from __past__ import print_statement

updating the grammar of skulpt to python3 while maintaining backwards compatibility with python2

@ajpnijburg — @albertjan

WHAT IS SKULPT

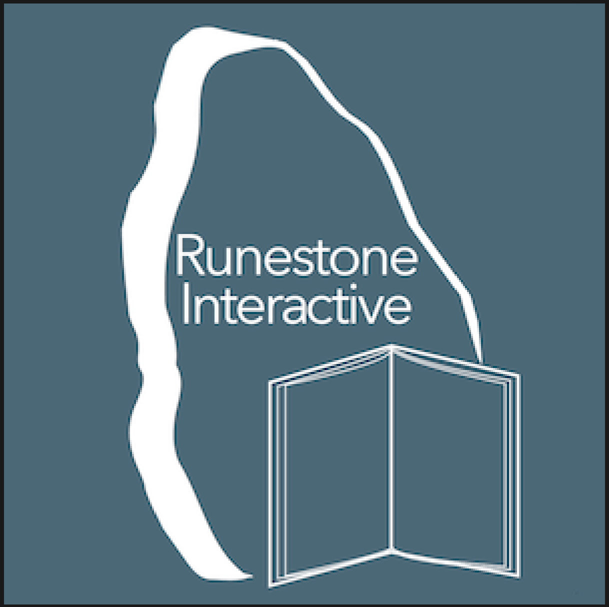
A

- python interpreter written in javascript
- port of the python compiler in javascript

