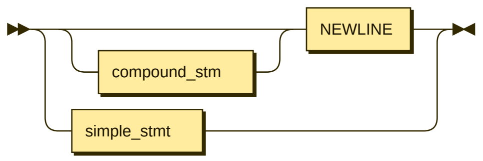
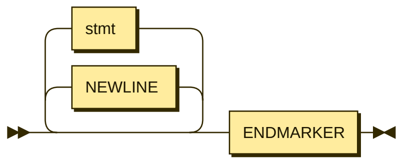
single_input:



```
single_input
         ::= compound_stm? NEWLINE
           | simple_stmt
```

no references

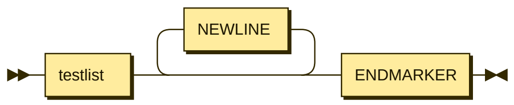
file_input:



```
file_input
         ::= ( NEWLINE | stmt )* ENDMARKER
```

no references

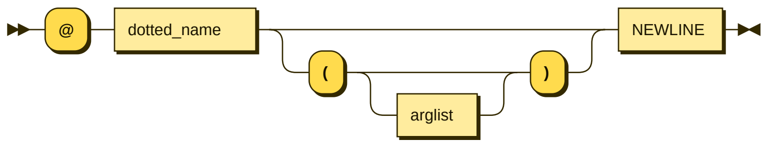
eval_input:



```
eval_input
         ::= testlist NEWLINE* ENDMARKER
```

no references

decorator:

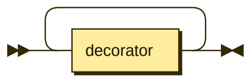


```
decorator
         ::= '@' dotted_name ( '(' arglist? ')' )? NEWLINE
```

referenced by:

- decorators

decorators:

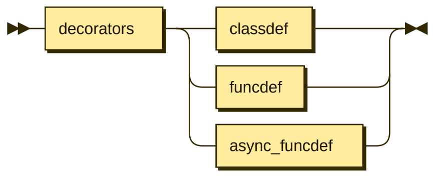


```
decorators
         ::= decorator+
```

referenced by:

- decorated

decorated:

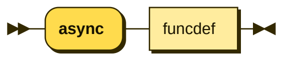


```
decorated
         ::= decorators ( classdef | funcdef | async_funcdef )
```

referenced by:

- compound_stmt

async_funcdef:

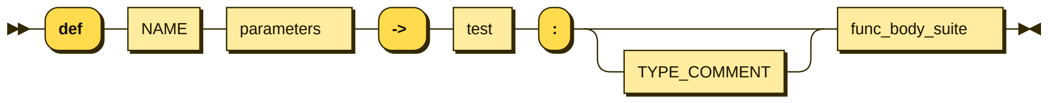


```
async_funcdef
         ::= 'async' funcdef
```

referenced by:

- decorated

funcdef:

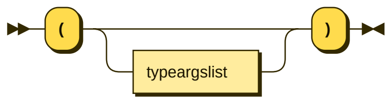


```
funcdef  ::= 'def' NAME parameters '->' test ':' TYPE_COMMENT? func_body_suite
```

referenced by:

- async_funcdef
- async_stmt
- compound_stmt
- decorated

parameters:

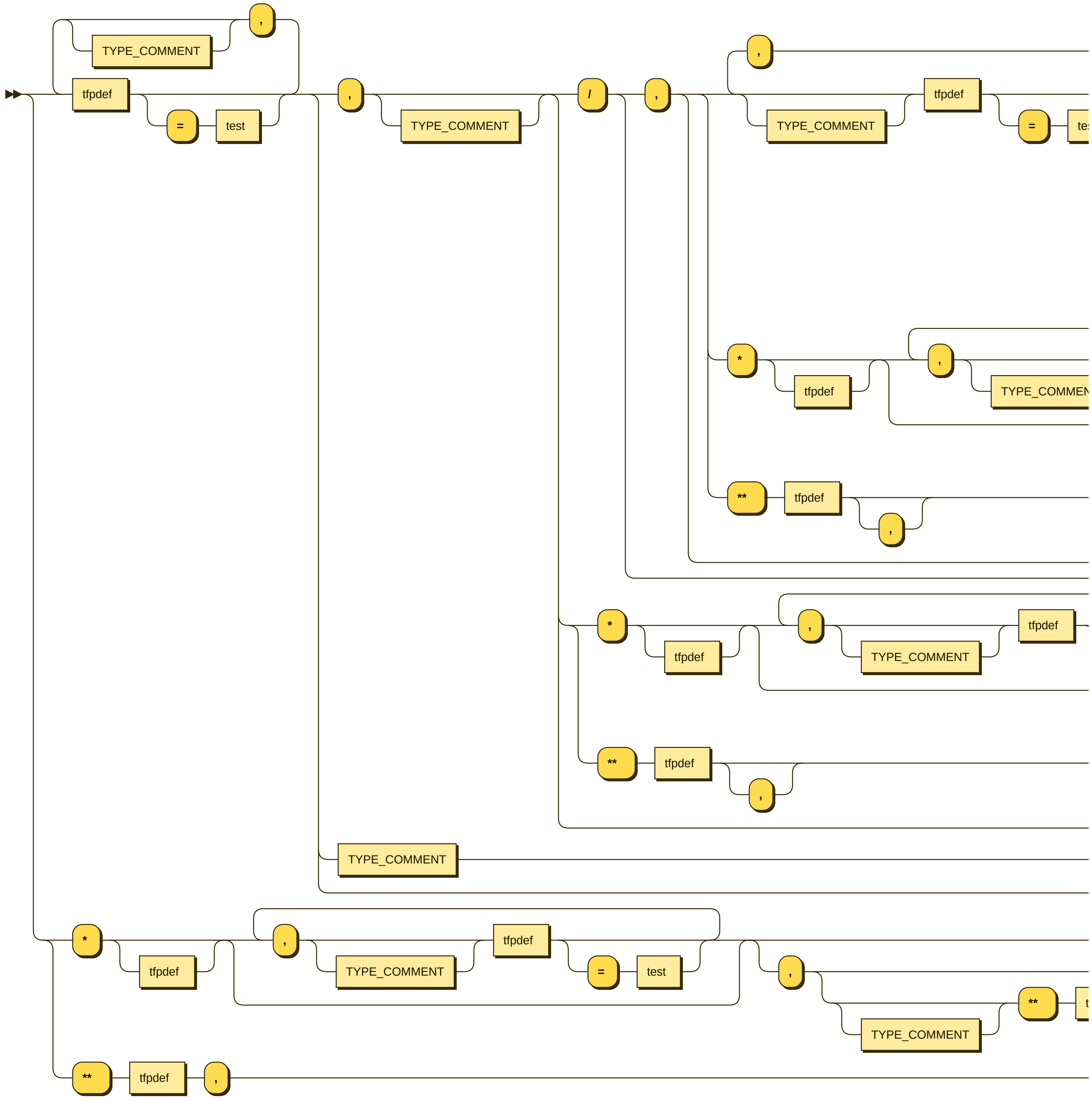


```
parameters
         ::= '(' typeargslist? ')'
```

referenced by:

- funcdef

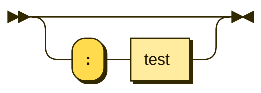
typedargslist:



```
typedargslist
         ::= tfpdef ( '=' test )? ( ',' TYPE_COMMENT? tfpdef ( '=' test )? )* ( ',' TYPE_COMMENT? ( '/' ( ',' ( ( TYPE_COMMENT? tfpdef (
             '=' test )? ( ',' TYPE_COMMENT? tfpdef ( '=' test )? )* ( ',' ( TYPE_COMMENT? ( '*' tfpdef? ( ',' TYPE_COMMENT? tfpdef ( '='
             test )? )* ( ',' ( TYPE_COMMENT? '**' tfpdef ','? )? )? | '**' tfpdef ','? ) )? )? | '*' tfpdef? ( ',' TYPE_COMMENT? tfpdef
             ( '=' test )? )* ( ',' ( TYPE_COMMENT? '**' tfpdef ',' )? )? | '**' tfpdef ','? ) TYPE_COMMENT? )? )? | ( '*' tfpdef? ( ','
             TYPE_COMMENT? tfpdef ( '=' test )? )* ( ',' ( TYPE_COMMENT? '**' tfpdef ','? )? )? | '**' tfpdef ','? ) TYPE_COMMENT? )? |
             TYPE_COMMENT )?
           | ( '*' tfpdef? ( ',' TYPE_COMMENT? tfpdef ( '=' test )? )* ( ',' ( TYPE_COMMENT? '**' tfpdef ','? )? )? | '**' tfpdef ',' )
             TYPE_COMMENT?
```

no references

tfpdef:

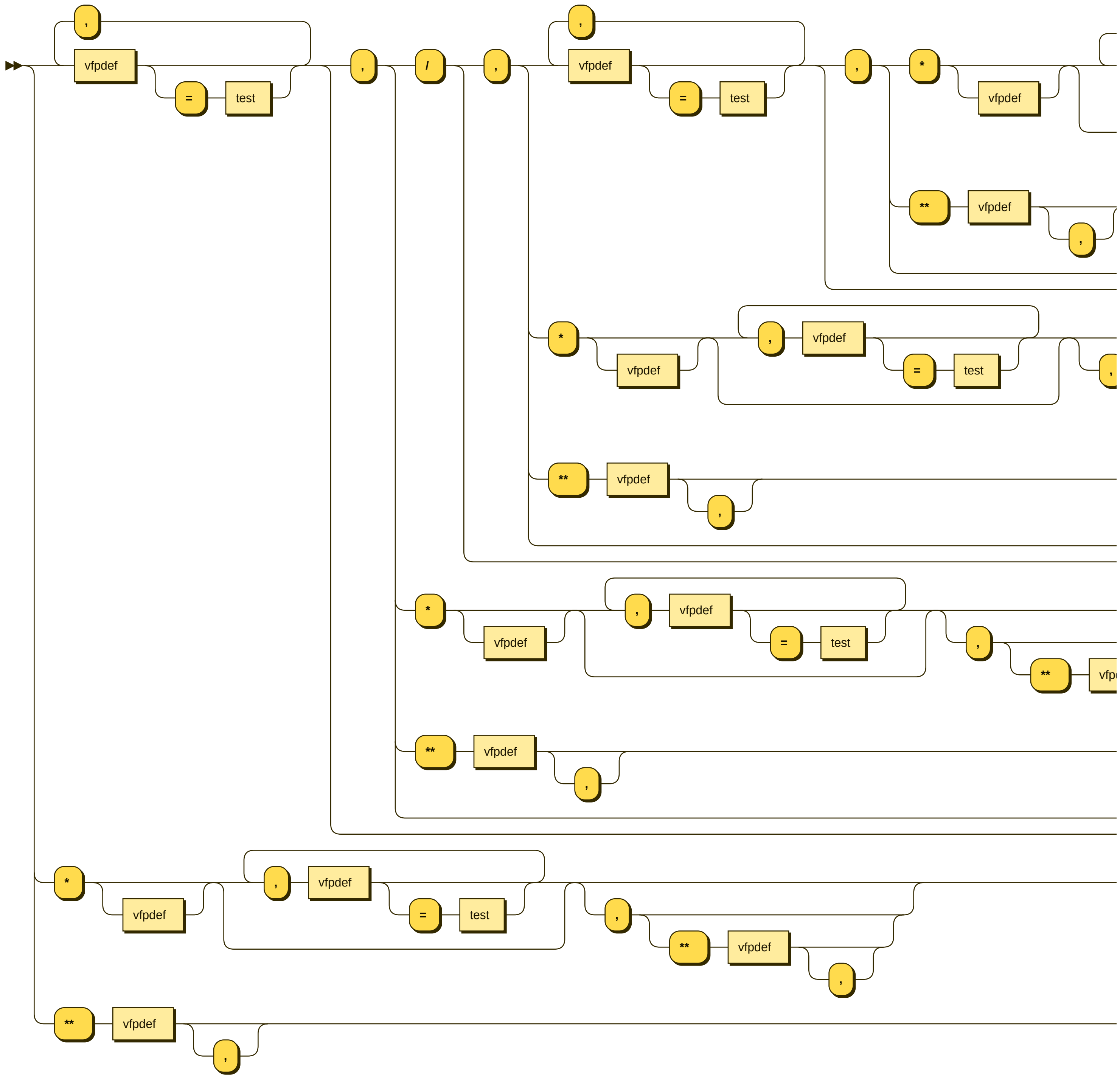


```
tfpdef   ::= ( ':' test )?
```

referenced by:

- typedargslist

varargslist:

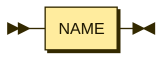


```
varargslist
         ::= vfpdef ( '=' test )? ( ',' vfpdef ( '=' test )? )* ( ',' ( '/' ( ',' ( vfpdef ( '=' test )? ( ',' vfpdef ( '=' test )? )* (
             ',' ( '*' vfpdef? ( ',' vfpdef ( '=' test )? )* ( ',' ( '**' vfpdef ','? )? )? | '**' vfpdef ','? )? )? | '*' vfpdef? ( ','
             vfpdef ( '=' test )? )* ( ',' ( '**' vfpdef ','? )? )? | '**' vfpdef ','? )? )? | '*' vfpdef? ( ',' vfpdef ( '=' test )? )*
             ( ',' ( '**' vfpdef ','? )? )? | '**' vfpdef ','? )? )?
           | '*' vfpdef? ( ',' vfpdef ( '=' test )? )* ( ',' ( '**' vfpdef ','? )? )?
           | '**' vfpdef ','?
```

referenced by:

- lambdef
- lambdef_nocond

vfpdef:

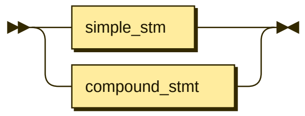


```
vfpdef   ::= NAME
```

referenced by:

- varargslist

stmt:

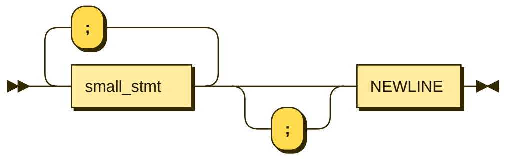


```
stmt     ::= simple_stm
           | compound_stmt
```

referenced by:

- file_input
- func_body_suite
- suite

simple_stmt:

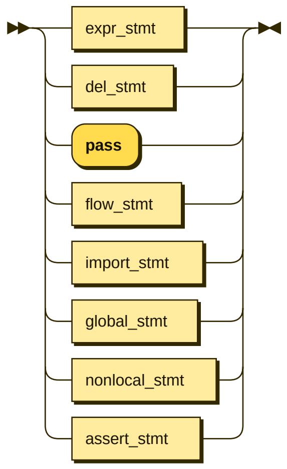


```
simple_stmt
         ::= small_stmt ( ';' small_stmt )* ';'? NEWLINE
```

referenced by:

- func_body_suite
- single_input
- suite

small_stmt:

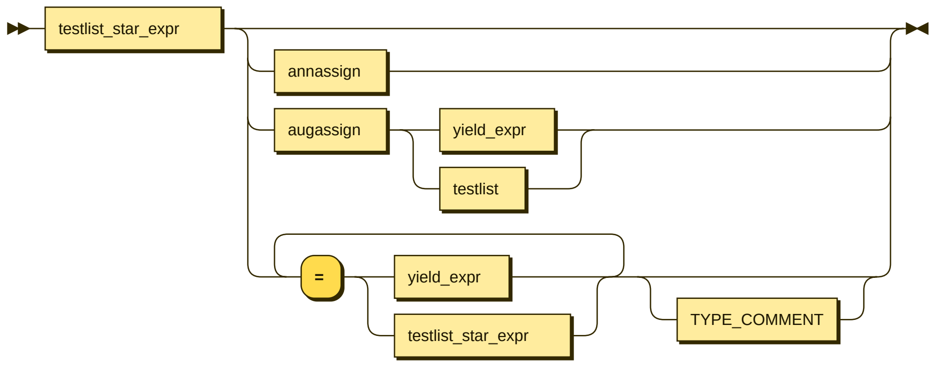


```
small_stmt
         ::= expr_stmt
           | del_stmt
           | 'pass'
           | flow_stmt
           | import_stmt
           | global_stmt
           | nonlocal_stmt
           | assert_stmt
```

referenced by:

- simple_stmt

expr_stmt:

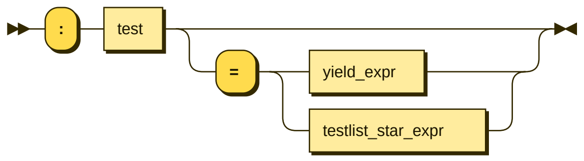


```
expr_stmt
         ::= testlist_star_expr ( annassign | augassign ( yield_expr | testlist ) | ( '=' ( yield_expr | testlist_star_expr ) )+
             TYPE_COMMENT? )?
```

referenced by:

- small_stmt

annassign:

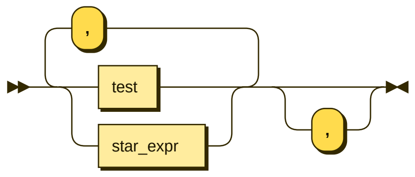


```
annassign
         ::= ':' test ( '=' ( yield_expr | testlist_star_expr ) )?
```

referenced by:

- expr_stmt

testlist_star_expr:

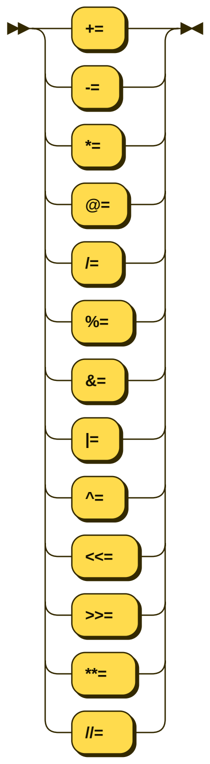


```
testlist_star_expr
         ::= ( test | star_expr ) ( ',' ( test | star_expr ) )* ','?
```

referenced by:

- annassign
- expr_stmt
- yield_arg

augassign:

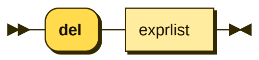


```
augassign
         ::= '+='
           | '-='
           | '*='
           | '@='
           | '/='
           | '%='
           | '&='
           | '|='
           | '^='
           | '<<='
           | '>>='
           | '**='
           | '//='
```

referenced by:

- expr_stmt

del_stmt:

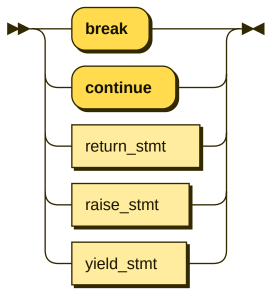


```
del_stmt ::= 'del' exprlist
```

referenced by:

- small_stmt

flow_stmt:

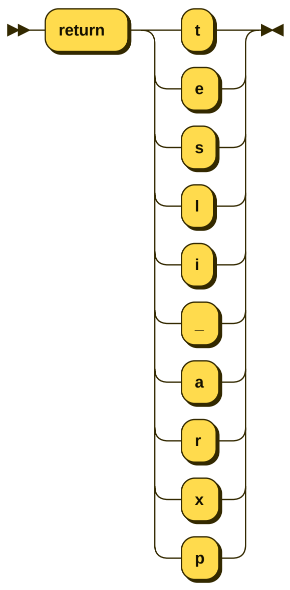


```
flow_stmt
         ::= 'break'
           | 'continue'
           | return_stmt
           | raise_stmt
           | yield_stmt
```

referenced by:

- small_stmt

return_stmt:

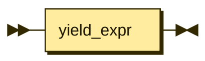


```
return_stmt
         ::= 'return' [tesli_arxp]
```

referenced by:

- flow_stmt

yield_stmt:

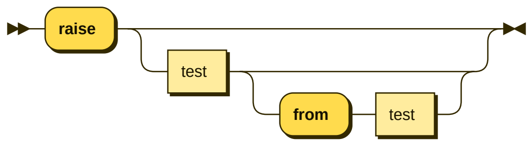


```
yield_stmt
         ::= yield_expr
```

referenced by:

- flow_stmt

raise_stmt:

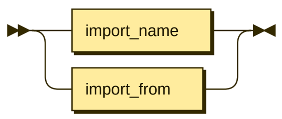


```
raise_stmt
         ::= 'raise' ( test ( 'from' test )? )?
```

referenced by:

- flow_stmt

import_stmt:

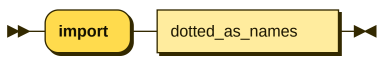


```
import_stmt
         ::= import_name
           | import_from
```

referenced by:

- small_stmt

import_name:

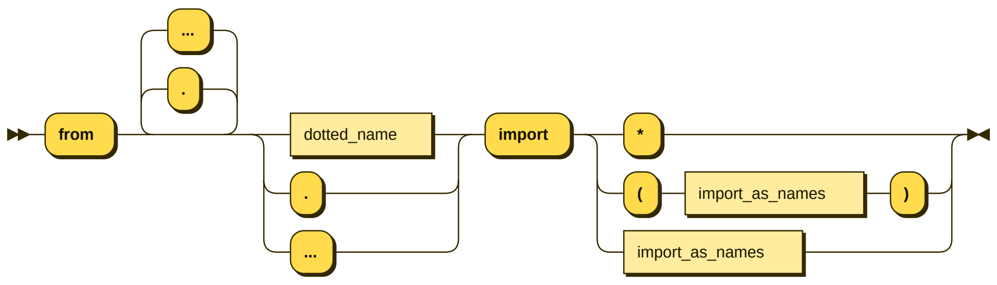


```
import_name
         ::= 'import' dotted_as_names
```

referenced by:

- import_stmt

import_from:

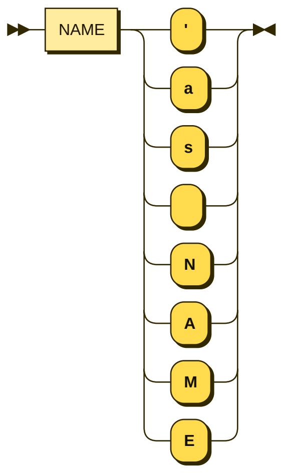


```
import_from
         ::= 'from' ( '.' | '...' )* ( dotted_name | '.' | '...' ) 'import' ( '*' | '(' import_as_names ')' | import_as_names )
```

referenced by:

- import_stmt

import_as_name:

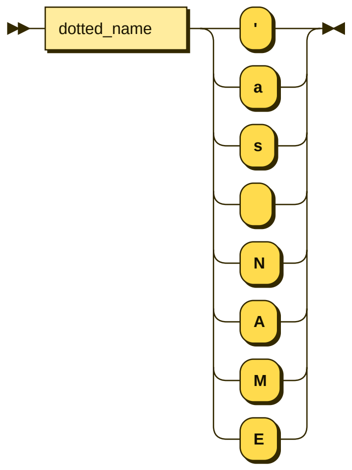


```
import_as_name
         ::= NAME ['as NAME]
```

referenced by:

- import_as_names

dotted_as_name:

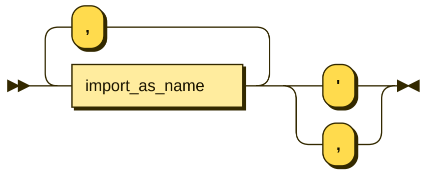


```
dotted_as_name
         ::= dotted_name ['as NAME]
```

referenced by:

- dotted_as_names

import_as_names:

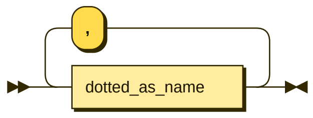


```
import_as_names
         ::= import_as_name ( ',' import_as_name )* [',]
```

referenced by:

- import_from

dotted_as_names:

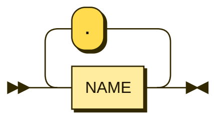


```
dotted_as_names
         ::= dotted_as_name ( ',' dotted_as_name )*
```

referenced by:

- import_name

dotted_name:

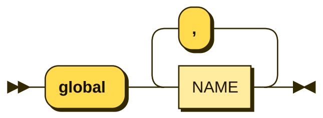


```
dotted_name
         ::= NAME ( '.' NAME )*
```

referenced by:

- decorator
- dotted_as_name
- import_from

global_stmt:

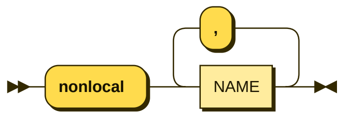


```
global_stmt
         ::= 'global' NAME ( ',' NAME )*
```

referenced by:

- small_stmt

nonlocal_stmt:

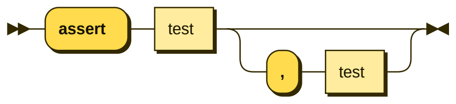


```
nonlocal_stmt
         ::= 'nonlocal' NAME ( ',' NAME )*
```

referenced by:

- small_stmt

assert_stmt:

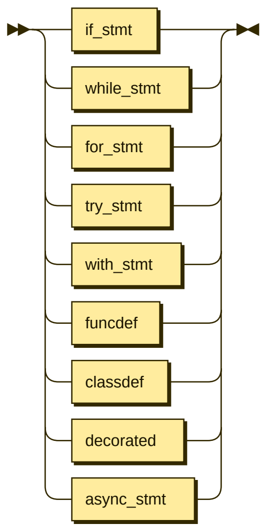


```
assert_stmt
         ::= 'assert' test ( ',' test )?
```

referenced by:

- small_stmt

compound_stmt:

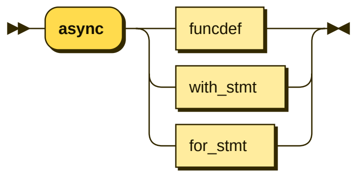


```
compound_stmt
         ::= if_stmt
           | while_stmt
           | for_stmt
           | try_stmt
           | with_stmt
           | funcdef
           | classdef
           | decorated
           | async_stmt
```

referenced by:

- stmt

async_stmt:

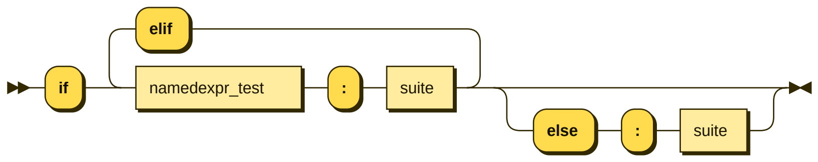


```
async_stmt
         ::= 'async' ( funcdef | with_stmt | for_stmt )
```

referenced by:

- compound_stmt

if_stmt:

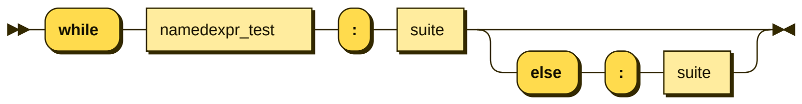


```
if_stmt  ::= 'if' namedexpr_test ':' suite ( 'elif' namedexpr_test ':' suite )* ( 'else' ':' suite )?
```

referenced by:

- compound_stmt

while_stmt:

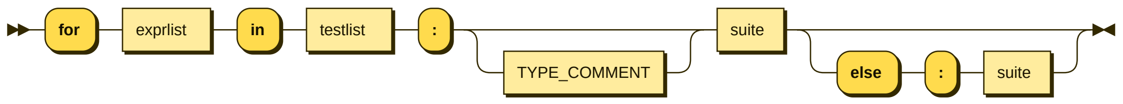


```
while_stmt
         ::= 'while' namedexpr_test ':' suite ( 'else' ':' suite )?
```

referenced by:

- compound_stmt

for_stmt:

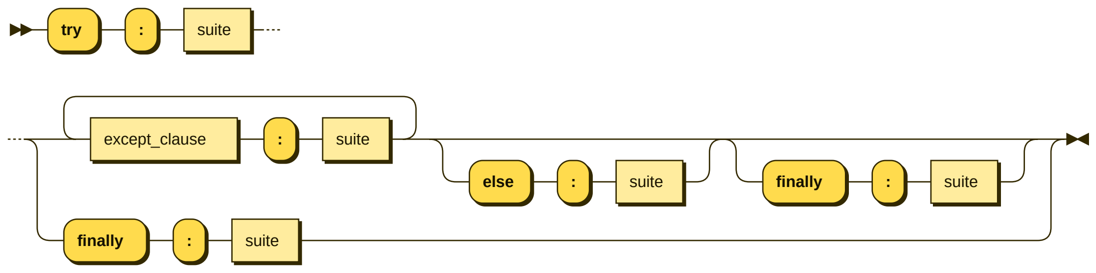


```
for_stmt ::= 'for' exprlist 'in' testlist ':' TYPE_COMMENT? suite ( 'else' ':' suite )?
```

referenced by:

- async_stmt
- compound_stmt

try_stmt:

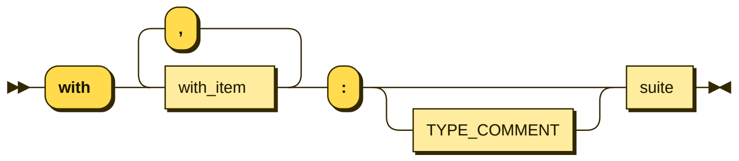


```
try_stmt ::= 'try' ':' suite ( ( except_clause ':' suite )+ ( 'else' ':' suite )? ( 'finally' ':' suite )? | 'finally' ':' suite )
```

referenced by:

- compound_stmt

with_stmt:

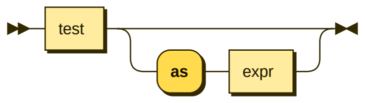


```
with_stmt
         ::= 'with' with_item ( ',' with_item )* ':' TYPE_COMMENT? suite
```

referenced by:

- async_stmt
- compound_stmt

with_item:

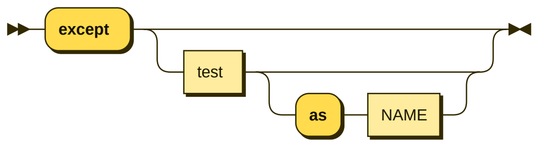


```
with_item
         ::= test ( 'as' expr )?
```

referenced by:

- with_stmt

except_clause:

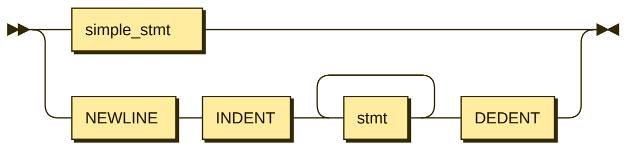


```
except_clause
         ::= 'except' ( test ( 'as' NAME )? )?
```

referenced by:

- try_stmt

suite:

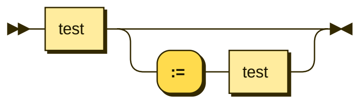


```
suite    ::= simple_stmt
           | NEWLINE INDENT stmt+ DEDENT
```

referenced by:

- classdef
- for_stmt
- if_stmt
- try_stmt
- while_stmt
- with_stmt

namedexpr_test:

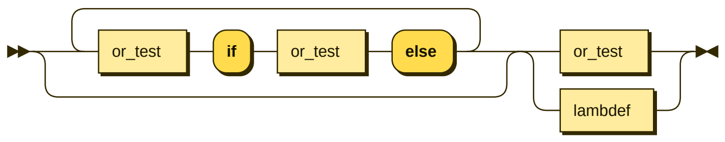


```
namedexpr_test
         ::= test ( ':=' test )?
```

referenced by:

- if_stmt
- testlist_comp
- while_stmt

test:

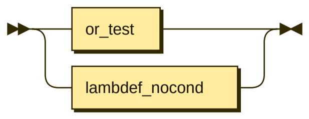


```
test     ::= ( or_test 'if' or_test 'else' )* ( or_test | lambdef )
```

referenced by:

- annassign
- argument
- assert_stmt
- dictorsetmaker
- except_clause
- func_type
- funcdef
- lambdef
- namedexpr_test
- raise_stmt
- sliceop
- subscript
- testlist
- testlist_star_expr
- tfpdef
- typedargslist
- typelist
- varargslist
- with_item
- yield_arg

test_nocond:

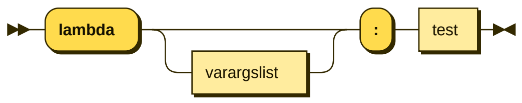


```
test_nocond
         ::= or_test
           | lambdef_nocond
```

referenced by:

- comp_if
- lambdef_nocond

lambdef:

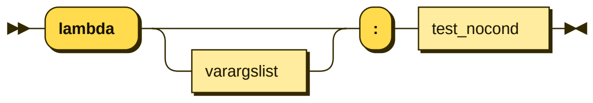


```
lambdef  ::= 'lambda' varargslist? ':' test
```

referenced by:

- test

lambdef_nocond:

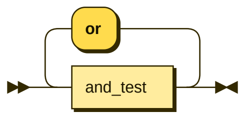


```
lambdef_nocond
         ::= 'lambda' varargslist? ':' test_nocond
```

referenced by:

- test_nocond

or_test:

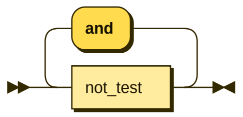


```
or_test  ::= and_test ( 'or' and_test )*
```

referenced by:

- sync_comp_for
- test
- test_nocond

and_test:

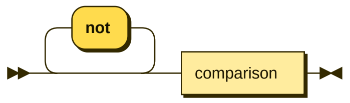


```
and_test ::= not_test ( 'and' not_test )*
```

referenced by:

- or_test

not_test:

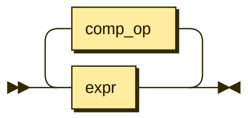


```
not_test ::= 'not'* comparison
```

referenced by:

- and_test

comparison:

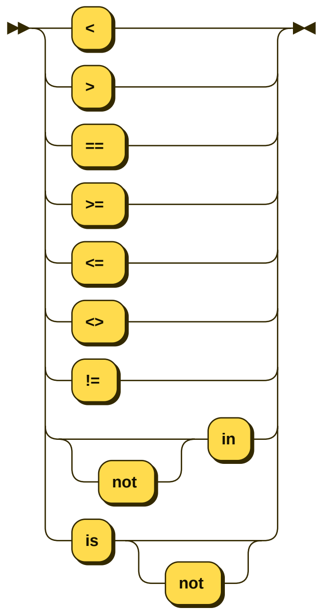


```
comparison
         ::= expr ( comp_op expr )*
```

referenced by:

- not_test

comp_op:



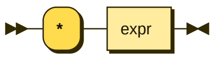
```
comp_op  ::= '<'
           | '>'
           | '=='
```

```
              | '>='
              | '<='
              | '<>'
              | '!='
              | 'not'? 'in'
              | 'is' 'not'?
```

referenced by:

- comparison

star_expr:

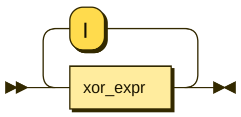


```
star_expr
         ::= '*' expr
```

referenced by:

- dictorsetmaker
- exprlist
- testlist_comp
- testlist_star_expr

expr:

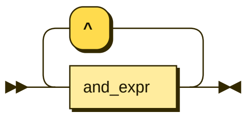


```
expr     ::= xor_expr ( '|' xor_expr )*
```

referenced by:

- comparison
- dictorsetmaker
- exprlist
- star_expr
- with_item

xor_expr:

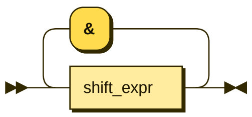


```
xor_expr ::= and_expr ( '^' and_expr )*
```

referenced by:

- expr

and_expr:

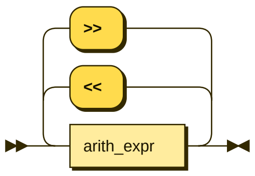


```
and_expr ::= shift_expr ( '&' shift_expr )*
```

referenced by:

- xor_expr

shift_expr:

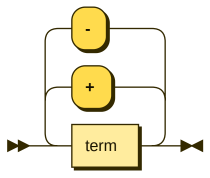


```
shift_expr
         ::= arith_expr ( ( '<<' | '>>' ) arith_expr )*
```

referenced by:

- and_expr

arith_expr:

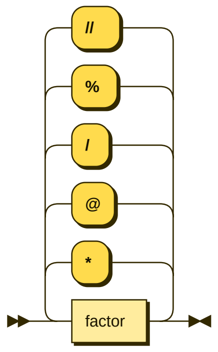


```
arith_expr
         ::= term ( ( '+' | '-' ) term )*
```

referenced by:

- shift_expr

term:

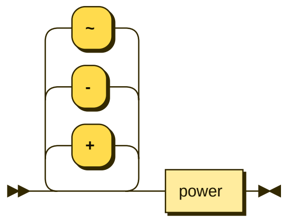


```
term     ::= factor ( ( '*' | '@' | '/' | '%' | '//' ) factor )*
```

referenced by:

- arith_expr

factor:

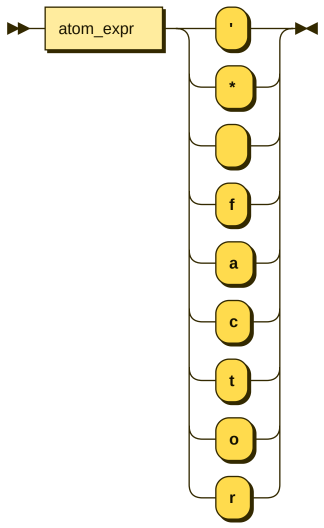


```
factor   ::= ( '+' | '-' | '~' )* power
```

referenced by:

- term

power:

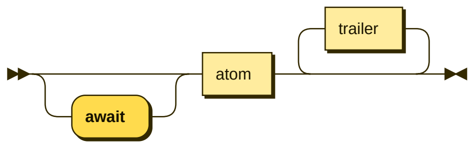


```
power    ::= atom_expr ['* factor]
```

referenced by:

- factor

atom_expr:

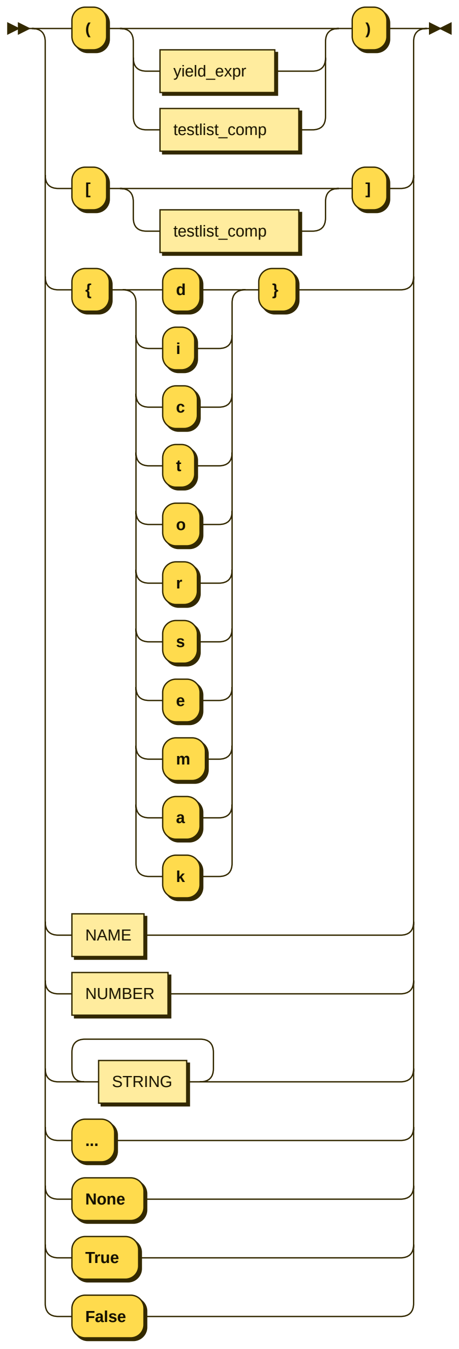


```
atom_expr
         ::= 'await'? atom trailer*
```

referenced by:

- power

atom:

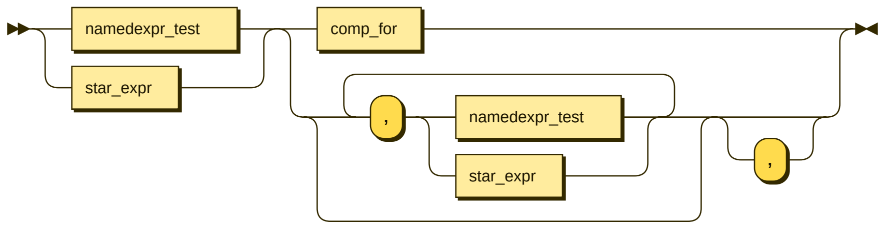


```
atom     ::= '(' ( yield_expr | testlist_comp )? ')'
           | '[' testlist_comp? ']'
           | '{' [dictorsemak] '}'
           | NAME
           | NUMBER
           | STRING+
           | '...'
           | 'None'
           | 'True'
           | 'False'
```

referenced by:

- atom_expr

testlist_comp:

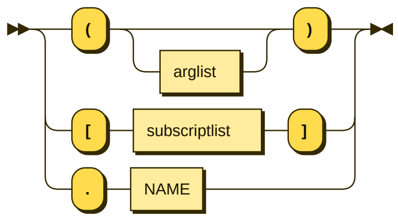


```
testlist_comp
         ::= ( namedexpr_test | star_expr ) ( comp_for | ( ',' ( namedexpr_test | star_expr ) )* ','? )
```

referenced by:

- atom

trailer:

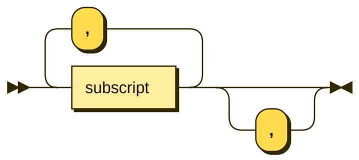


```
trailer  ::= '(' arglist? ')'
           | '[' subscriptlist ']'
           | '.' NAME
```

referenced by:

- atom_expr

subscriptlist:

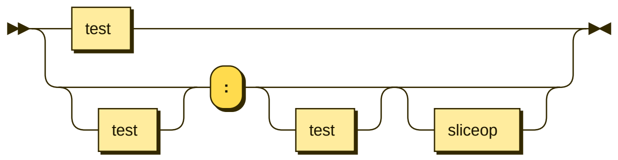


```
subscriptlist
         ::= subscript ( ',' subscript )* ','?
```

referenced by:

- trailer

subscript:

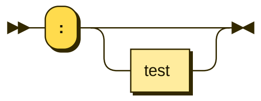


```
subscript
         ::= test
           | test? ':' test? sliceop?
```

referenced by:

- subscriptlist

sliceop:

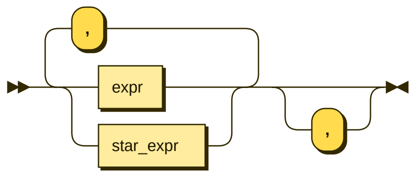


```
sliceop  ::= ':' test?
```

referenced by:

- subscript

exprlist:

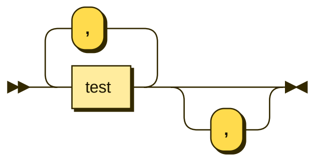


```
exprlist ::= ( expr | star_expr ) ( ',' ( expr | star_expr ) )* ','?
```

referenced by:

- del_stmt
- for_stmt
- sync_comp_for

testlist:

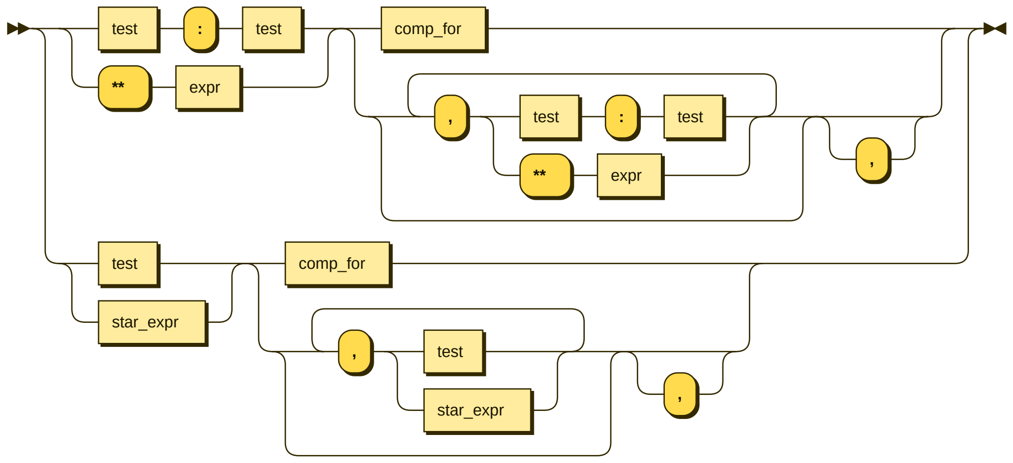


```
testlist ::= test ( ',' test )* ','?
```

referenced by:

- eval_input
- expr_stmt
- for_stmt

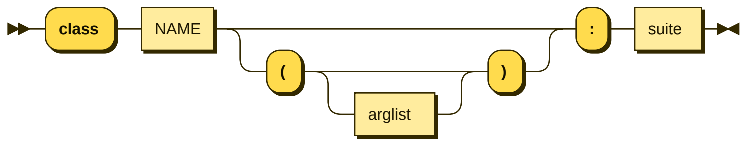
dictorsetmaker:



```
dictorsetmaker
         ::= ( test ':' test | '**' expr ) ( comp_for | ( ',' ( test ':' test | '**' expr ) )* ','? )
           | ( test | star_expr ) ( comp_for | ( ',' ( test | star_expr ) )* ','? )
```

no references

classdef:

