Compound statements

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst`, line 7)

Unknown directive type "index".

```
.. index:: pair: compound; statement
```

Compound statements contain (groups of) other statements; they affect or control the execution of those other statements in some way. In general, compound statements span multiple lines, although in simple incarnations a whole compound statement may be contained in one line.

The :keyword:`if`, :keyword:`while` and :keyword:`for` statements implement traditional control flow constructs. :keyword:`try` specifies exception handlers and/or cleanup code for a group of statements, while the :keyword:`with` statement allows the execution of initialization and finalization code around a block of code. Function and class definitions are also syntactically compound statements.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst`, line 14); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst`, line 14); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst`, line 14); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst`, line 14); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst`, line 14); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst`, line 21)

Unknown directive type "index".

```
.. index::
   single: clause
   single: suite
   single: ; (semicolon)
```

A compound statement consists of one or more 'clauses.' A clause consists of a header and a 'suite.' The clause headers of a particular compound statement are all at the same indentation level. Each clause header begins with a uniquely identifying keyword and ends with a colon. A suite is a group of statements controlled by a clause. A suite can be one or more semicolon-separated simple statements on the same line as the header, following the header's colon, or it can be one or more indented statements on subsequent lines. Only the latter form of a suite can contain nested compound statements; the following is illegal, mostly because it wouldn't be clear to which :keyword:`if` clause a following :keyword:`else` clause would belong:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst`, line 26); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 26); *backlink*

Unknown interpreted text role "keyword".

```
if test1: if test2: print(x)
```

Also note that the semicolon binds tighter than the colon in this context, so that in the following example, either all or none of the :func:`print` calls are executed:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 39); *backlink*

Unknown interpreted text role "func".

```
if x < y < z: print(x); print(y); print(z)
```

Summarizing:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 48)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   compound_stmt: `if_stmt`
                : | `while_stmt`
                : | `for_stmt`
                : | `try_stmt`
                : | `with_stmt`
                : | `match_stmt`
                : | `funcdef`
                : | `classdef`
                : | `async_with_stmt`
                : | `async_for_stmt`
                : | `async_funcdef`
   suite: `stmt_list` NEWLINE | NEWLINE INDENT `statement`+ DEDENT
   statement: `stmt_list` NEWLINE | `compound_stmt`
   stmt_list: `simple_stmt` (";" `simple_stmt`)* [";"]
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 64)

Unknown directive type "index".

```
.. index::
   single: NEWLINE token
   single: DEDENT token
   pair: dangling; else
```

Note that statements always end in a `NEWLINE` possibly followed by a `DEDENT`. Also note that optional continuation clauses always begin with a keyword that cannot start a statement, thus there are no ambiguities (the 'dangling :keyword:`else`' problem is solved in Python by requiring nested :keyword:`if` statements to be indented).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 69); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 69); *backlink*

Unknown interpreted text role "keyword".

The formatting of the grammar rules in the following sections places each clause on a separate line for clarity.

The :keyword:`!if` statement

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 83); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 86)

Unknown directive type "index".

```
.. index::
   ! statement: if
   keyword: elif
   keyword: else
   single: : (colon); compound statement
```

The :keyword:`if` statement is used for conditional execution:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 92); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 94)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   if_stmt: "if" `assignment_expression` ":" `suite`
          : ("elif" `assignment_expression` ":" `suite`)*
          : ["else" ":" `suite`]
```

It selects exactly one of the suites by evaluating the expressions one by one until one is found to be true (see section :ref:`booleans` for the definition of true and false); then that suite is executed (and no other part of the :keyword:`if` statement is executed or evaluated). If all expressions are false, the suite of the :keyword:`else` clause, if present, is executed.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 99); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 99); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 99); *backlink*

Unknown interpreted text role "keyword".

The :keyword:`!while` statement

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 108); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 111)

Unknown directive type "index".

```
.. index::
   ! statement: while
   keyword: else
   pair: loop; statement
   single: : (colon); compound statement
```

The :keyword:`while` statement is used for repeated execution as long as an expression is true:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 117); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 120)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   while_stmt: "while" `assignment_expression` ":" `suite`
             : ["else" ":" `suite`]
```

This repeatedly tests the expression and, if it is true, executes the first suite; if the expression is false (which may be the first time it is tested) the suite of the :keyword:`!else` clause, if present, is executed and the loop terminates.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 124); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 129)

Unknown directive type "index".

```
.. index::
   statement: break
   statement: continue
```

A :keyword:`break` statement executed in the first suite terminates the loop without executing the :keyword:`!else` clause's suite. A :keyword:`continue` statement executed in the first suite skips the rest of the suite and goes back to testing the expression.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 133); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 133); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 133); *backlink*

Unknown interpreted text role "keyword".

The :keyword:`!for` statement

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 141); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 144)

Unknown directive type "index".

```
.. index::
   ! statement: for
   keyword: in
   keyword: else
   pair: target; list
   pair: loop; statement
   object: sequence
   single: : (colon); compound statement
```

The :keyword:`for` statement is used to iterate over the elements of a sequence (such as a string, tuple or list) or other iterable object:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 153); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 156)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   for_stmt: "for" `target_list` "in" `starred_list` ":" `suite`
           : ["else" ":" `suite`]
```

The `starred_list` expression is evaluated once; it should yield an :term:`iterable` object. An :term:`iterator` is created for that iterable. The first item provided by the iterator is then assigned to the target list using the standard rules for assignments (see :ref:`assignment`), and the suite is executed. This repeats for each item provided by the iterator. When the iterator is exhausted, the suite in the :keyword:`!else` clause, if present, is executed, and the loop terminates.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 160); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 160); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 160); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 160); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 169)

Unknown directive type "index".

