Lexical rules

1. Comments in Python begin with "#" and continue to the end of the line.
2. Lines containing only blanks, tabs, and comments are ignored entirely (they have no effect on indentation level). We call these blank lines hereafter.
3. Blanks and tabs may separate lexical tokens. They are required only to separate two tokens only when they could be interpreted differently if concatenated together (e.g. "for" followed by "x" could be interpreted as "forx" without intervening space). Blanks and tabs may not appear within individual lexical tokens (described in this section) other than string literals.

Lexical tokens

```
NEWLINE: the new-line character
ENDMARKER: the end of file token
NUMBER: is defined by regular expression [0-9]+
ID: is defined by regular expression [_a-zA-Z][_a-zA-Z0-9]*
STRING: a squence of characters enclosed by either single or double quotes
```

Grammar notations

1. `{ rule }*` means any number of applications of *rule*, including no application (zero application) of *rule.*
2. `{ rule }+` means one or more applications of *rule*.
3. `[ rule ]` means *rule* is optional.
4. `{( or-separated-entities ) rule }` means if one of the *or-separated-entities* is present, then *rule* applies.
5. Strings with single quotes around them are terminals.

Grammar rules

1. `file_input: {NEWLINE | stmt}* ENDMARKER`

2. stmt: simple_stmt | compound_stmt
3. simple_stmt: (print_stmt | assign_stmt) NEWLINE
4. print_stmt: 'print' '(' [testlist] ')'
5. assign_stmt: ID '=' test
6. compound_stmt: for_stmt

7. for_stmt: 'for' '(' assign_stmt ';' test ';' assign_stmt ')' '{' NEWLINE stmt+ '}' NEWLINE

8. testlist: test {',' test}*

9. test: comparision

10. comparison: arith_expr {comp_op arith_expr}*

11. comp_op: '<'|'>'|'=='|'>='|'<='|'<>'|'!='

12. arith_expr: term {('+'|'-') term}*
13. term: factor {('*'|'/'|'%'|'//') factor}*
14. factor: {'-'} factor | atom
15. atom: ID | NUMBER | STRING+ | '(' test ')'