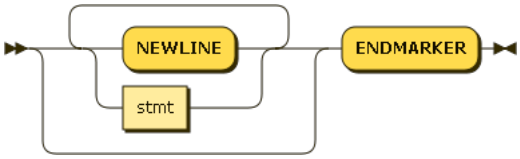
file_input:




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

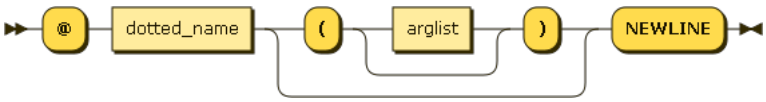
no references

eval_input:


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

no references

decorator:




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

referenced by:

- decorators

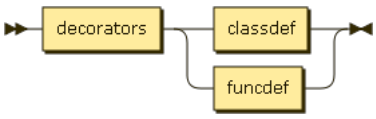
decorators:


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

referenced by:

- decorated

decorated:

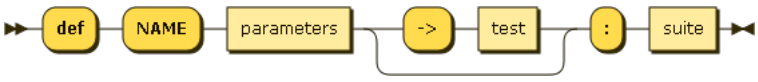



```
decorated
        ::= decorators ( classdef | funcdef )
```

referenced by:

- compound_stmt

funcdef:

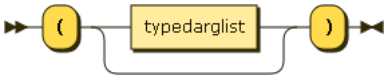


```
funcdef  ::= 'def' 'NAME' parameters ( '->' test )? ':' suite
```

referenced by:

- compound_stmt
- decorated

parameters:

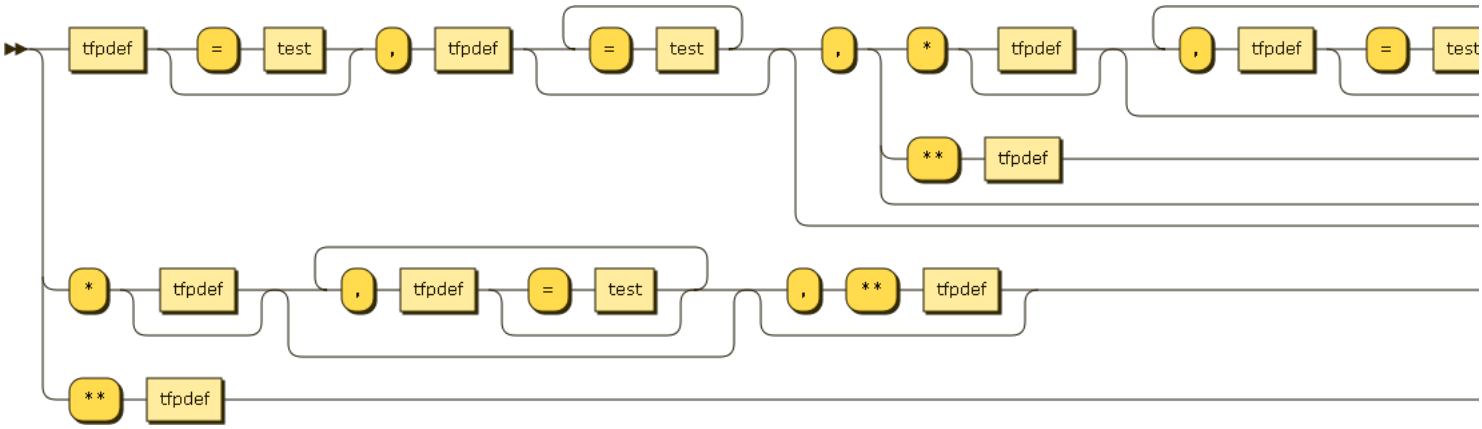



```
parameters
        ::= '(' typedarglist? ')'
```

referenced by:

- funcdef

typedarglist:

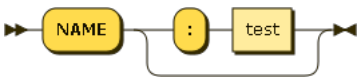



```
typedarglist
        ::= tfpdef ( '=' test )? ',' tfpdef ( '=' test )* ( ',' ( '*' tfpdef? ( ',' tfpdef ( '=' test )? )* ( ',' '**' tfpdef )? | '**' tfpdef )? )?
          | '*' tfpdef? ( ',' tfpdef ( '=' test )? )* ( ',' '**' tfpdef )?
          | '**' tfpdef
```

referenced by:

- parameters

tfpdef:

