  → {atom, ', ( }
<E>  → {atom, ', ( }
<Es> → {atom, ', (, ε}

FOLLOW sets:
<P>  → {ε}
<E>  → {ε, atom, ', (, ) }
<Es> → { ) }
atom → {ε, atom, ', (, ) }
'    → {atom, ', ( }
(    → {atom, ', ( }
)    → {ε, atom, ', (, ) }

PREDICT sets:
<P>  ::= <E>             → {atom, ', ( }
<E>  ::= atom            → {atom}
<E>  ::= ' <E>           → { ' }
<E>  ::= ( <E> <Es> )    → { ( }
<Es> ::= <E> <Es>        → {atom, ', ( }
<Es> ::= ε               → { ) }
```

```
"""Correctness parser for S-expression grammar. Note that this program
commits a massive style faux pas: the variables token and tokens are
global variables. In other words, they are accessible from anywhere in
the program. I chose to do this because this any attempt to do this
'right' results in a bunch of parameters getting passed around that
obscures the idea that this demonstration program is trying to
convey. Sometimes 'proper' isn't always 'simplest.'"""

import sys

def tokenize(lexemes):
    global tokens
    # Make a copy of lexemes (don't just point tokens at it)
    tokens = list(lexemes)
    for i in range(len(tokens)):
        if tokens[i] not in ["(", ")", "'", None]:
            tokens[i] = 'atom'

def nextToken():
    """Grab the next token. Set the current token to None if there are
    no more tokens."""
    global token
    #print 'Matched',token
    if len(tokens)>0:
        token = tokens.pop(0)
    else:
        token = None

```

```
29 def match(expected):
30     """Match the current token against the expected value. If
31     successful, grab the next token. Set the current token to None if
32     there are no more tokens."""
33     if token==expected:
34         nextToken()
35     else:
36         raise Exception('Parse error',token,expected)
37 
38 def P():
39     if token in ['atom' , "'" , '(']:
40         E()
41     else:
42         raise Exception('Parse error')
43 
44 def E():
45     if token == 'atom':
46         match('atom')
47     elif token == "'":
48         match("'")
49         E()
50     elif token == "(":
51         match("(")
52         E()
53         Es()
54         match(")")
55     else:
56         raise Exception('Parse error')
57 
58 def Es():
59     if token in ['atom' , "'" , '(']:
60         E()
61         Es()
62     elif token == ")":
63         pass
64 
65 """Open up the file, grab the program, and parse it."""
66 
67  file  = open(sys.argv[1],'r')
68 data = file.read()
69  file.close()
70 lexemes = data.split()
71 tokenize(lexemes)
72 token = tokens.pop(0)
73 P()
74 print "Success"
```