```
.. index::
   statement: break
   statement: continue
```

A :keyword:`break` statement executed in the first suite terminates the loop without executing the :keyword:`!else` clause's suite. A :keyword:`continue` statement executed in the first suite skips the rest of the suite and continues with the next item, or with the :keyword:`!else` clause if there is no next item.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 173); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 173); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 173); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 173); *backlink*

Unknown interpreted text role "keyword".

The for-loop makes assignments to the variables in the target list. This overwrites all previous assignments to those variables including those made in the suite of the for-loop:

```
for i in range(10):
    print(i)
    i = 5             # this will not affect the for-loop
                      # because i will be overwritten with the next
                      # index in the range
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 190)

Unknown directive type "index".

```
.. index::
   builtin: range
```

Names in the target list are not deleted when the loop is finished, but if the sequence is empty, they will not have been assigned to at all by the loop. Hint: the built-in function :func:`range` returns an iterator of integers suitable to emulate the effect of Pascal's `for i := a to b do`; e.g., `list(range(3))` returns the list `[0, 1, 2]`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 193); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 199)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.11
   Starred elements are now allowed in the expression list.
```

The :keyword:`!try` statement

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 207); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 210)

Unknown directive type "index".

```
.. index::
   ! statement: try
   keyword: except
   keyword: finally
   keyword: else
   keyword: as
   single: : (colon); compound statement
```

The :keyword:`try` statement specifies exception handlers and/or cleanup code for a group of statements:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 218); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 221)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   try_stmt: `try1_stmt` | `try2_stmt` | `try3_stmt`
   try1_stmt: "try" ":" `suite`
            : ("except" [`expression` ["as" `identifier`]] ":" `suite`)+
            : ["else" ":" `suite`]
            : ["finally" ":" `suite`]
   try2_stmt: "try" ":" `suite`
            : ("except" "*" `expression` ["as" `identifier`] ":" `suite`)+
            : ["else" ":" `suite`]
            : ["finally" ":" `suite`]
   try3_stmt: "try" ":" `suite`
            : "finally" ":" `suite`
```

The :keyword:`except` clause(s) specify one or more exception handlers. When no exception occurs in the :keyword:`try` clause, no exception handler is executed. When an exception occurs in the :keyword:`!try` suite, a search for an exception handler is started. This search inspects the except clauses in turn until one is found that matches the exception. An expression-less except clause, if present, must be last; it matches any exception. For an except clause with an expression, that expression is evaluated, and the clause matches the exception if the resulting object is "compatible" with the exception. An object is compatible with an exception if the object is the class or a :term:`non-virtual base class <abstract base class>` of the exception object, or a tuple containing an item that is the class or a non-virtual base class of the exception object.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 235); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 235); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 235); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 235); *backlink*

Unknown interpreted text role "term".

If no except clause matches the exception, the search for an exception handler continues in the surrounding code and on the invocation stack. [1]

If the evaluation of an expression in the header of an except clause raises an exception, the original search for a handler is canceled and a search starts for the new exception in the surrounding code and on the call stack (it is treated as if the entire :keyword:`try` statement raised the exception).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 251); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 256)

Unknown directive type "index".

```
.. index:: single: as; except clause
```

When a matching except clause is found, the exception is assigned to the target specified after the :keyword:`!as` keyword in that except clause, if present, and the except clause's suite is executed. All except clauses must have an executable block. When the end of this block is reached, execution continues normally after the entire try statement. (This means that if two nested handlers exist for

the same exception, and the exception occurs in the try clause of the inner handler, the outer handler will not handle the exception.)

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 258); *backlink*

Unknown interpreted text role "keyword".

When an exception has been assigned using `as target`, it is cleared at the end of the except clause. This is as if

```
except E as N:
    foo
```

was translated to

```
except E as N:
    try:
        foo
    finally:
        del N
```

This means the exception must be assigned to a different name to be able to refer to it after the except clause. Exceptions are cleared because with the traceback attached to them, they form a reference cycle with the stack frame, keeping all locals in that frame alive until the next garbage collection occurs.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 285)

Unknown directive type "index".

```
.. index::
   module: sys
   object: traceback
```

Before an except clause's suite is executed, details about the exception are stored in the :mod:`sys` module and can be accessed via :func:`sys.exc_info`. :func:`sys.exc_info` returns a 3-tuple consisting of the exception class, the exception instance and a traceback object (see section :ref:`types`) identifying the point in the program where the exception occurred. The details about the exception accessed via :func:`sys.exc_info` are restored to their previous values when leaving an exception handler:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 289); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 289); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 289); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 289); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 289); *backlink*

Unknown interpreted text role "func".

```
>>> print(sys.exc_info())
(None, None, None)
>>> try:
...     raise TypeError
... except:
...     print(sys.exc_info())
...     try:
...         raise ValueError
...     except:
```

```
...         print(sys.exc_info())
...     print(sys.exc_info())
...
(<class 'TypeError'>, TypeError(), <traceback object at 0x10efad080>)
(<class 'ValueError'>, ValueError(), <traceback object at 0x10efad040>)
(<class 'TypeError'>, TypeError(), <traceback object at 0x10efad080>)
>>> print(sys.exc_info())
(None, None, None)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 315)

Unknown directive type "index".

```
.. index::
   keyword: except_star
```

The :keyword:`except*<except_star>` clause(s) are used for handling :exc:`ExceptionGroup`s. The exception type for matching is interpreted as in the case of :keyword:`except`, but in the case of exception groups we can have partial matches when the type matches some of the exceptions in the group. This means that multiple except* clauses can execute, each handling part of the exception group. Each clause executes once and handles an exception group of all matching exceptions. Each exception in the group is handled by at most one except* clause, the first that matches it.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 318); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 318); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 318); *backlink*

Unknown interpreted text role "keyword".

```
>>> try:
...     raise ExceptionGroup("eg",
...         [ValueError(1), TypeError(2), OSError(3), OSError(4)])
... except* TypeError as e:
...     print(f'caught {type(e)} with nested {e.exceptions}')
... except* OSError as e:
...     print(f'caught {type(e)} with nested {e.exceptions}')
...
caught <class 'ExceptionGroup'> with nested (TypeError(2),)
caught <class 'ExceptionGroup'> with nested (OSError(3), OSError(4))
  + Exception Group Traceback (most recent call last):
  |   File "<stdin>", line 2, in <module>
  | ExceptionGroup: eg
  +-+---------------- 1 ----------------
    | ValueError: 1
    +------------------------------------
>>>

Any remaining exceptions that were not handled by any except* clause
are re-raised at the end, combined into an exception group along with
all exceptions that were raised from within except* clauses.