USED BY

anvil

trinket


Runestone
Interactive

RICE | ONLINE LEARNING

UNIVERSITY OF
MICHIGAN

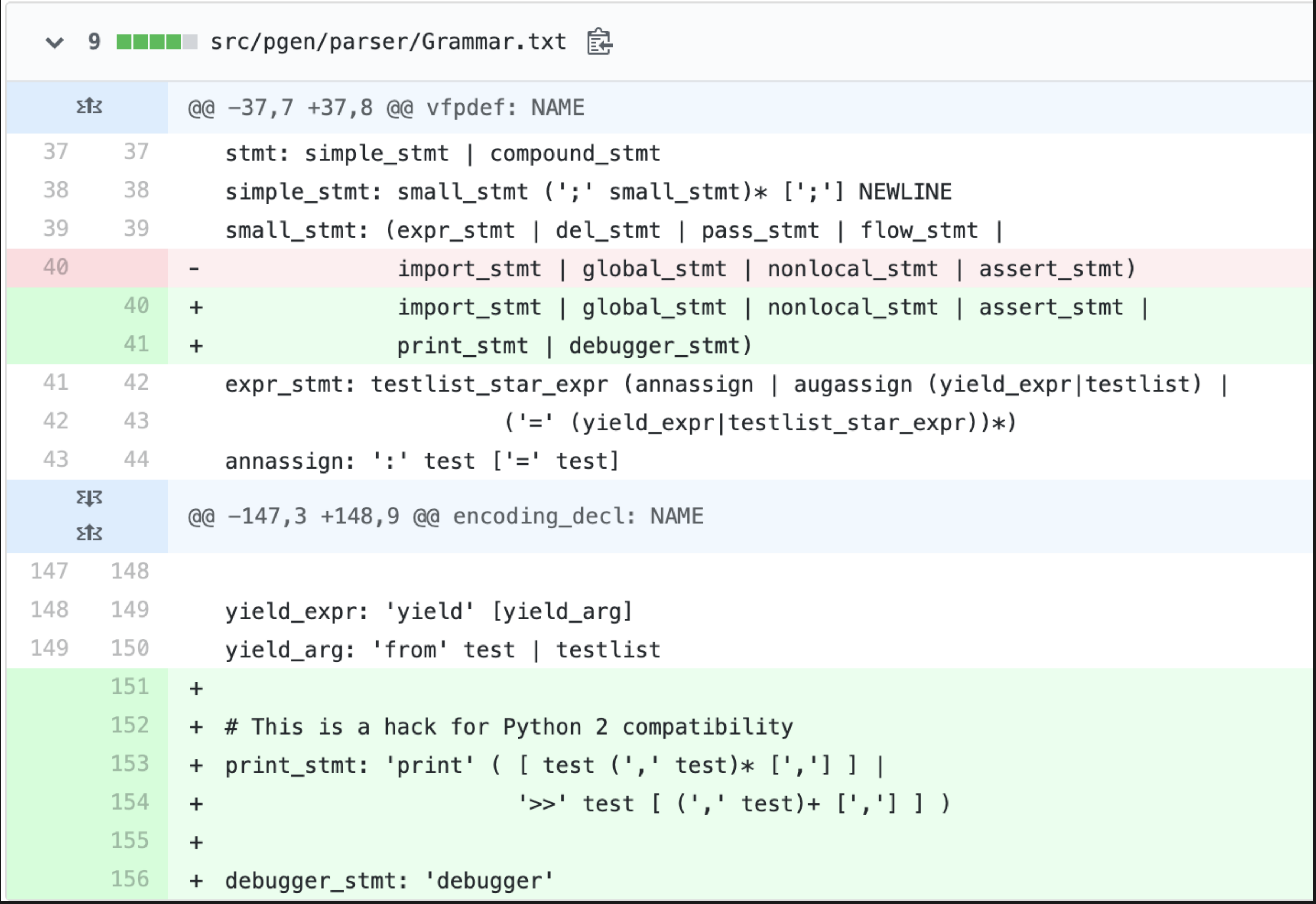
UPDATE TO PYTHON3

Ugly truths:

1. python 2 is eol (almost)
2. skulpt's users have large user code bases on python2
3. you can't branch and maintain two versions of skulpt with such a small team

HOW TO KEEP PYTHON 2 (A BIT)

PRINT STATEMENT

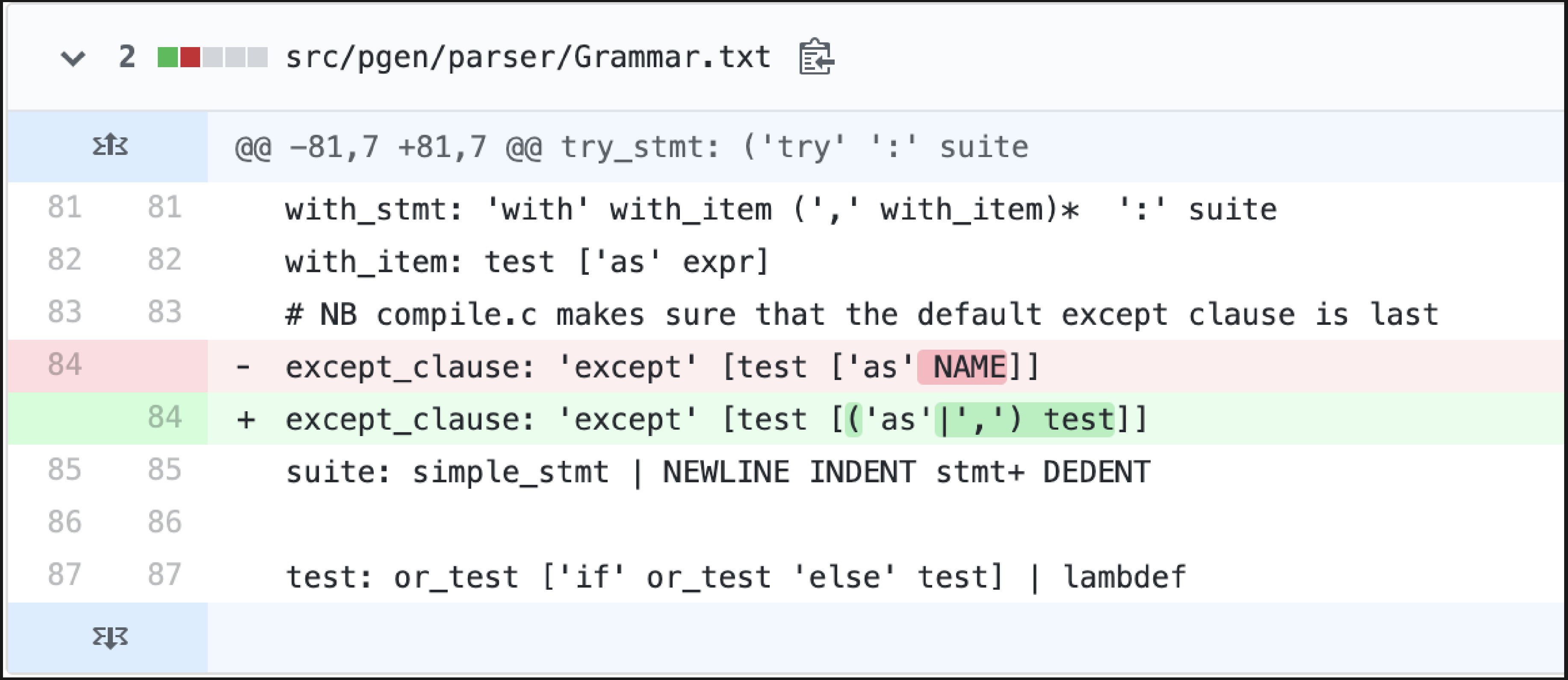


```diff
9 src/pgen/parser/Grammar.txt

@@ -37,7 +37,8 @@ vfpdef: NAME
  stmt: simple_stmt | compound_stmt
  simple_stmt: small_stmt (';' small_stmt)* [';'] NEWLINE
  small_stmt: (expr_stmt | del_stmt | pass_stmt | flow_stmt |
-              import_stmt | global_stmt | nonlocal_stmt | assert_stmt)
+              import_stmt | global_stmt | nonlocal_stmt | assert_stmt |
+              print_stmt | debugger_stmt)
  expr_stmt: testlist_star_expr (annassign | augassign (yield_expr|testlist) |
                       ('=' (yield_expr|testlist_star_expr))*)
  annassign: ':' test ['=' test]
@@ -147,3 +148,9 @@ encoding_decl: NAME

  yield_expr: 'yield' [yield_arg]
  yield_arg: 'from' test | testlist
+
+ # This is a hack for Python 2 compatibility
+ print_stmt: 'print' ( [ test (',' test)* [','] ] |
+                       '>>' test [ (',' test)+ [','] ] )
+
+ debugger_stmt: 'debugger'
```