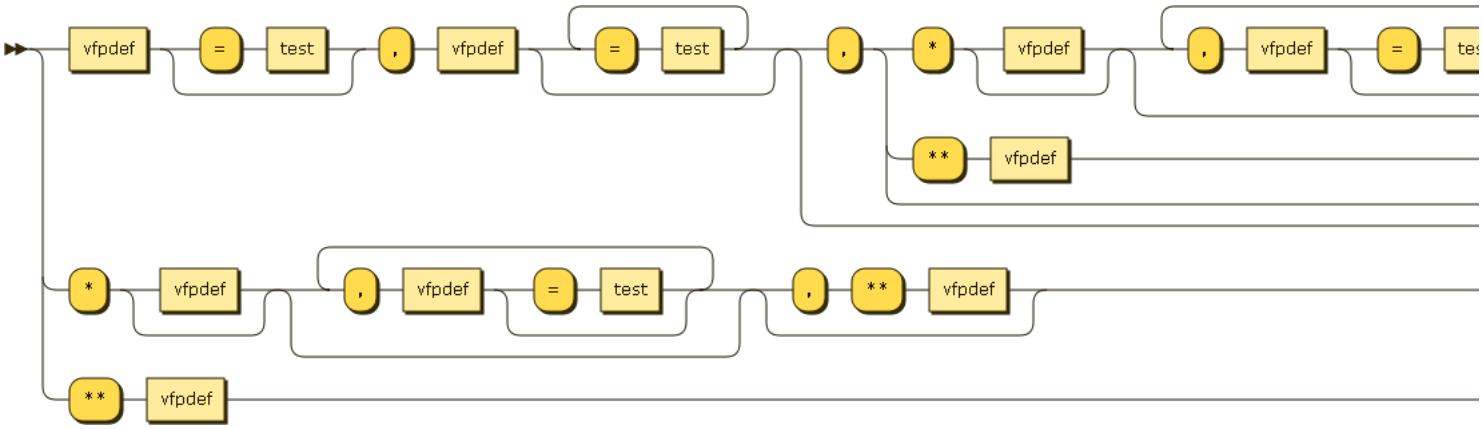



```
tfpdef   ::= 'NAME' ( ':' test )?
```

referenced by:

- typedarglist

varargslist:




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

referenced by:

- lambdef
- lambdef_nocond

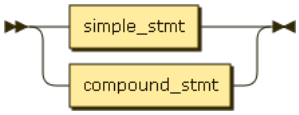
vfpdef:


```
vfpdef   ::= 'NAME'
```

referenced by:

- varargslist

stmt:

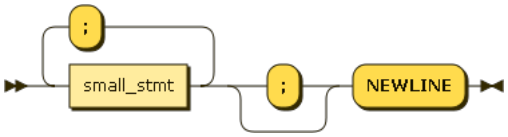



```
stmt     ::= simple_stmt
           | compound_stmt
```

referenced by:

- file_input
- suite

simple_stmt:

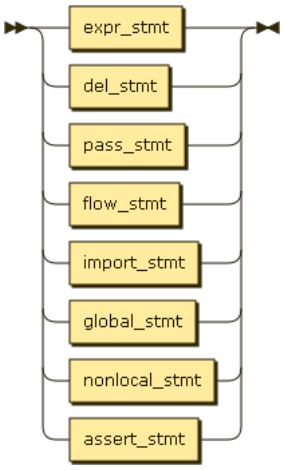



```
simple_stmt
        ::= small_stmt ( ';' small_stmt )* ';'? 'NEWLINE'
```

referenced by:

- stmt
- suite

small_stmt:

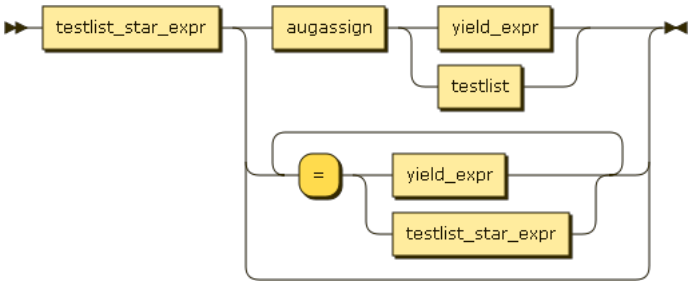



```
small_stmt
          ::= expr_stmt
            | del_stmt
            | pass_stmt
            | flow_stmt
            | import_stmt
            | global_stmt
            | nonlocal_stmt
            | assert_stmt
```

referenced by:

