|  |  |  |  |
| --- | --- | --- | --- |
|  | Name | Concrete Syntax | Abstract syntax |
| Statements | Assignment | ID ‘ = ‘ Expression | (assign id exp) |
| Return\_stmt | ‘return‘ | ‘return‘ Expression | (return)| (return exp) |
| Global\_stmt | ‘global‘ ID | (global id) |
| Pass | ‘pass’ | (pass) |
| Continue | ‘continue’ | (continue) |
| Break | ‘break’ | (break) |
| Fucntion\_def | ‘def‘ ID ‘(‘ Params ‘)‘ ‘ : ‘ Statements |  ‘def‘ ID ‘() : ‘ Statements | (proc id (param1 param2 …) (stmt1 stmt2 …)) |
| params | Param\_with\_default | Params ‘, ‘ Param\_with\_default | (param1 param2 …) |
| Param\_with\_default | ID ‘ = ‘ Expression | (default\_param id expression) |
| If\_stmt | ‘if ‘ Expression ‘ : ‘ Statements ‘else‘ ‘ : ‘ Statements | (if expression stmts1 stmts2) |
| For\_stmt | ‘for‘ ID ‘in‘ Expression ‘ : ‘ Statements | (for id expression stmts) |
| Expressions | Disjunction | Disjunction ‘or‘ Conjunction | (or disjunction conjunction) |
| Conjunction | Conjunction ‘and‘ Inversion | (and conjunction inversion) |
| Inversion | ‘not‘ Inversion | (not inversion) |
| Comparison | Sum compare\_op\_Sum\_pairs | (comparison sum compare\_op\_pairs) |
| Compare\_op\_Sum\_pairs | - | (compare1 compare2 …) |
| Eq\_Sum | ‘ == ‘ Sum | (equal sum) |
| Lt\_Sum | ‘ < ‘ Sum | (less sum) |
| Gt\_Sum | ‘ > ‘ Sum | (more sum) |
| Sum | Sum ‘ + ‘ Term | (plus sum term) |
| Sum | Sum ‘ - ‘ Term | (minus sum term) |
| Term | Term ‘ ∗ ‘ Factor | (star term factor) |
| Term | Term ‘ / ‘ Factor | (slash term factor) |
| Factor | ‘ + ‘ Factor | (plus factor) |
| Factor | ‘ - ‘ Factor | (minus factor) |
| Power | Atom ‘ ∗ ∗‘ Factor | (doublestar atom factor) |
| Primary | Primary ‘[‘ Expression ‘]‘ | (index primary expression) |
| Primary | Primary ‘()‘ | Primary ‘(‘ Arguments ‘)‘ | (call primary (argument1 argument2 …)) |
| Arguments | Expression | Arguments ‘, ‘ Expression | (exp1 exp2 … ) |
| Atom | ID | (var id) |
| Atom | True | (true) |
| Atom | False | (false) |
| Atom | None | (none) |
| Atom | NUMBER | (num number) |
| List | ‘[‘ Expressions ‘]‘ | ‘[]‘ | (list (exp1 exp2 …)) |
| Print | Print ‘(’ Atom’)’ | (print atom) |