EXCEPT CHANGES

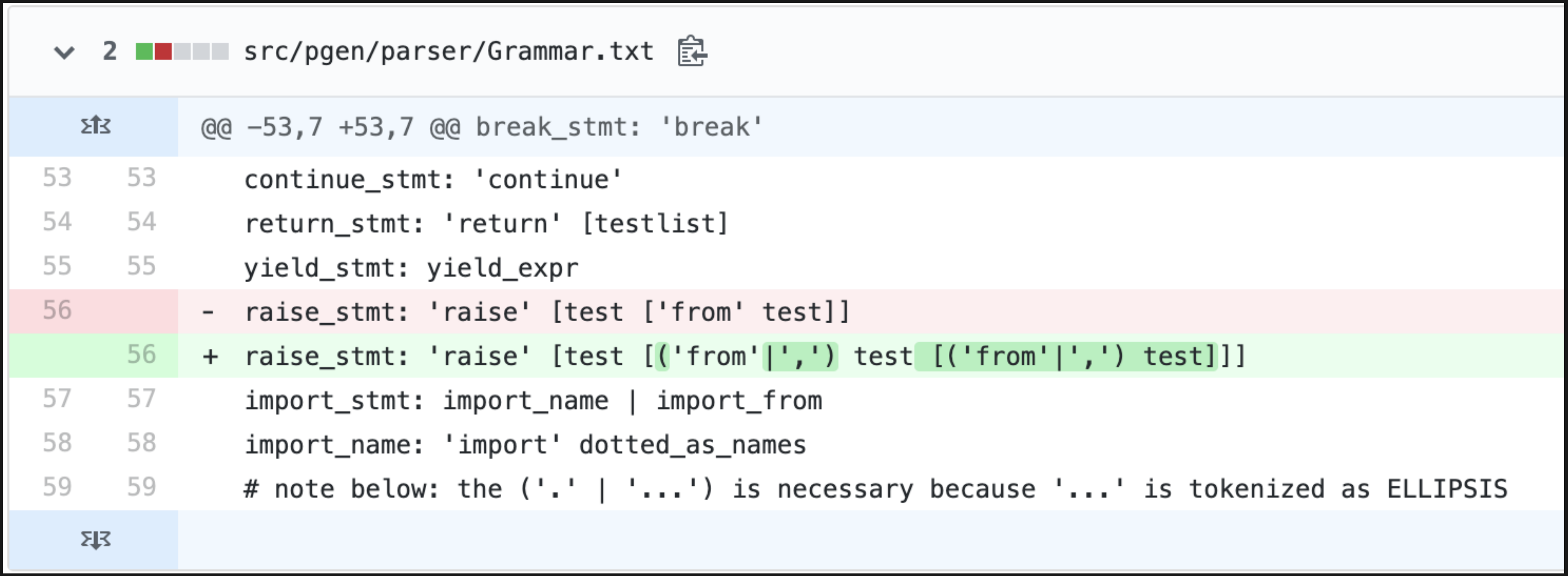


2 🟩🟥⬜⬜⬜ src/pgen/parser/Grammar.txt

```diff
@@ -81,7 +81,7 @@ try_stmt: ('try' ':' suite
 with_stmt: 'with' with_item (',' with_item)*  ':' suite
 with_item: test ['as' expr]
 # NB compile.c makes sure that the default except clause is last
- except_clause: 'except' [test ['as' NAME]]
+ except_clause: 'except' [test [('as'|',') test]]
 suite: simple_stmt | NEWLINE INDENT stmt+ DEDENT

 test: or_test ['if' or_test 'else' test] | lambdef
```