An except* clause must have a matching type, and this type cannot be a
subclass of :exc:`BaseExceptionGroup`. It is not possible to mix except
and except* in the same :keyword:`try`. :keyword:`break`,
:keyword:`continue` and :keyword:`return` cannot appear in an except*
clause.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 356)

Unknown directive type "index".

```
.. index::
   keyword: else
   statement: return
   statement: break
   statement: continue
```

The optional :keyword:`!else` clause is executed if the control flow leaves the :keyword:`try` suite, no exception was raised, and no :keyword:`return`, :keyword:`continue`, or :keyword:`break` statement was executed. Exceptions in the :keyword:`!else` clause are not handled by the preceding :keyword:`except` clauses.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst`, **line 362**); ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst`, **line 362**); ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst`, **line 362**); ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst`, **line 362**); ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst`, **line 362**); ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst`, **line 362**); ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst`, **line 362**); ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst`, **line 368**)

Unknown directive type "index".

```
.. index:: keyword: finally
```

If :keyword:`finally` is present, it specifies a 'cleanup' handler. The :keyword:`try` clause is executed, including any :keyword:`except` and :keyword:`!else` clauses. If an exception occurs in any of the clauses and is not handled, the exception is temporarily saved. The :keyword:`!finally` clause is executed. If there is a saved exception it is re-raised at the end of the :keyword:`!finally` clause. If the :keyword:`!finally` clause raises another exception, the saved exception is set as the context of the new exception. If the :keyword:`!finally` clause executes a :keyword:`return`, :keyword:`break` or :keyword:`continue` statement, the saved exception is discarded:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst`, **line 370**); ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst`, **line 370**); ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-`

main\Doc\reference\[cpython-main] [Doc] [reference]compound_stmts.rst, line 370); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main] [Doc] [reference]compound_stmts.rst, line 370); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main] [Doc] [reference]compound_stmts.rst, line 370); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main] [Doc] [reference]compound_stmts.rst, line 370); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main] [Doc] [reference]compound_stmts.rst, line 370); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main] [Doc] [reference]compound_stmts.rst, line 370); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main] [Doc] [reference]compound_stmts.rst, line 370); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main] [Doc] [reference]compound_stmts.rst, line 370); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main] [Doc] [reference]compound_stmts.rst, line 370); *backlink*

Unknown interpreted text role "keyword".

```
>>> def f():
...     try:
...         1/0
...     finally:
...         return 42
...
>>> f()
42
```

The exception information is not available to the program during execution of the :keyword:`finally` clause.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main] [Doc] [reference]compound_stmts.rst, line 389); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main] [Doc] [reference]compound_stmts.rst, line 392)

Unknown directive type "index".

```
.. index::
   statement: return
   statement: break
   statement: continue
```

When a :keyword:`return`, :keyword:`break` or :keyword:`continue` statement is executed in the :keyword:`try` suite of a :keyword:`!try`...:keyword:`!finally` statement, the :keyword:`finally` clause is also executed 'on the way out.'

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 397); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 397); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 397); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 397); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 397); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 397); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 397); *backlink*

Unknown interpreted text role "keyword".

The return value of a function is determined by the last :keyword:`return` statement executed. Since the :keyword:`finally` clause always executes, a :keyword:`!return` statement executed in the :keyword:`!finally` clause will always be the last one executed:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 401); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 401); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 401); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 401); *backlink*

Unknown interpreted text role "keyword".

```
>>> def foo():
...     try:
...         return 'try'
...     finally:
...         return 'finally'
...
>>> foo()
```

```
'finally'
```

Additional information on exceptions can be found in section :ref:`exceptions`, and information on using the :keyword:`raise` statement to generate exceptions may be found in section :ref:`raise`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 415); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 415); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 415); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 419)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.8
   Prior to Python 3.8, a :keyword:`continue` statement was illegal in the
   :keyword:`finally` clause due to a problem with the implementation.
```

The :keyword:`!with` statement

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 427); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 430)

Unknown directive type "index".

```
.. index::
   ! statement: with
   keyword: as
   single: as; with statement
   single: , (comma); with statement
   single: : (colon); compound statement
```

The :keyword:`with` statement is used to wrap the execution of a block with methods defined by a context manager (see section :ref:`context-managers`). This allows common :keyword:`try`...:keyword:`except`...:keyword:`finally` usage patterns to be encapsulated for convenient reuse.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 437); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 437); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 437); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 437); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 437); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 442)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   with_stmt: "with" ( "(" `with_stmt_contents` ","? ")" | `with_stmt_contents` ) ":" `suite`
   with_stmt_contents: `with_item` ("," `with_item`)*
   with_item: `expression` ["as" `target`]
```

The execution of the :keyword:`with` statement with one "item" proceeds as follows:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 447); *backlink*

Unknown interpreted text role "keyword".

1. The context expression (the expression given in the :token:`~python-grammar:with_item`) is evaluated to obtain a context manager.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 449); *backlink*

   Unknown interpreted text role "token".

2. The context manager's :meth:`__enter__` is loaded for later use.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 452); *backlink*

   Unknown interpreted text role "meth".

3. The context manager's :meth:`__exit__` is loaded for later use.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 454); *backlink*

   Unknown interpreted text role "meth".

4. The context manager's :meth:`__enter__` method is invoked.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 456); *backlink*

   Unknown interpreted text role "meth".

5. If a target was included in the :keyword:`with` statement, the return value from :meth:`__enter__` is assigned to it.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 458); *backlink*

   Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 458); *backlink*

Unknown interpreted text role "meth".

Note

The :keyword:`with` statement guarantees that if the :meth:`__enter__` method returns without an error, then :meth:`__exit__` will always be called. Thus, if an error occurs during the assignment to the target list, it will be treated the same as an error occurring within the suite would be. See step 6 below.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 463); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 463); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 463); *backlink*

Unknown interpreted text role "meth".

6. The suite is executed.
7. The context manager's :meth:`__exit__` method is invoked. If an exception caused the suite to be exited, its type, value, and traceback are passed as arguments to :meth:`__exit__`. Otherwise, three :const:`None` arguments are supplied.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 471); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 471); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 471); *backlink*

Unknown interpreted text role "const".