- simple_stmt

expr_stmt:

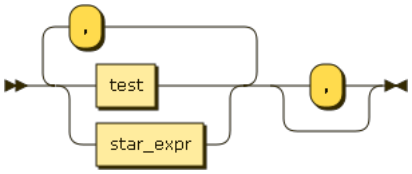



```
expr_stmt
          ::= testlist_star_expr ( augassign ( yield_expr | testlist ) | ( '=' ( yield_expr | testlist_star_expr ) )* )
```

referenced by:

- small_stmt

testlist_star_expr:

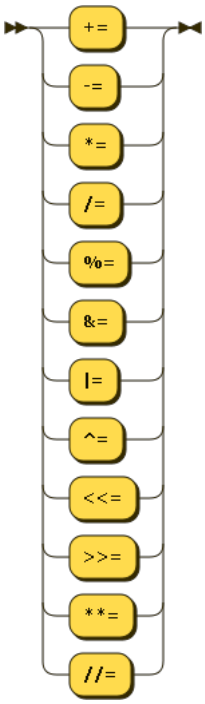



```
testlist_star_expr
          ::= ( test | star_expr ) ( ',' ( test | star_expr ) )* ','?
```

referenced by:

- expr_stmt

augassign:

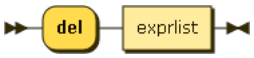



```
augassign
          ::= '+='
            | '-='
            | '*='
            | '/='
            | '%='
            | '&='
            | '|='
            | '^='
            | '<<='
            | '>>='
            | '**='
            | '//='
```

referenced by:

- expr_stmt

del_stmt:




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

referenced by:

- small_stmt

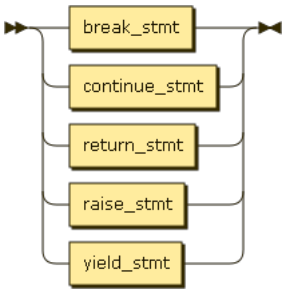
pass_stmt:


```
pass_stmt
         ::= 'pass'
```

referenced by:

- small_stmt

flow_stmt:




```
flow_stmt
         ::= break_stmt
           | continue_stmt
           | return_stmt
           | raise_stmt
           | yield_stmt
```

referenced by:

- small_stmt

break_stmt:


```
break_stmt
         ::= 'break'
```

referenced by:

- flow_stmt

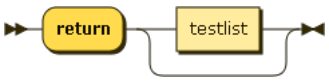
continue_stmt:


```
continue_stmt
         ::= 'continue'
```

referenced by:

- flow_stmt

return_stmt:




```
return_stmt
         ::= 'return' testlist?
```

referenced by:

- flow_stmt

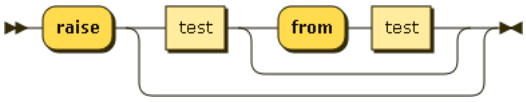
yield_stmt:


```
yield_stmt
         ::= yield_expr
```

referenced by:

- flow_stmt

raise_stmt:

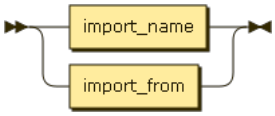



```
raise_stmt
         ::= 'raise' ( test ( 'from' test )? )?
```

referenced by:

- flow_stmt

import_stmt:




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

referenced by:

- small_stmt

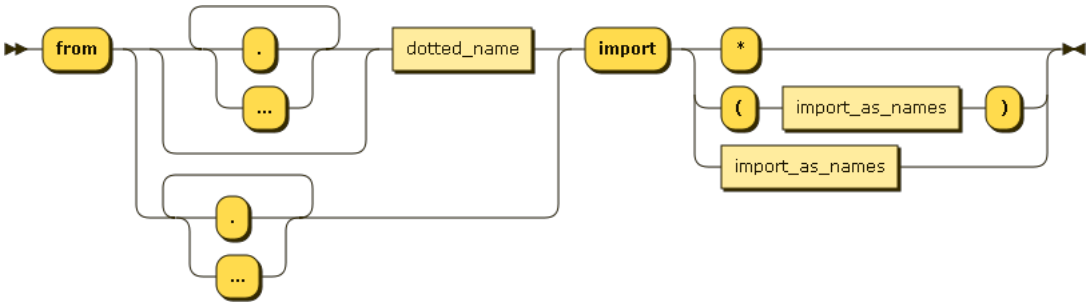
import_name:


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

referenced by:

- import_stmt

import_from:

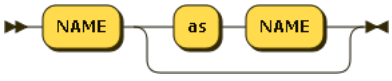



```
import_from
         ::= 'from' ( ( '.' | '...' )* dotted_name | ( '.' | '...' )+ ) 'import' ( '*' | '(' import_as_names ')' | import_as_names )
```

referenced by:

- import_stmt

import_as_name:

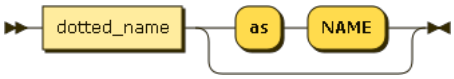



```
import_as_name
         ::= 'NAME' ( 'as' 'NAME' )?
```

referenced by:

- import_as_names

dotted_as_name:

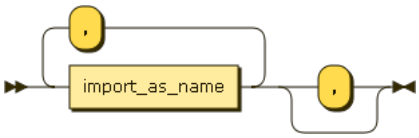



```
dotted_as_name
        ::= dotted_name ( 'as' 'NAME' )?
```

referenced by:

- dotted_as_names

import_as_names:

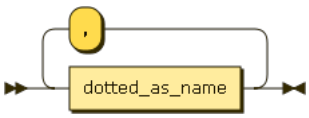



```
import_as_names
        ::= import_as_name ( ',' import_as_name )* ','?
```

referenced by:

- import_from

dotted_as_names:

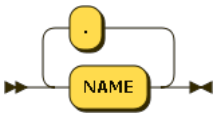



```
dotted_as_names
        ::= dotted_as_name ( ',' dotted_as_name )*
```

referenced by:

- import_name

dotted_name:

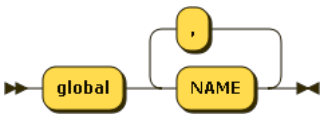



```
dotted_name
        ::= 'NAME' ( '.' 'NAME' )*
```

referenced by:

- decorator
- dotted_as_name
- import_from

global_stmt:

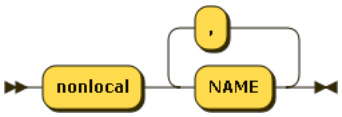



```
global_stmt
        ::= 'global' 'NAME' ( ',' 'NAME' )*
```

referenced by:

- small_stmt

nonlocal_stmt:

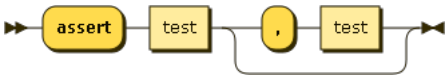



```
nonlocal_stmt
        ::= 'nonlocal' 'NAME' ( ',' 'NAME' )*
```

referenced by:

- small_stmt

assert_stmt:

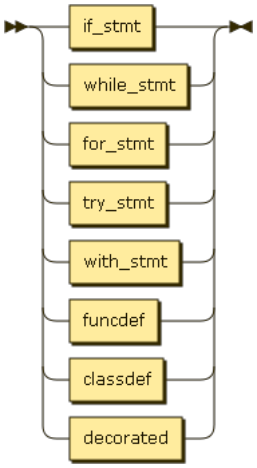



```
assert_stmt
        ::= 'assert' test ( ',' test )?
```

referenced by:

- small_stmt

compound_stmt:

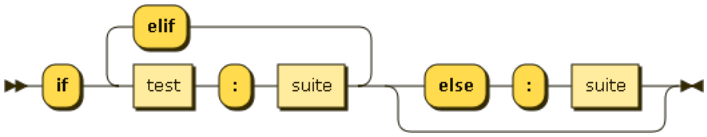



```
compound_stmt
        ::= if_stmt
          | while_stmt
          | for_stmt
          | try_stmt
          | with_stmt
          | funcdef
          | classdef
          | decorated
```

referenced by:

- stmt

if_stmt:

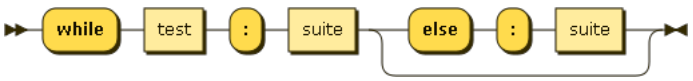



```
if_stmt  ::= 'if' test ':' suite ( 'elif' test ':' suite )* ( 'else' ':' suite )?
```

referenced by:

- compound_stmt

while_stmt:

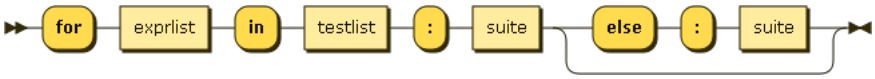



```
while_stmt
        ::= 'while' test ':' suite ( 'else' ':' suite )?
```

referenced by:

- compound_stmt

for_stmt:

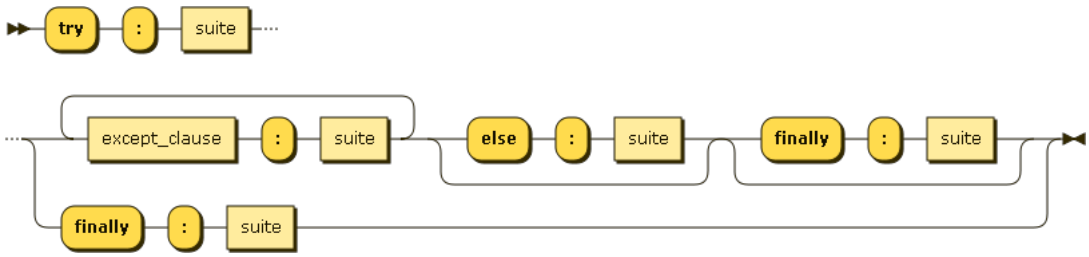



```
for_stmt ::= 'for' exprlist 'in' testlist ':' suite ( 'else' ':' suite )?
```

referenced by:

- compound_stmt

try_stmt:

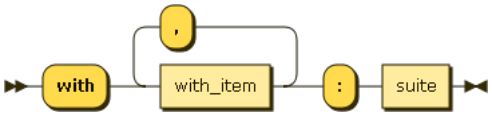



```
try_stmt ::= 'try' ':' suite ( ( except_clause ':' suite )+ ( 'else' ':' suite )? ( 'finally' ':' suite )? | 'finally' ':' suite )
```

referenced by:

- compound_stmt

with_stmt:

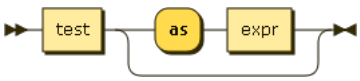



```
with_stmt
        ::= 'with' with_item ( ',' with_item )* ':' suite
```

referenced by:

- compound_stmt

with_item:

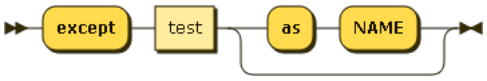



```
with_item
        ::= test ( 'as' expr )?
```

referenced by:

- with_stmt

except_clause:

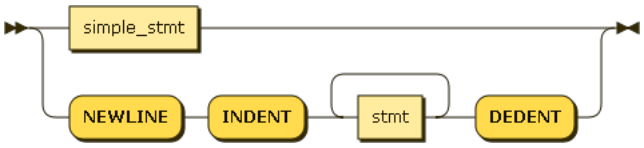



```
except_clause
        ::= 'except' test ( 'as' 'NAME' )?
```

referenced by:

- try_stmt

suite:

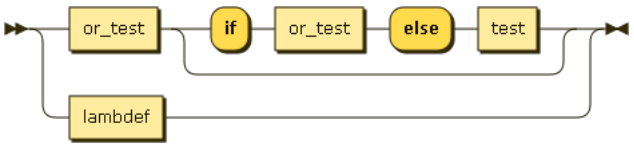



```
suite    ::= simple_stmt
           | 'NEWLINE' 'INDENT' stmt+ 'DEDENT'
```

referenced by:

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

test:

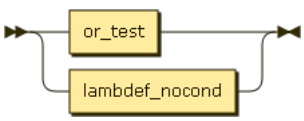



```
test     ::= or_test ( 'if' or_test 'else' test )?
           | lambdef
```

referenced by:

- arglist
- argument
- assert_stmt
- dictorsetmaker
- except_clause
- funcdef
- if_stmt
- lambdef
- raise_stmt
- sliceop
- subscript
- test
- testlist
- testlist_comp
- testlist_star_expr
- tfpdef
- typedarglist
- varargslist
- while_stmt
- with_item
- yield_arg

test_nocond:




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

referenced by:

- comp_if
- lambdef_nocond

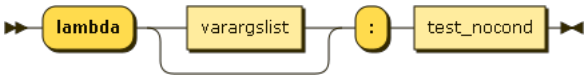
lambdef:


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

referenced by:

- test

lambdef_nocond:

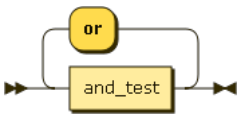



```
lambdef_nocond
         ::= 'lambda' varargslist? ':' test_nocond
```

referenced by:

- test_nocond

or_test:

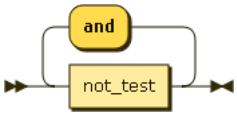



```
or_test  ::= and_test ( 'or' and_test )*
```

referenced by:

- comp_for
- test
- test_nocond

and_test:

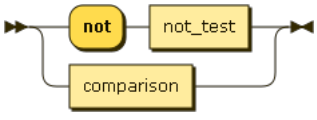



```
and_test ::= not_test ( 'and' not_test )*
```

referenced by:

- or_test

not_test:

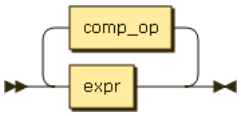



```
not_test ::= 'not' not_test
           | comparison
```

referenced by:

- and_test
- not_test

comparison:

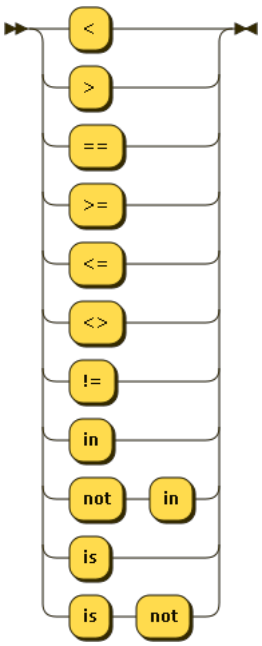



```
comparison
        ::= expr ( comp_op expr )*
```

referenced by:

- not_test

comp_op:

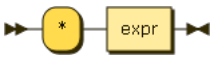



```
comp_op  ::= '<'
           | '>'
           | '=='
           | '>='
           | '<='
           | '<>'
           | '!='
           | 'in'
           | 'not' 'in'
           | 'is'
           | 'is' 'not'
```

referenced by:

- comparison

star_expr:

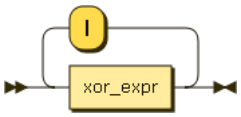



```
star_expr
        ::= '*' expr
```

referenced by:

- exprlist
- testlist_comp
- testlist_star_expr

expr:

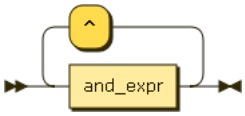



```
expr     ::= xor_expr ( '|' xor_expr )*
```

referenced by:

- comparison
- exprlist
- star_expr
- with_item

xor_expr:

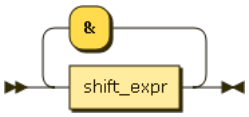



```
xor_expr ::= and_expr ( '^' and_expr )*
```

referenced by:

- expr

and_expr:

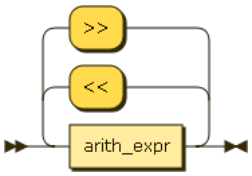



```
and_expr ::= shift_expr ( '&' shift_expr )*
```

referenced by:

- xor_expr

shift_expr:

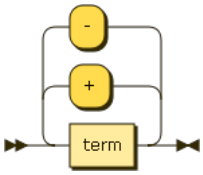



```
shift_expr
         ::= arith_expr ( ( '<<' | '>>' ) arith_expr )*
```

referenced by:

- and_expr

arith_expr:

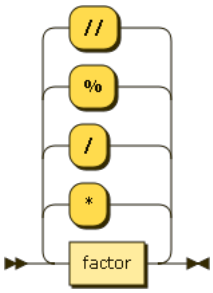



```
arith_expr
         ::= term ( ( '+' | '-' ) term )*
```

referenced by:

- shift_expr

term:

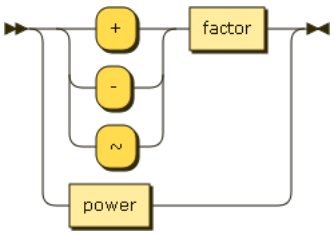



```
term     ::= factor ( ( '*' | '/' | '%' | '//' ) factor )*
```

referenced by:

- arith_expr

factor:

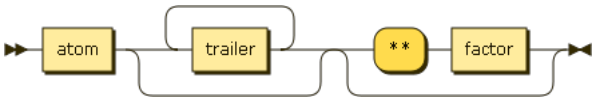



```
factor   ::= ( '+' | '-' | '~' ) factor
           | power
```

referenced by:

- factor
- power
- term

power:

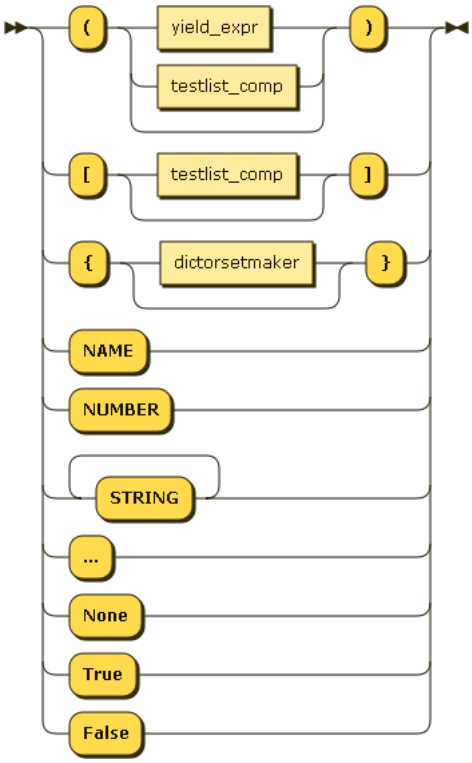



```
power    ::= atom trailer* ( '**' factor )?
```

referenced by:

- factor

atom:

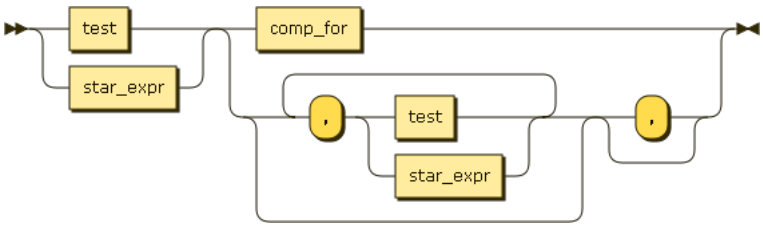



```
atom     ::= '(' ( yield_expr | testlist_comp )? ')'
           | '[' testlist_comp? ']'
           | '{' dictorsetmaker? '}'
           | 'NAME'
           | 'NUMBER'
           | 'STRING'+
           | '...'
           | 'None'
           | 'True'
           | 'False'
```

referenced by:

- power

testlist_comp:

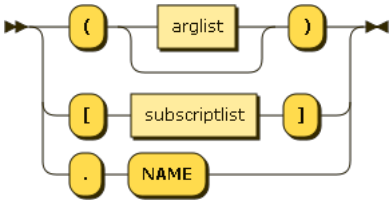



```
testlist_comp
         ::= ( test | star_expr ) ( comp_for | ( ',' ( test | star_expr ) )* ','? )
```

referenced by:

- atom

trailer:



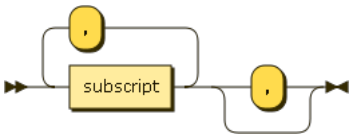

```
trailer  ::= '(' arglist? ')'
```

```
            | '[' subscriptlist ']'
            | '.' 'NAME'
```

referenced by:

- power

subscriptlist:

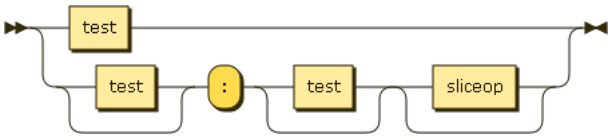



```
subscriptlist
        ::= subscript ( ',' subscript )* ','?
```

referenced by:

- trailer

subscript:

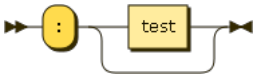



```
subscript
        ::= test
          | test? ':' test? sliceop?
```

referenced by:

- subscriptlist

sliceop:

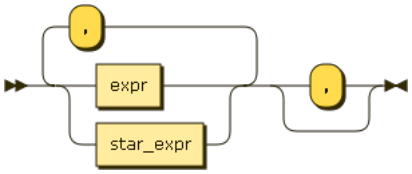



```
sliceop  ::= ':' test?
```

referenced by:

- subscript

exprlist:

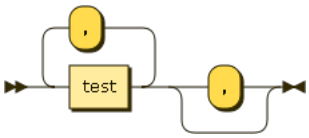



```
exprlist ::= ( expr | star_expr ) ( ',' ( expr | star_expr ) )* ','?
```

referenced by:

- comp_for
- del_stmt
- for_stmt

testlist:

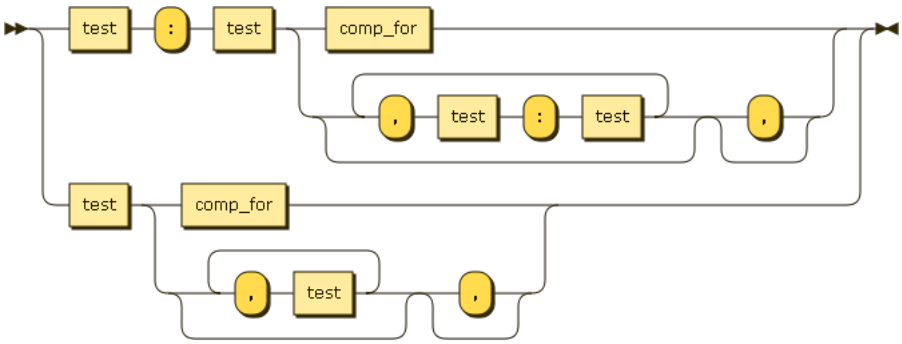



```
testlist ::= test ( ',' test )* ','?
```

referenced by:

- eval_input
- expr_stmt
- for_stmt
- return_stmt
- yield_arg

dictorsetmaker:

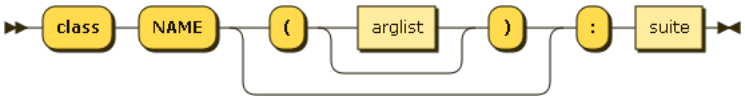



```
dictorsetmaker
        ::= test ':' test ( comp_for | ( ',' test ':' test )* ','? )
          | test ( comp_for | ( ',' test )* ','? )
```

referenced by:

- atom

classdef:

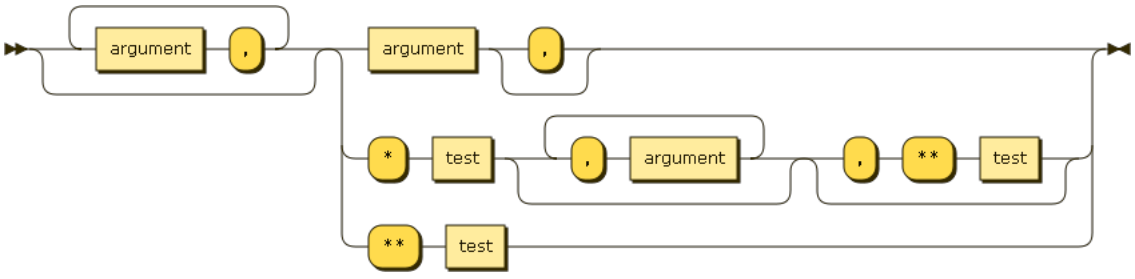



```
classdef ::= 'class' 'NAME' ( '(' arglist? ')' )? ':' suite
```

referenced by:

- compound_stmt
- decorated

arglist:

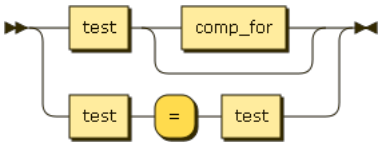



```
arglist  ::= ( argument ',' )* ( argument ','? | '*' test ( ',' argument )* ( ',' '**' test )? | '**' test )
```

referenced by:

- classdef
- decorator
- trailer

argument:

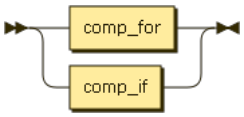



```
argument ::= test comp_for?
           | test '=' test
```

referenced by:

- arglist

comp_iter:

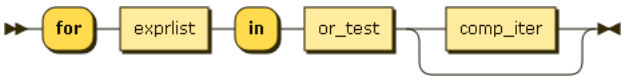



```
comp_iter
          ::= comp_for
            | comp_if
```

referenced by:

- comp_for
- comp_if

comp_for:

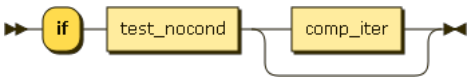



```
comp_for ::= 'for' exprlist 'in' or_test comp_iter?
```

referenced by:

- argument
- comp_iter
- dictorsetmaker
- testlist_comp

comp_if:




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

referenced by:

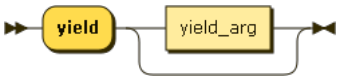
- comp_iter

encoding_decl:


```
encoding_decl
          ::= 'NAME'
```

no references

yield_expr:

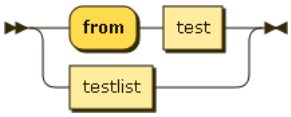



```
yield_expr
        ::= 'yield' yield_arg?
```

referenced by:

- atom
- expr_stmt
- yield_stmt

yield_arg:




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

referenced by:

- yield_expr

CSC330
Group
A