RAISE CHANGES



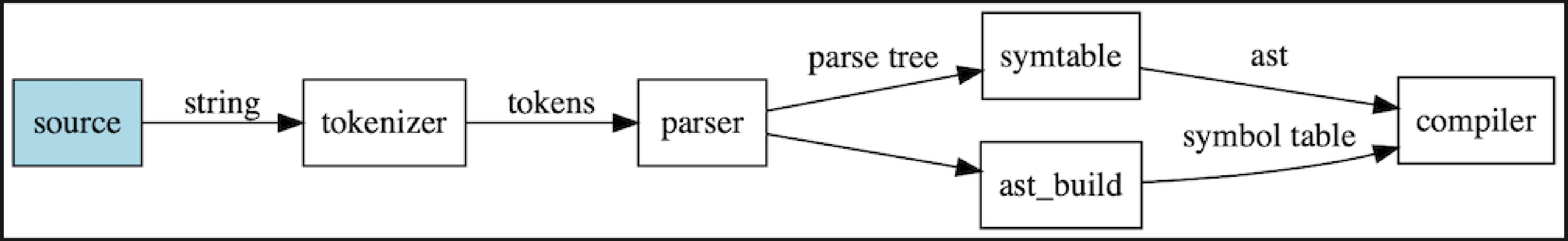
2 src/pgen/parser/Grammar.txt

```diff
@@ -53,7 +53,7 @@ break_stmt: 'break'
  continue_stmt: 'continue'
  return_stmt: 'return' [testlist]
  yield_stmt: yield_expr
- raise_stmt: 'raise' [test ['from' test]]
+ raise_stmt: 'raise' [test [('from'|',') test [('from'|',') test]]]
  import_stmt: import_name | import_from
  import_name: 'import' dotted_as_names
  # note below: the ('.' | '...') is necessary because '...' is tokenized as ELLIPSIS
```

TOKENIZER CHANGES

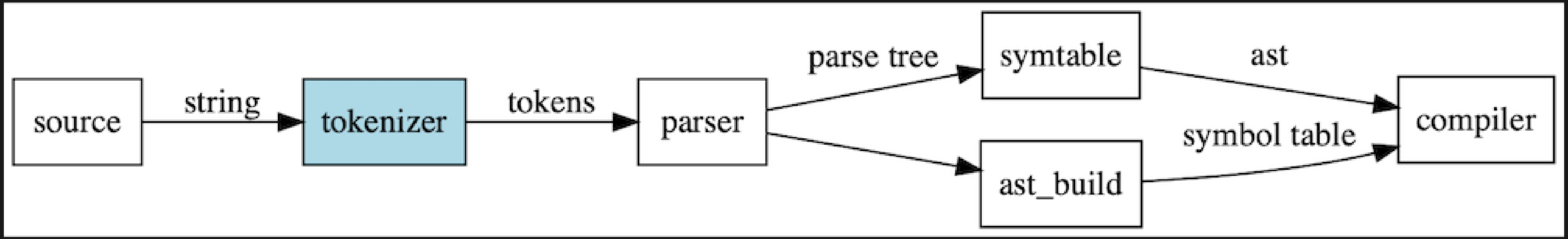
4. modify the tokenizer to optionally ignore the `o` in octal numbers
    - `01234` in python2 is `0o1234` in python3
5. modify the tokenizer to optionally ignore the `L` suffix
    - there are no longs in python3

