- **Loops:**

while_stmt: 'while' '(' test ')' [‘with’ annassign ; augassign] ‘:' expr ['else' ':' expr]
for_stmt: 'for' ‘(‘exprlist 'in' testlist ‘)' ':' suite ['else' ':' suite]
for2_stmt: ‘for’ ‘(annassign ‘;’ test ‘;’ augassign)’ ‘:’ suite ['else' ':' suite]

- **Condicionais:**

if_stmt: 'if' test ':' suite ('elif' test ':' suite)* ('else if' test ':' suite)* ['else' ':' suite]

- **Variáveis:**

type_specifier: string
| int
| float
| bool

- **Funções:**

funcdef: 'def' NAME parameters ['->' test] ':' suite

parameters: '(' [typedargslist] ')'
test: or_test ['if' or_test 'else' test] | lambdef
annassign: ':' test ['=' test]
augassign: ('+=' | '-=' | '*=' | '@=' | '/=' | '%=' | '&=' | '|=' | '^=' |
'<<=' | '>>=' | '**=' | '//=')
expr: xor_expr ('|' xor_expr)*
exprlist: (expr|star_expr) (',' (expr|star_expr))* [',']
star_expr: '*' expr
testlist: test (',' test)* [',']
suite: simple_stmt | NEWLINE INDENT stmt+ DEDENT
or_test: and_test ('or' and_test)*
and_test: not_test ('and' not_test)*
not_test: 'not' not_test | comparison
comparison: expr (comp_op expr)*
<> isn't actually a valid comparison operator in Python. It's here for the
sake of a __future__ import described in PEP 401 (which really works :-)
comp_op: '<'|'>'|'=='|'>='|'<='|'<>'|'!='|'in'|'not' 'in'|'is'|'is' 'not'
lambdef_nocond: 'lambda' [varargslist] ':' test_nocond
test_nocond: or_test | lambdef_nocond

xor_expr: and_expr ('^' and_expr)*
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)

```
expr_stmt: testlist_star_expr (annassign | augassign (yield_expr|testlist) |
                    ('=' (yield_expr|testlist_star_expr))*)
varargslist: (vfpdef ['=' test] (',' vfpdef ['=' test])* [',' [
        '*' [vfpdef] (',' vfpdef ['=' test])* [',' ['**' vfpdef [',']]]
      | '**' vfpdef [',']]]
  | '*' [vfpdef] (',' vfpdef ['=' test])* [',' ['**' vfpdef [',']]]
  | '**' vfpdef [',']
)
vfpdef: NAME
typedargslist: (tfpdef ['=' test] (',' tfpdef ['=' test])* [',' [
        '*' [tfpdef] (',' tfpdef ['=' test])* [',' ['**' tfpdef [',']]]
      | '**' tfpdef [',']]]
  | '*' [tfpdef] (',' tfpdef ['=' test])* [',' ['**' tfpdef [',']]]
  | '**' tfpdef [',']
tfpdef: NAME [':' test]
compound_stmt: if_stmt | while_stmt | for_stmt | try_stmt | with_stmt | funcdef | classdef |
decorated | async_stmt
testlist_star_expr: (test|star_expr) (',' (test|star_expr))* [',']
yield_expr: 'yield' [yield_arg]
yield_arg: 'from' test | testlist
and_expr: shift_expr ('&' shift_expr)*
shift_expr: arith_expr (('<<'|'>>') arith_expr)*
arith_expr: term (('+'|'-') term)*
term: factor (('*'|'@'|'/'|'%'|'//') factor)*
factor: ('+'|'-'|'~') factor | power
power: atom_expr ['**' factor]
atom_expr: ['await'] atom trailer*
atom: ('(' [yield_expr|testlist_comp] ')' |
       '[' [testlist_comp] ']' |
       '{' [dictorsetmaker] '}' |
       NAME | NUMBER | STRING+ | '...' | 'None' | 'True' | 'False')
testlist_comp: (test|star_expr) ( comp_for | (',' (test|star_expr))* [','] )
trailer: '(' [arglist] ')' | '[' subscriptlist ']' | '.' NAME
subscriptlist: subscript (',' subscript)* [',']
subscript: test | [test] ':' [test] [sliceop]
sliceop: ':' [test]
arglist: argument (',' argument)*  [',']
dictorsetmaker: ( ((test ':' test | '**' expr)
                   (comp_for | (',' (test ':' test | '**' expr))* [','])) |
                  ((test | star_expr)
                   (comp_for | (',' (test | star_expr))* [',']) )
comp_iter: comp_for | comp_if
sync_comp_for: 'for' exprlist 'in' or_test [comp_iter]
```

```
comp_for: ['async'] sync_comp_for
comp_if: 'if' test_nocond [comp_iter]
argument: ( test [comp_for] |
        test '=' test |
        '**' test |
        '*' test )
```