If the suite was exited due to an exception, and the return value from the :meth:`__exit__` method was false, the exception is reraised. If the return value was true, the exception is suppressed, and execution continues with the statement following the :keyword:`with` statement.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 476); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 476); *backlink*

Unknown interpreted text role "keyword".

If the suite was exited for any reason other than an exception, the return value from :meth:`__exit__` is ignored, and execution proceeds at the normal location for the kind of exit that was taken.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 481); *backlink*

Unknown interpreted text role "meth".

The following code:

```
with EXPRESSION as TARGET:
    SUITE
```

is semantically equivalent to:

```
manager = (EXPRESSION)
enter = type(manager).__enter__
exit = type(manager).__exit__
value = enter(manager)
hit_except = False

try:
    TARGET = value
    SUITE
except:
    hit_except = True
    if not exit(manager, *sys.exc_info()):
        raise
finally:
    if not hit_except:
        exit(manager, None, None, None)
```

With more than one item, the context managers are processed as if multiple :keyword:`with` statements were nested:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 509); *backlink*

Unknown interpreted text role "keyword".

```
with A() as a, B() as b:
    SUITE
```

is semantically equivalent to:

```
with A() as a:
    with B() as b:
        SUITE
```

You can also write multi-item context managers in multiple lines if the items are surrounded by parentheses. For example:

```
with (
    A() as a,
    B() as b,
):
    SUITE
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 530)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.1
   Support for multiple context expressions.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 533)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.10
   Support for using grouping parentheses to break the statement in multiple lines.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 536)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`343` - The "with" statement
      The specification, background, and examples for the Python :keyword:`with`
      statement.
```

The :keyword:`!match` statement

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 544); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 547)

Unknown directive type "index".

```
.. index::
   ! statement: match
   ! keyword: case
   ! single: pattern matching
   keyword: if
   keyword: as
   pair: match; case
   single: : (colon); compound statement
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 556)

Unknown directive type "versionadded".

```
.. versionadded:: 3.10
```

The match statement is used for pattern matching. Syntax:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 560)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   match_stmt: 'match' `subject_expr` ":" NEWLINE INDENT `case_block`+ DEDENT
   subject_expr: `star_named_expression` "," `star_named_expressions`?
               : | `named_expression`
   case_block: 'case' `patterns` [`guard`] ":" `block`
```

Note

This section uses single quotes to denote :ref:`soft keywords <soft-keywords>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 567); *backlink*

Unknown interpreted text role "ref".

Pattern matching takes a pattern as input (following `case`) and a subject value (following `match`). The pattern (which may contain subpatterns) is matched against the subject value. The outcomes are:

- A match success or failure (also termed a pattern success or failure).
- Possible binding of matched values to a name. The prerequisites for this are further discussed below.

The `match` and `case` keywords are :ref:`soft keywords <soft-keywords>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 579); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 581)

Unknown directive type "seealso".

```
.. seealso::

   * :pep:`634` -- Structural Pattern Matching: Specification
   * :pep:`636` -- Structural Pattern Matching: Tutorial
```

Overview

Here's an overview of the logical flow of a match statement:

1. The subject expression `subject_expr` is evaluated and a resulting subject value obtained. If the subject expression contains a comma, a tuple is constructed using :ref`the standard rules <typesseq-tuple>`.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 593); *backlink*

   Unknown interpreted text role "ref".

2. Each pattern in a `case_block` is attempted to match with the subject value. The specific rules for success or failure are described below. The match attempt can also bind some or all of the standalone names within the pattern. The precise pattern binding rules vary per pattern type and are specified below. **Name bindings made during a successful pattern match outlive the executed block and can be used after the match statement**.

   > **Note**
   >
   > During failed pattern matches, some subpatterns may succeed. Do not rely on bindings being made for a failed match. Conversely, do not rely on variables remaining unchanged after a failed match. The exact behavior is dependent on implementation and may vary. This is an intentional decision made to allow different implementations to add optimizations.

3. If the pattern succeeds, the corresponding guard (if present) is evaluated. In this case all name bindings are guaranteed to have happened.
   - If the guard evaluates as true or is missing, the `block` inside `case_block` is executed.
   - Otherwise, the next `case_block` is attempted as described above.
   - If there are no further case blocks, the match statement is completed.

> **Note**
>
> Users should generally never rely on a pattern being evaluated. Depending on implementation, the interpreter may cache values or use other optimizations which skip repeated evaluations.

A sample match statement:

```
>>> flag = False
>>> match (100, 200):
...     case (100, 300):  # Mismatch: 200 != 300
...         print('Case 1')
...     case (100, 200) if flag:  # Successful match, but guard fails
...         print('Case 2')
...     case (100, y):  # Matches and binds y to 200
...         print(f'Case 3, y: {y}')
...     case _:  # Pattern not attempted
...         print('Case 4, I match anything!')
...
Case 3, y: 200
```

In this case, `if flag` is a guard. Read more about that in the next section.

Guards

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 650)

Unknown directive type "index".

```
.. index:: ! guard
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 652)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   guard: "if" `named_expression`
```

A `guard` (which is part of the `case`) must succeed for code inside the `case` block to execute. It takes the form: :keyword:`if` followed by an expression.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 655); *backlink*

Unknown interpreted text role "keyword".

The logical flow of a `case` block with a `guard` follows:

1. Check that the pattern in the `case` block succeeded. If the pattern failed, the `guard` is not evaluated and the next `case` block is checked.
2. If the pattern succeeded, evaluate the `guard`.
   - If the `guard` condition evaluates as true, the case block is selected.
   - If the `guard` condition evaluates as false, the case block is not selected.
   - If the `guard` raises an exception during evaluation, the exception bubbles up.

Guards are allowed to have side effects as they are expressions. Guard evaluation must proceed from the first to the last case block, one at a time, skipping case blocks whose pattern(s) don't all succeed. (I.e., guard evaluation must happen in order.) Guard evaluation must stop once a case block is selected.

Irrefutable Case Blocks

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 689)

Unknown directive type "index".

```
.. index:: irrefutable case block, case block
```

An irrefutable case block is a match-all case block. A match statement may have at most one irrefutable case block, and it must be last.