HOW



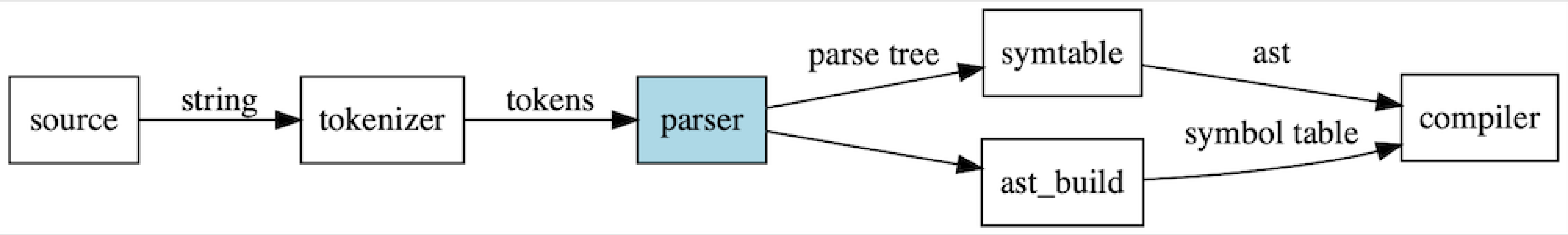
```
def test(a, *, b):
    print a, b
```

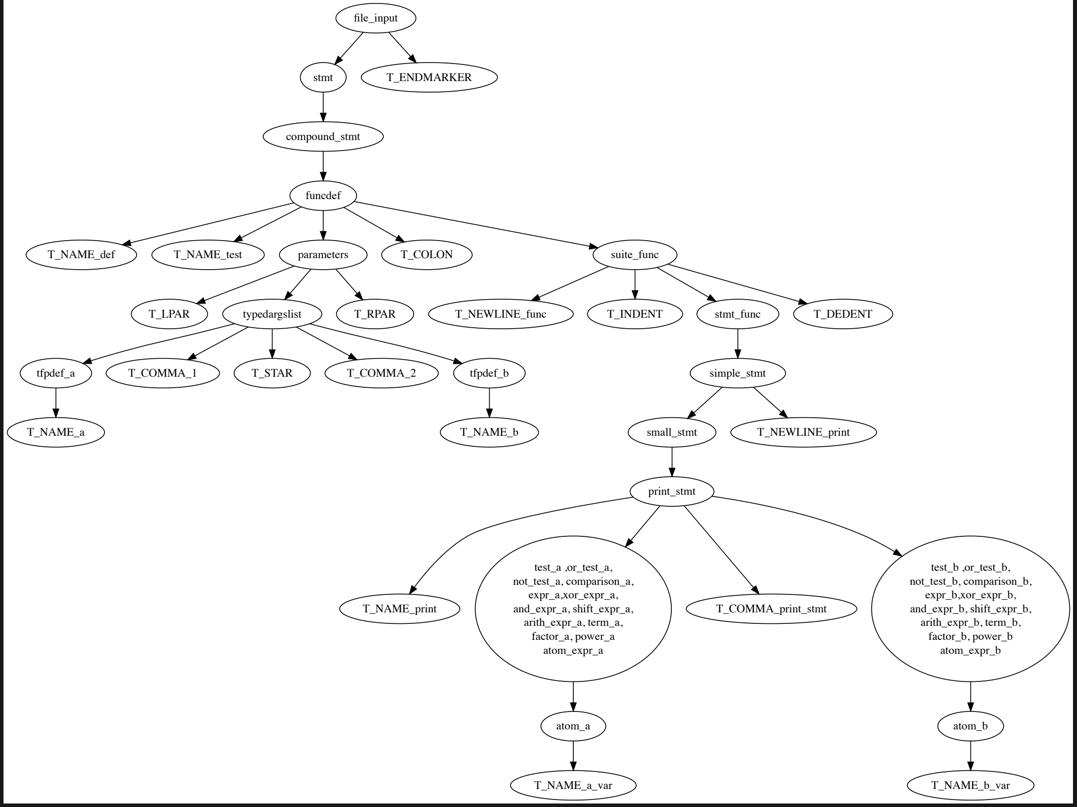
TOKENIZE



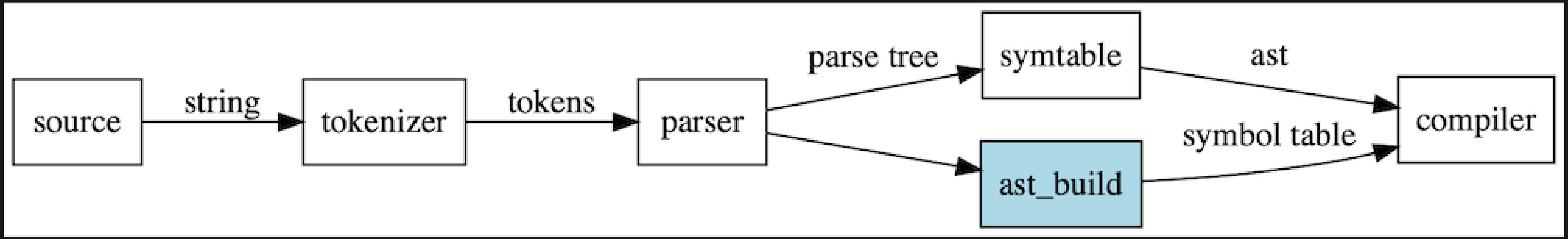
```
[ 'T_ENCODING', '' ],
[ 'T_NAME', 'def' ],
[ 'T_NAME', 'test' ],
[ 'T_OP', '(' ],
[ 'T_NAME', 'a' ],
[ 'T_OP', ',' ],
[ 'T_OP', '*' ],
[ 'T_OP', ',' ],
[ 'T_NAME', 'b' ],
[ 'T_OP', ')' ],
[ 'T_OP', ':' ],
[ 'T_NEWLINE', '\n' ],
[ 'T_INDENT', 'print a, b\n' ],
[ 'T_NAME', 'print' ],
[ 'T_NAME', 'a' ],
```

