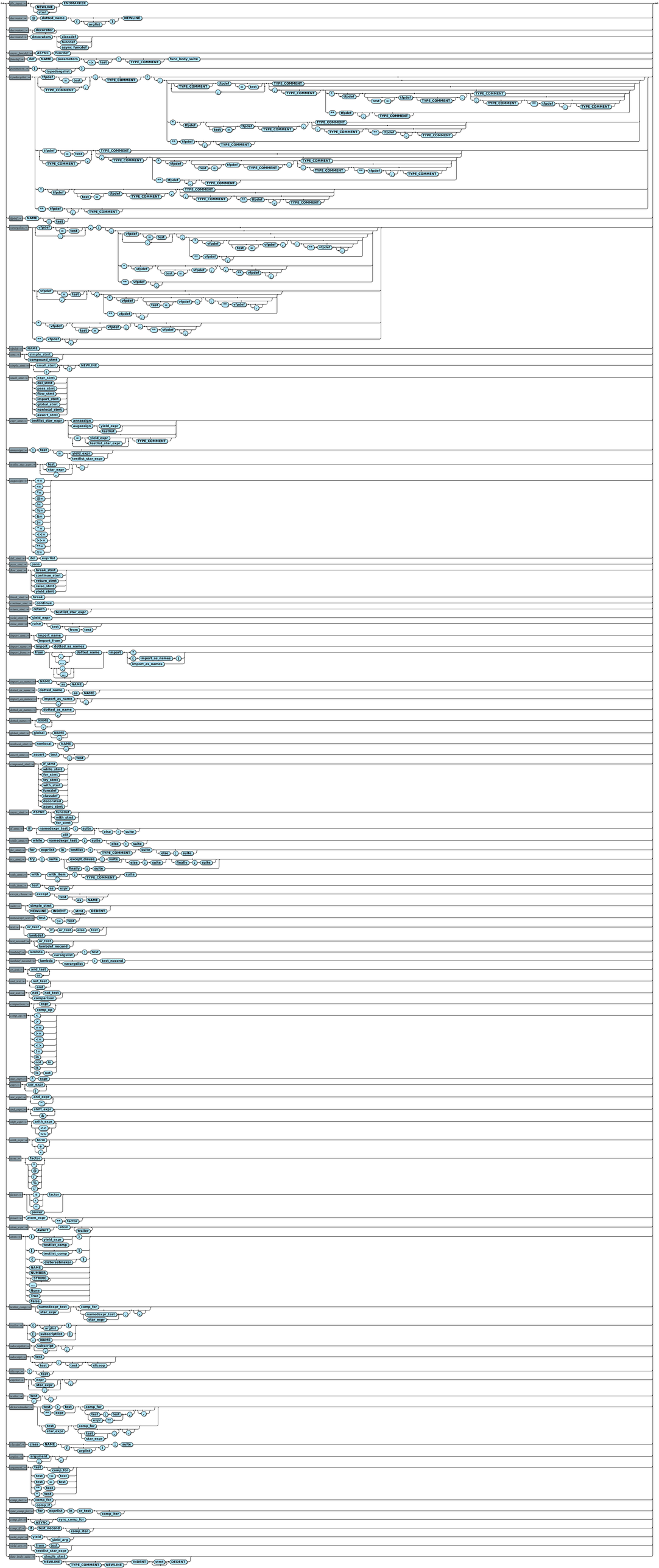
file_input
ENDMARKER
NEWLINE
stmt
decorator
@
dotted_name
(
arglist
)
NEWLINE
decorators
decorated
classdef
funcdef
async_funcdef
async_funcdef
ASYNC
funcdef
def
NAME
parameters
->
test
TYPE_COMMENT
func_body_suite
parameters
(
)
typedargslist
tfpdef
:
test
=
TYPE_COMMENT
,
*
**
vfpdef
name
simple_stmt
small_stmt
;
NEWLINE
compound_stmt
expr_stmt
del_stmt
pass_stmt
flow_stmt
import_stmt
global_stmt
nonlocal_stmt
assert_stmt
testlist_star_expr
annassign
augassign
yield_expr
testlist
=
testlist_star_expr
+=
-=
*=
@=
/=
%=
&=
|=
^=
<<=
>>=
**=
//=
del_stmt
del
exprlist
pass_stmt
pass
flow_stmt
break_stmt
continue_stmt
return_stmt
raise_stmt
yield_stmt
break_stmt
break
continue_stmt
continue
return_stmt
return
testlist_star_expr
yield_expr
raise_stmt
raise
test
from
import_stmt
import_name
import_from
import_name
import
dotted_as_names
import_from
from
.
...
dotted_name
import
*
(
import_as_names
)
import_as_name
dotted_as_name
NAME
as
dotted_as_names
dotted_name
global_stmt
global
NAME
nonlocal_stmt
nonlocal
NAME
assert_stmt
assert
test
compound_stmt
if_stmt
while_stmt
for_stmt
try_stmt
with_stmt
funcdef
classdef
decorated
async_stmt
async_stmt
ASYNC
funcdef
with_stmt
for_stmt
if_stmt
if
namedexpr_test
:
suite
elif
else
while_stmt
while
namedexpr_test
suite
else
for_stmt
for
exprlist
in
testlist
suite
else
try_stmt
try
suite
except_clause
suite
else
finally
with_stmt
with
with_item
suite
with_item
test
as
expr
except_clause
except
test
as
NAME
suite
simple_stmt
NEWLINE
INDENT
stmt
DEDENT
namedexpr_test
test
:=
test
test
if
or_test
else
test
or_test
and_test
or
lambdef
lambdef_nocond
lambdef
lambda
:
test
varargslist
lambdef_nocond
lambda
varargslist
:
test_nocond
and_test
not_test
and
not_test
not_test
not
comparison
comparison
expr
comp_op
comp_op
<
>
==
>=
<=
<>
!=
in
not
in
is
is
not
star_expr
*
expr
expr
xor_expr
|
xor_expr
xor_expr
and_expr
^
and_expr
shift_expr
arith_expr
shift_expr
<<
>>
arith_expr
term
+
-
term
factor
*
@
/
%
//
factor
+
-
~
power
power
atom_expr
**
factor
atom_expr
AWAIT
atom
trailer
atom
(
)
yield_expr
testlist_comp
[
]
{
}
dictorsetmaker
NAME
NUMBER
STRING
...
None
True
False
testlist_comp
namedexpr_test
comp_for
star_expr
namedexpr_test
,
trailer
(
arglist
)
[
subscriptlist
]
.
NAME
subscriptlist
subscript
,
subscript
subscript
test
:
test
sliceop
sliceop
:
test
exprlist
star_expr
,
testlist
test
dictorsetmaker
test
:
test
comp_for
**
expr
test
:
test
,
expr
**
test
comp_for
star_expr
classdef
class
NAME
(
arglist
)
:
suite
arglist
argument
,
argument
test
comp_for
test
=
test
test
:=
test
**
test
*
test
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
yield_expr
yield
yield_arg
from
test
testlist_star_expr
func_body_suite
simple_stmt
NEWLINE
TYPE_COMMENT
NEWLINE
INDENT
stmt
DEDENT