A case block is considered irrefutable if it has no guard and its pattern is irrefutable. A pattern is considered irrefutable if we can prove from its syntax alone that it will always succeed. Only the following patterns are irrefutable:

- :ref:`as-patterns` whose left-hand side is irrefutable

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 699); *backlink*

  Unknown interpreted text role "ref".

- :ref:`or-patterns` containing at least one irrefutable pattern

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 701); *backlink*

  Unknown interpreted text role "ref".

- :ref:`capture-patterns`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 703); *backlink*

Unknown interpreted text role "ref".

- :ref:`wildcard-patterns`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 705); *backlink*

Unknown interpreted text role "ref".

- parenthesized irrefutable patterns

Patterns

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 713)

Unknown directive type "index".

```
.. index::
   single: ! patterns
   single: AS pattern, OR pattern, capture pattern, wildcard pattern
```

Note

This section uses grammar notations beyond standard EBNF:

- the notation `SEP.RULE+` is shorthand for `RULE (SEP RULE)*`
- the notation `!RULE` is shorthand for a negative lookahead assertion

The top-level syntax for `patterns` is:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 727)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   patterns: `open_sequence_pattern` | `pattern`
   pattern: `as_pattern` | `or_pattern`
   closed_pattern: | `literal_pattern`
                 : | `capture_pattern`
                 : | `wildcard_pattern`
                 : | `value_pattern`
                 : | `group_pattern`
                 : | `sequence_pattern`
                 : | `mapping_pattern`
                 : | `class_pattern`
```

The descriptions below will include a description "in simple terms" of what a pattern does for illustration purposes (credits to Raymond Hettinger for a document that inspired most of the descriptions). Note that these descriptions are purely for illustration purposes and **may not** reflect the underlying implementation. Furthermore, they do not cover all valid forms.

OR Patterns

An OR pattern is two or more patterns separated by vertical bars `|`. Syntax:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 754)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   or_pattern: "|".`closed_pattern`+
```

Only the final subpattern may be :ref:`irrefutable <irrefutable_case>`, and each subpattern must bind the same set of names to avoid

ambiguity.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 757); *backlink*

Unknown interpreted text role "ref".

An OR pattern matches each of its subpatterns in turn to the subject value, until one succeeds. The OR pattern is then considered successful. Otherwise, if none of the subpatterns succeed, the OR pattern fails.

In simple terms, `P1 | P2 | ...` will try to match `P1`, if it fails it will try to match `P2`, succeeding immediately if any succeeds, failing otherwise.

AS Patterns

An AS pattern matches an OR pattern on the left of the :keyword:`as` keyword against a subject. Syntax:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 772); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 775)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   as_pattern: `or_pattern` "as" `capture_pattern`
```

If the OR pattern fails, the AS pattern fails. Otherwise, the AS pattern binds the subject to the name on the right of the as keyword and succeeds. `capture_pattern` cannot be a a `_`.

In simple terms `P as NAME` will match with `P`, and on success it will set `NAME = <subject>`.

Literal Patterns

A literal pattern corresponds to most :ref:`literals <literals>` in Python. Syntax:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 791); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 794)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   literal_pattern: `signed_number`
                  : | `signed_number` "+" NUMBER
                  : | `signed_number` "-" NUMBER
                  : | `strings`
                  : | "None"
                  : | "True"
                  : | "False"
                  : | `signed_number`: NUMBER | "-" NUMBER
```

The rule `strings` and the token `NUMBER` are defined in the :doc:`standard Python grammar <./grammar>`. Triple-quoted strings are supported. Raw strings and byte strings are supported. :ref:`f-strings` are not supported.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 804); *backlink*

Unknown interpreted text role "doc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 804); *backlink*

Unknown interpreted text role "ref".

The forms `signed_number '+' NUMBER` and `signed_number '-' NUMBER` are for expressing :ref:`complex numbers <imaginary>`; they require a real number on the left and an imaginary number on the right. E.g. `3 + 4j`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 809); *backlink***
>
> Unknown interpreted text role "ref".

In simple terms, `LITERAL` will succeed only if `<subject> == LITERAL`. For the singletons `None`, `True` and `False`, the :keyword:`is` operator is used.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 813); *backlink***
>
> Unknown interpreted text role "keyword".

Capture Patterns

A capture pattern binds the subject value to a name. Syntax:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 824)**
>
> Unknown directive type "productionlist".
>
> ```
> .. productionlist:: python-grammar
>    capture_pattern: !'_' NAME
> ```

A single underscore _ is not a capture pattern (this is what `!'_'` expresses). It is instead treated as a :token:`~python-grammar:wildcard_pattern`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 827); *backlink***
>
> Unknown interpreted text role "token".

In a given pattern, a given name can only be bound once. E.g. `case x, x: ...` is invalid while `case [x] | x: ...` is allowed.

Capture patterns always succeed. The binding follows scoping rules established by the assignment expression operator in PEP 572; the name becomes a local variable in the closest containing function scope unless there's an applicable :keyword:`global` or :keyword:`nonlocal` statement.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 834); *backlink***
>
> Unknown interpreted text role "keyword".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 834); *backlink***
>
> Unknown interpreted text role "keyword".

In simple terms `NAME` will always succeed and it will set `NAME = <subject>`.

Wildcard Patterns

A wildcard pattern always succeeds (matches anything) and binds no name. Syntax:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 849)**
>
> Unknown directive type "productionlist".
>
> ```
> .. productionlist:: python-grammar
>    wildcard_pattern: '_'
> ```

_ is a :ref:`soft keyword <soft-keywords>` within any pattern, but only within patterns. It is an identifier, as usual, even within `match` subject expressions, `guard`s, and `case` blocks.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 852); *backlink*

Unknown interpreted text role "ref".

In simple terms, _ will always succeed.