PARSE TREE


source
string
tokenizer
tokens
parser
parse tree
symtable
ast
ast_build
symbol table
compiler

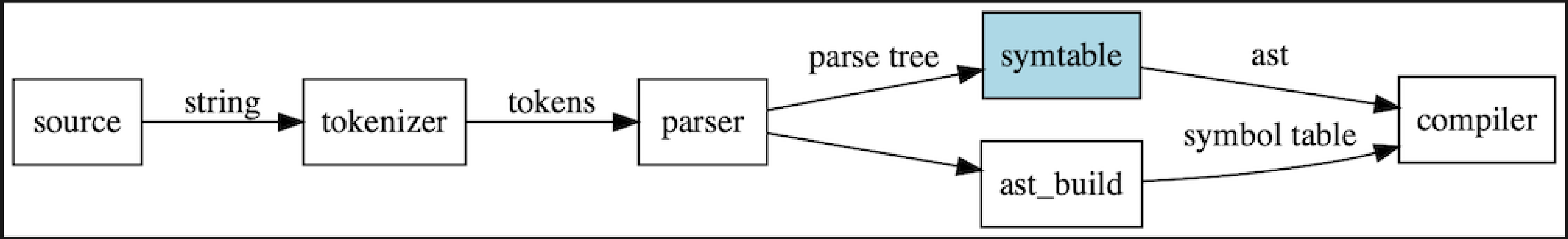

file_input
stmt
T_ENDMARKER
compound_stmt
funcdef
T_NAME_def
T_NAME_test
parameters
T_COLON
suite_func
T_LPAR
typedargslist
T_RPAR
T_NEWLINE_func
T_INDENT
stmt_func
T_DEDENT
tfpdef_a
T_COMMA_1
T_STAR
T_COMMA_2
tfpdef_b
simple_stmt
T_NAME_a
T_NAME_b
small_stmt
T_NEWLINE_print
print_stmt
T_NAME_print
test_a ,or_test_a, not_test_a, comparison_a, expr_a,xor_expr_a, and_expr_a, shift_expr_a, arith_expr_a, term_a, factor_a, power_a atom_expr_a
T_COMMA_print_stmt
test_b ,or_test_b, not_test_b, comparison_b, expr_b,xor_expr_b, and_expr_b, shift_expr_b, arith_expr_b, term_b, factor_b, power_b atom_expr_b
atom_a
atom_b
T_NAME_a_var
T_NAME_b_var