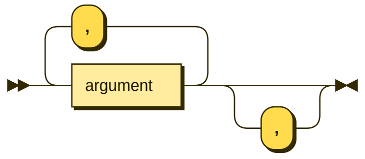


```
classdef ::= 'class' NAME ( '(' arglist? ')' )? ':' suite
```

referenced by:

- compound_stmt
- decorated

arglist:

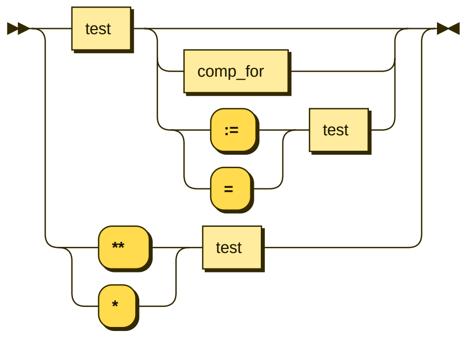


```
arglist  ::= argument ( ',' argument )* ','?
```

referenced by:

- classdef
- decorator
- trailer

argument:

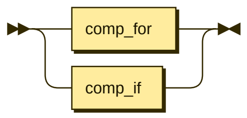


```
argument ::= test ( comp_for | ( ':=' | '=' ) test )?
           | ( '**' | '*' ) test
```

referenced by:

- arglist

comp_iter:

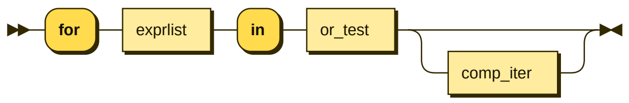


```
comp_iter
         ::= comp_for
           | comp_if
```

referenced by:

- comp_if
- sync_comp_for

sync_comp_for:

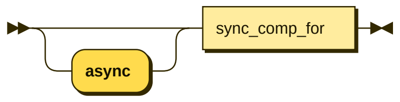


```
sync_comp_for
         ::= 'for' exprlist 'in' or_test comp_iter?
```

referenced by:

- comp_for

comp_for:



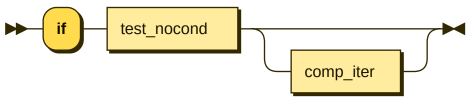
```
comp_for ::= 'async'? sync_comp_for
```

referenced by:

- argument

- comp_iter
- dictorsetmaker
- testlist_comp

comp_if:



```
comp_if  ::= 'if' test_nocond comp_iter?
```

referenced by:

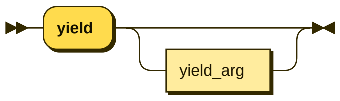
- comp_iter

encoding_decl:

```
encoding_decl
         ::= NAME
```

no references

yield_expr:

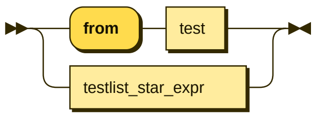


```
yield_expr
         ::= 'yield' yield_arg?
```

referenced by:

- annassign
- atom
- expr_stmt
- yield_stmt

yield_arg:

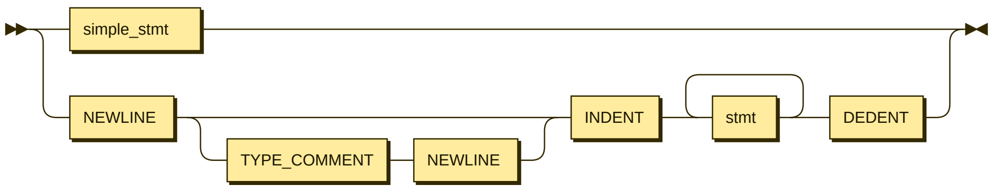


```
yield_arg
         ::= 'from' test
           | testlist_star_expr
```

referenced by:

- yield_expr

func_body_suite:

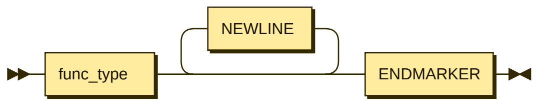


```
func_body_suite
         ::= simple_stmt
           | NEWLINE ( TYPE_COMMENT NEWLINE )? INDENT stmt+ DEDENT
```

referenced by:

- funcdef

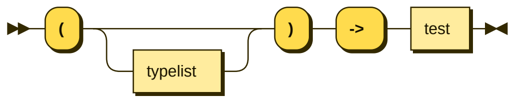
func_type_input:



```
func_type_input
         ::= func_type NEWLINE* ENDMARKER
```

no references

func_type:

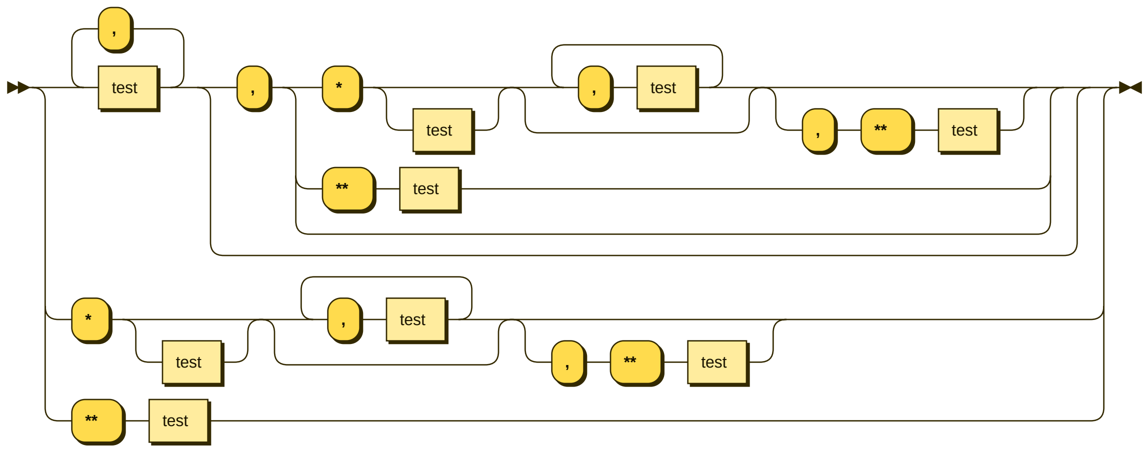


```
func_type
         ::= '(' typelist? ')' '->' test
```

referenced by:

- func_type_input

typelist:



```
typelist ::= test ( ',' test )* ( ',' ( '*' test? ( ',' test )* ( ',' '**' test )? | '**' test )? )?
           | '*' test? ( ',' test )* ( ',' '**' test )?
           | '**' test
```

referenced by:

- func_type

... generated by Railroad Diagram Generator