Value Patterns

A value pattern represents a named value in Python. Syntax:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 866)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   value_pattern: `attr`
   attr: `name_or_attr` "." NAME
   name_or_attr: `attr` | NAME
```

The dotted name in the pattern is looked up using standard Python :ref:`name resolution rules <resolve_names>`. The pattern succeeds if the value found compares equal to the subject value (using the == equality operator).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 871); *backlink*

Unknown interpreted text role "ref".

In simple terms `NAME1.NAME2` will succeed only if `<subject> == NAME1.NAME2`

Note

If the same value occurs multiple times in the same match statement, the interpreter may cache the first value found and reuse it rather than repeat the same lookup. This cache is strictly tied to a given execution of a given match statement.

Group Patterns

A group pattern allows users to add parentheses around patterns to emphasize the intended grouping. Otherwise, it has no additional syntax. Syntax:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 894)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   group_pattern: "(" `pattern` ")"
```

In simple terms `(P)` has the same effect as `P`.

Sequence Patterns

A sequence pattern contains several subpatterns to be matched against sequence elements. The syntax is similar to the unpacking of a list or tuple.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 907)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
  sequence_pattern: "[" [`maybe_sequence_pattern`] "]"
                  : | "(" [`open_sequence_pattern`] ")"
  open_sequence_pattern: `maybe_star_pattern` "," [`maybe_sequence_pattern`]
  maybe_sequence_pattern: ",".`maybe_star_pattern`+ ","?
  maybe_star_pattern: `star_pattern` | `pattern`
  star_pattern: "*" (`capture_pattern` | `wildcard_pattern`)
```

There is no difference if parentheses or square brackets are used for sequence patterns (i.e. `(...)` vs `[...]`).

Note

A single pattern enclosed in parentheses without a trailing comma (e.g. `(3 | 4)`) is a :ref:`group pattern <group-patterns>`. While a single pattern enclosed in square brackets (e.g. `[3 | 4]`) is still a sequence pattern.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 919); *backlink*

Unknown interpreted text role "ref".

At most one star subpattern may be in a sequence pattern. The star subpattern may occur in any position. If no star subpattern is present, the sequence pattern is a fixed-length sequence pattern; otherwise it is a variable-length sequence pattern.

The following is the logical flow for matching a sequence pattern against a subject value:

1. If the subject value is not a sequence [2], the sequence pattern fails.
2. If the subject value is an instance of `str`, `bytes` or `bytearray` the sequence pattern fails.
3. The subsequent steps depend on whether the sequence pattern is fixed or variable-length.

   If the sequence pattern is fixed-length:

   1. If the length of the subject sequence is not equal to the number of subpatterns, the sequence pattern fails
   2. Subpatterns in the sequence pattern are matched to their corresponding items in the subject sequence from left to right. Matching stops as soon as a subpattern fails. If all subpatterns succeed in matching their corresponding item, the sequence pattern succeeds.

   Otherwise, if the sequence pattern is variable-length:

   1. If the length of the subject sequence is less than the number of non-star subpatterns, the sequence pattern fails.
   2. The leading non-star subpatterns are matched to their corresponding items as for fixed-length sequences.
   3. If the previous step succeeds, the star subpattern matches a list formed of the remaining subject items, excluding the remaining items corresponding to non-star subpatterns following the star subpattern.
   4. Remaining non-star subpatterns are matched to their corresponding subject items, as for a fixed-length sequence.

   Note

   The length of the subject sequence is obtained via :func:`len` (i.e. via the :meth:`__len__` protocol). This length may be cached by the interpreter in a similar manner as :ref:`value patterns <value-patterns>`.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 966); *backlink*

   Unknown interpreted text role "func".

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 966); *backlink*

   Unknown interpreted text role "meth".

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 966); *backlink*

   Unknown interpreted text role "ref".

In simple terms `[P1, P2, P3, ... , P<N>]` matches only if all the following happens:

- check `<subject>` is a sequence
- `len(subject) == <N>`
- `P1` matches `<subject>[0]` (note that this match can also bind names)
- `P2` matches `<subject>[1]` (note that this match can also bind names)
- ... and so on for the corresponding pattern/element.

Mapping Patterns

A mapping pattern contains one or more key-value patterns. The syntax is similar to the construction of a dictionary. Syntax:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 990)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   mapping_pattern: "{" [`items_pattern`] "}"
   items_pattern: ",".`key_value_pattern`+ ","?
   key_value_pattern: (`literal_pattern` | `value_pattern`) ":" `pattern`
                    : | `double_star_pattern`
   double_star_pattern: "**" `capture_pattern`
```

At most one double star pattern may be in a mapping pattern. The double star pattern must be the last subpattern in the mapping pattern.

Duplicate keys in mapping patterns are disallowed. Duplicate literal keys will raise a :exc:`SyntaxError`. Two keys that otherwise have the same value will raise a :exc:`ValueError` at runtime.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1000); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1000); *backlink*

Unknown interpreted text role "exc".

The following is the logical flow for matching a mapping pattern against a subject value:

1. If the subject value is not a mapping [3],the mapping pattern fails.
2. If every key given in the mapping pattern is present in the subject mapping, and the pattern for each key matches the corresponding item of the subject mapping, the mapping pattern succeeds.
3. If duplicate keys are detected in the mapping pattern, the pattern is considered invalid. A :exc:`SyntaxError` is raised for duplicate literal values; or a :exc:`ValueError` for named keys of the same value.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1013); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1013); *backlink*

Unknown interpreted text role "exc".

Note

Key-value pairs are matched using the two-argument form of the mapping subject's `get()` method. Matched key-value pairs must already be present in the mapping, and not created on-the-fly via :meth:`__missing__` or :meth:`__getitem__`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1017); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1017); *backlink*

Unknown interpreted text role "meth".

In simple terms `{KEY1: P1, KEY2: P2, ... }` matches only if all the following happens:

- check `<subject>` is a mapping
- `KEY1 in <subject>`
- `P1` matches `<subject>[KEY1]`
- ... and so on for the corresponding KEY/pattern pair.