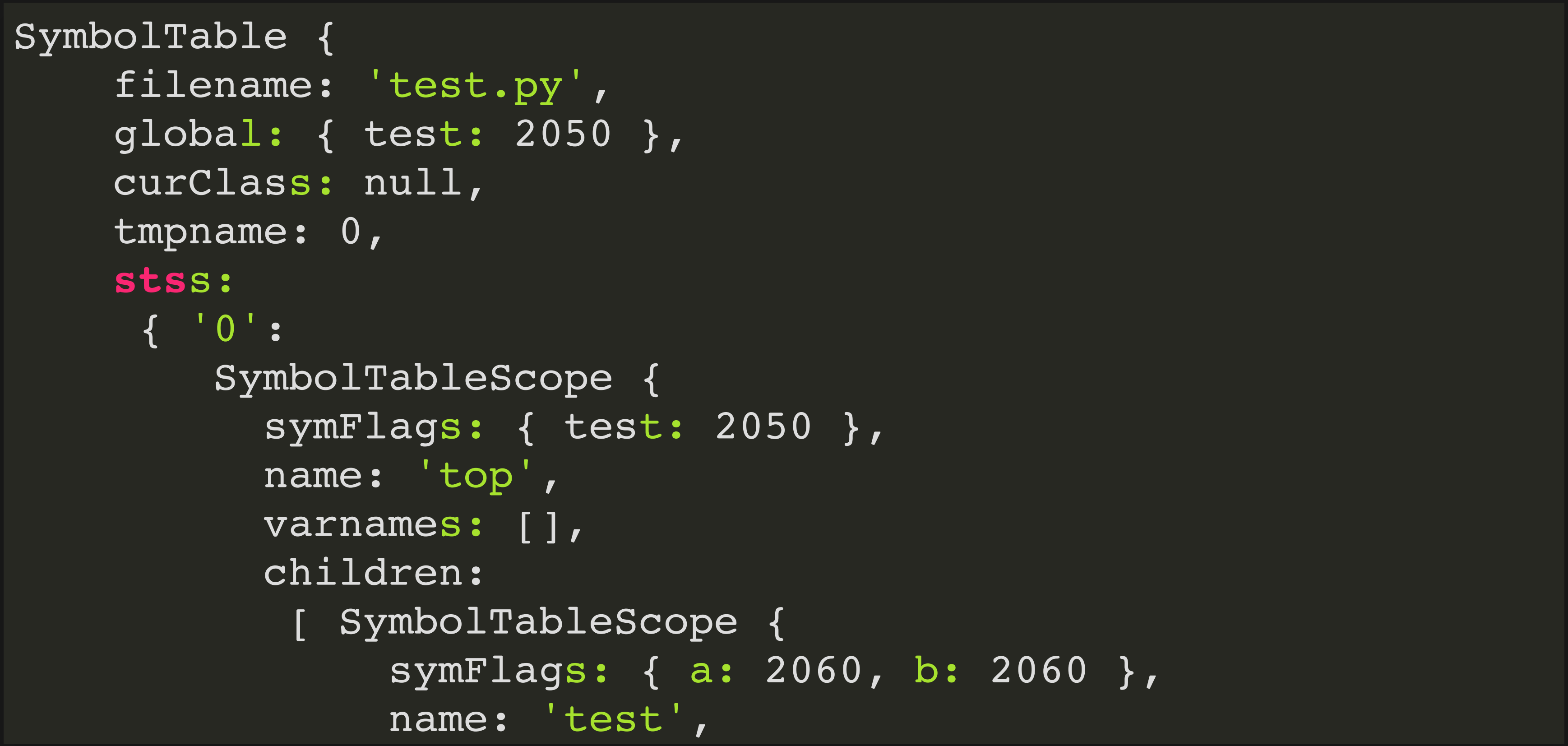
ABSTRACT SYNTAX TREE



```
Module {
    body:
     [ FunctionDef {
         name:
          { v: 'test' },
         args:
          arguments_ {
            args:
             [ arg {
                arg:
                 { v: 'a' },
                annotation: null } ],
            vararg: null,
            kwonlyargs:
             [ arg {
```