Class Patterns

A class pattern represents a class and its positional and keyword arguments (if any). Syntax:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1039)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
  class_pattern: `name_or_attr` "(" [`pattern_arguments` ","?] ")"
  pattern_arguments: `positional_patterns` ["," `keyword_patterns`]
                   : | `keyword_patterns`
  positional_patterns: ",".`pattern`+
  keyword_patterns: ",".`keyword_pattern`+
  keyword_pattern: NAME "=" `pattern`
```

The same keyword should not be repeated in class patterns.

The following is the logical flow for matching a class pattern against a subject value:

1. If `name_or_attr` is not an instance of the builtin :class:`type` , raise :exc:`TypeError`.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1052); *backlink*

   Unknown interpreted text role "class".

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1052); *backlink*

   Unknown interpreted text role "exc".

2. If the subject value is not an instance of `name_or_attr` (tested via :func:`isinstance`), the class pattern fails.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1055); *backlink*

   Unknown interpreted text role "func".

3. If no pattern arguments are present, the pattern succeeds. Otherwise, the subsequent steps depend on whether keyword or positional argument patterns are present.

   For a number of built-in types (specified below), a single positional subpattern is accepted which will match the entire subject; for these types keyword patterns also work as for other types.

   If only keyword patterns are present, they are processed as follows, one by one:

   I. The keyword is looked up as an attribute on the subject.

   - If this raises an exception other than :exc:`AttributeError`, the exception bubbles up.

     System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1071); *backlink*

     Unknown interpreted text role "exc".

   - If this raises :exc:`AttributeError`, the class pattern has failed.

     **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1074); *backlink*

Unknown interpreted text role "exc".

- Else, the subpattern associated with the keyword pattern is matched against the subject's attribute value. If this fails, the class pattern fails; if this succeeds, the match proceeds to the next keyword.

II. If all keyword patterns succeed, the class pattern succeeds.

If any positional patterns are present, they are converted to keyword patterns using the :data:`~object.__match_args__` attribute on the class `name_or_attr` before matching:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1083); *backlink*

Unknown interpreted text role "data".

I. The equivalent of `getattr(cls, "__match_args__", ())` is called.

- If this raises an exception, the exception bubbles up.
- If the returned value is not a tuple, the conversion fails and :exc:`TypeError` is raised.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1091); *backlink*

Unknown interpreted text role "exc".

- If there are more positional patterns than `len(cls.__match_args__)`, :exc:`TypeError` is raised.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1094); *backlink*

Unknown interpreted text role "exc".

- Otherwise, positional pattern `i` is converted to a keyword pattern using `__match_args__[i]` as the keyword. `__match_args__[i]` must be a string; if not :exc:`TypeError` is raised.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1097); *backlink*

Unknown interpreted text role "exc".

- If there are duplicate keywords, :exc:`TypeError` is raised.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1101); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1103)

Unknown directive type "seealso".

```
.. seealso:: :ref:`class-pattern-matching`
```

II. Once all positional patterns have been converted to keyword patterns, the match proceeds as if there were only keyword patterns.

For the following built-in types the handling of positional subpatterns is different:

- :class:`bool`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1111); *backlink*

Unknown interpreted text role "class".

- :class:`bytearray`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1112); *backlink*

Unknown interpreted text role "class".

- :class:`bytes`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1113); *backlink*

Unknown interpreted text role "class".

- :class:`dict`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1114); *backlink*

Unknown interpreted text role "class".

- :class:`float`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1115); *backlink*

Unknown interpreted text role "class".

- :class:`frozenset`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1116); *backlink*

Unknown interpreted text role "class".

- :class:`int`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1117); *backlink*

Unknown interpreted text role "class".

- :class:`list`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1118); *backlink*

Unknown interpreted text role "class".

- :class:`set`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1119); *backlink*

Unknown interpreted text role "class".

- :class:`str`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1120); *backlink*

  Unknown interpreted text role "class".

- :class:`tuple`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1121); *backlink*

  Unknown interpreted text role "class".

These classes accept a single positional argument, and the pattern there is matched against the whole object rather than an attribute. For example `int(0|1)` matches the value `0`, but not the values `0.0` or `False`.

In simple terms `CLS(P1, attr=P2)` matches only if the following happens:

- `isinstance(<subject>, CLS)`
- convert `P1` to a keyword pattern using `CLS.__match_args__`
- For each keyword argument `attr=P2`:
  - `hasattr(<subject>, "attr")`
  - `P2` matches `<subject>.attr`
- ... and so on for the corresponding keyword argument/pattern pair.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1136)

Unknown directive type "seealso".

```
.. seealso::

   * :pep:`634` -- Structural Pattern Matching: Specification
   * :pep:`636` -- Structural Pattern Matching: Tutorial
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1142)

Unknown directive type "index".

```
.. index::
   single: parameter; function definition
```

Function definitions

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1151)

Unknown directive type "index".

```
.. index::
   statement: def
   pair: function; definition
   pair: function; name
   pair: name; binding
   object: user-defined function
   object: function
   pair: function; name
   pair: name; binding
   single: () (parentheses); function definition
   single: , (comma); parameter list
   single: : (colon); compound statement
```

A function definition defines a user-defined function object (see section :ref:`types`):

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1164); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1167)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   funcdef: [`decorators`] "def" `funcname` "(" [`parameter_list`] ")"
          : ["->" `expression`] ":" `suite`
   decorators: `decorator`+
   decorator: "@" `assignment_expression` NEWLINE
   parameter_list: `defparameter` ("," `defparameter`)* "," "/" ["," [`parameter_list_no_posonly`]]
                 :   | `parameter_list_no_posonly`
   parameter_list_no_posonly: `defparameter` ("," `defparameter`)* ["," [`parameter_list_starargs`]]
                            : | `parameter_list_starargs`
   parameter_list_starargs: "*" [`parameter`] ("," `defparameter`)* ["," ["**" `parameter` [","]]]
                          : | "**" `parameter` [","]
   parameter: `identifier` [":" `expression`]
   defparameter: `parameter` ["=" `expression`]
   funcname: `identifier`
```

A function definition is an executable statement. Its execution binds the function name in the current local namespace to a function object (a wrapper around the executable code for the function). This function object contains a reference to the current global namespace as the global namespace to be used when the function is called.

The function definition does not execute the function body; this gets executed only when the function is called. [4]

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1192)

Unknown directive type "index".