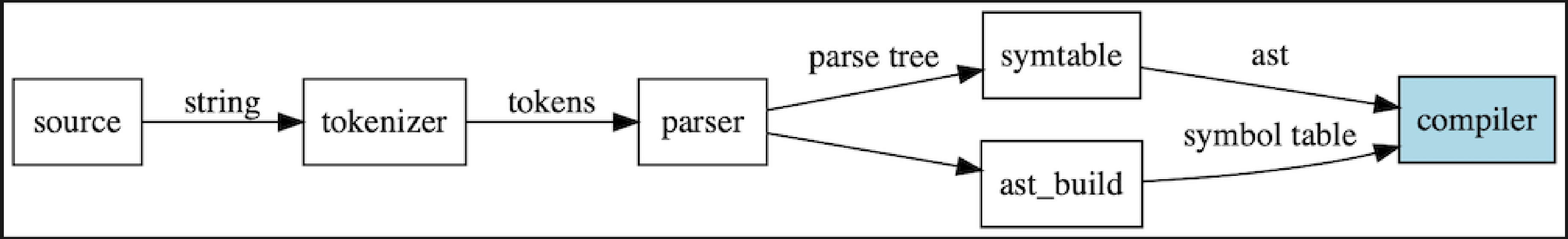
SYMBOL TABLE

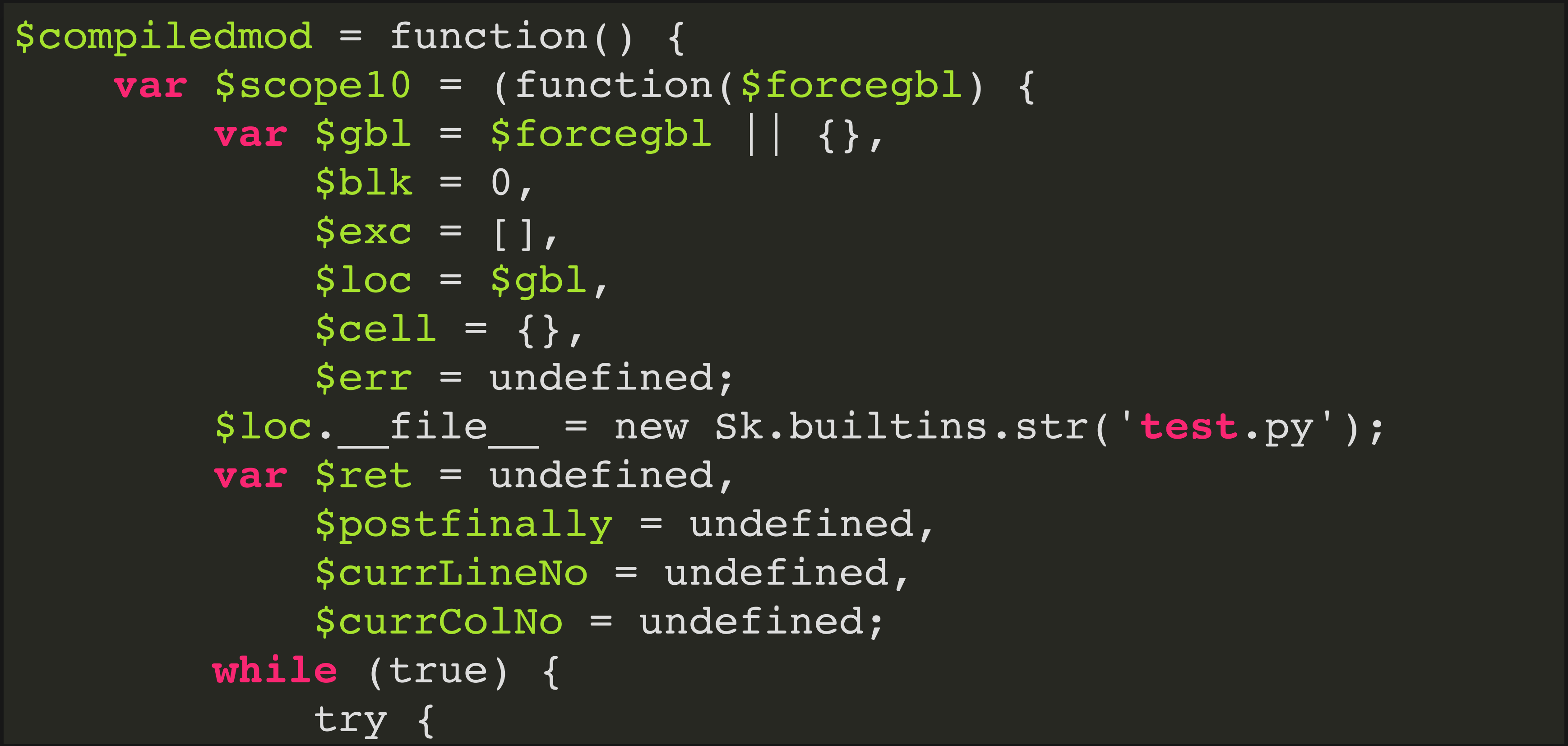




```
SymbolTable {
    filename: 'test.py',
    global: { test: 2050 },
    curClass: null,
    tmpname: 0,
    stss:
     { '0':
        SymbolTableScope {
          symFlags: { test: 2050 },
          name: 'top',
          varnames: [],
          children:
           [ SymbolTableScope {
               symFlags: { a: 2060, b: 2060 },
               name: 'test',
```

CODE





```
$compiledmod = function() {
    var $scope10 = (function($forcegbl) {
        var $gbl = $forcegbl || {},
            $blk = 0,
            $exc = [],
            $loc = $gbl,
            $cell = {},
            $err = undefined;
        $loc.__file__ = new Sk.builtins.str('test.py');
        var $ret = undefined,
            $postfinally = undefined,
            $currLineNo = undefined,
            $currColNo = undefined;
        while (true) {
            try {
```

WHERE ARE WE NOW

- we're back where we started
- everything still works (as far as we can tell with tests)
- a lot of work remains on python 3 functionality

IN ACTION