```
.. index::
   single: @ (at); function definition
```

A function definition may be wrapped by one or more :term:`decorator` expressions. Decorator expressions are evaluated when the function is defined, in the scope that contains the function definition. The result must be a callable, which is invoked with the function object as the only argument. The returned value is bound to the function name instead of the function object. Multiple decorators are applied in nested fashion. For example, the following code

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1195); *backlink*

Unknown interpreted text role "term".

```
@f1(arg)
@f2
def func(): pass
```

is roughly equivalent to

```
def func(): pass
func = f1(arg)(f2(func))
```

except that the original function is not temporarily bound to the name `func`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1213)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.9
   Functions may be decorated with any valid
   :token:`~python-grammar:assignment_expression`. Previously, the grammar was
   much more restrictive; see :pep:`614` for details.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1218)

Unknown directive type "index".

```
.. index::
   triple: default; parameter; value
   single: argument; function definition
   single: = (equals); function definition
```

When one or more :term:`parameters <parameter>` have the form *parameter* = *expression*, the function is said to have "default parameter values." For a parameter with a default value, the corresponding :term:`argument` may be omitted from a call, in which case the parameter's default value is substituted. If a parameter has a default value, all following parameters up until the "`*`" must also have a default value --- this is a syntactic restriction that is not expressed by the grammar.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1223); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1223); *backlink*

Unknown interpreted text role "term".

Default parameter values are evaluated from left to right when the function definition is executed. This means that the expression is evaluated once, when the function is defined, and that the same "pre-computed" value is used for each call. This is especially important to understand when a default parameter value is a mutable object, such as a list or a dictionary: if the function modifies the object (e.g. by appending an item to a list), the default parameter value is in effect modified. This is generally not what was intended. A way around this is to use `None` as the default, and explicitly test for it in the body of the function, e.g.:

```
def whats_on_the_telly(penguin=None):
    if penguin is None:
        penguin = []
    penguin.append("property of the zoo")
    return penguin
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1247)

Unknown directive type "index".

```
.. index::
   single: / (slash); function definition
   single: * (asterisk); function definition
   single: **; function definition
```

Function call semantics are described in more detail in section :ref:`calls`. A function call always assigns values to all parameters mentioned in the parameter list, either from positional arguments, from keyword arguments, or from default values. If the form "`*identifier`" is present, it is initialized to a tuple receiving any excess positional parameters, defaulting to the empty tuple. If the form "`**identifier`" is present, it is initialized to a new ordered mapping receiving any excess keyword arguments, defaulting to a new empty mapping of the same type. Parameters after "`*`" or "`*identifier`" are keyword-only parameters and may only be passed by keyword arguments. Parameters before "`/`" are positional-only parameters and may only be passed by positional arguments.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1252); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1264)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.8
   The ``/`` function parameter syntax may be used to indicate positional-only
   parameters. See :pep:`570` for details.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1268)

Unknown directive type "index".

```
.. index::
   pair: function; annotations
   single: ->; function annotations
   single: : (colon); function annotations
```

Parameters may have an :term:`annotation <function annotation>` of the form "`: expression`" following the parameter name. Any parameter may have an annotation, even those of the form `*identifier` or `**identifier`. Functions may have "return" annotation of the form "`-> expression`" after the parameter list. These annotations can be any valid Python expression. The presence of annotations does not change the semantics of a function. The annotation values are available as values of a dictionary keyed by the parameters' names in the :attr:`__annotations__` attribute of the function object. If the `annotations` import from :mod:`__future__` is used, annotations are preserved as strings at runtime which enables postponed evaluation. Otherwise, they are evaluated when the function definition is executed. In this case annotations may be evaluated in a different order than they appear in the source code.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1273); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1273); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1273); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1286)

Unknown directive type "index".

```
.. index:: pair: lambda; expression
```

It is also possible to create anonymous functions (functions not bound to a name), for immediate use in expressions. This uses lambda expressions, described in section :ref:`lambda`. Note that the lambda expression is merely a shorthand for a simplified function definition; a function defined in a "`:keyword:`def`" statement can be passed around or assigned to another name just like a function defined by a lambda expression. The "`:keyword:`!def`" form is actually more powerful since it allows the execution of multiple statements and annotations.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1288); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1288); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1288); *backlink*

Unknown interpreted text role "keyword".

Programmer's note: Functions are first-class objects. A "`def`" statement executed inside a function definition defines a local function that can be returned or passed around. Free variables used in the nested function can access the local variables of the function containing the def. See section :ref:`naming` for details.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1296); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1302)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`3107` - Function Annotations
      The original specification for function annotations.

   :pep:`484` - Type Hints
      Definition of a standard meaning for annotations: type hints.

   :pep:`526` - Syntax for Variable Annotations
      Ability to type hint variable declarations, including class
      variables and instance variables

   :pep:`563` - Postponed Evaluation of Annotations
      Support for forward references within annotations by preserving
      annotations in a string form at runtime instead of eager evaluation.
```

Class definitions

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1324)

Unknown directive type "index".

```
.. index::
   object: class
   statement: class
   pair: class; definition
   pair: class; name
   pair: name; binding
   pair: execution; frame
   single: inheritance
   single: docstring
   single: () (parentheses); class definition
   single: , (comma); expression list
   single: : (colon); compound statement
```

A class definition defines a class object (see section :ref:`types`):

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1337); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1339)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   classdef: [`decorators`] "class" `classname` [`inheritance`] ":" `suite`
   inheritance: "(" [`argument_list`] ")"
   classname: `identifier`
```

A class definition is an executable statement. The inheritance list usually gives a list of base classes (see :ref:`metaclasses` for more advanced uses), so each item in the list should evaluate to a class object which allows subclassing. Classes without an inheritance list inherit, by default, from the base class :class:`object`; hence,

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1344); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1344); *backlink*

Unknown interpreted text role "class".

```
class Foo:
    pass
```

is equivalent to

```
class Foo(object):
    pass
```

The class's suite is then executed in a new execution frame (see :ref:`naming`), using a newly created local namespace and the original global namespace. (Usually, the suite contains mostly function definitions.) When the class's suite finishes execution, its execution frame is discarded but its local namespace is saved. [5] A class object is then created using the inheritance list for the base classes and the saved local namespace for the attribute dictionary. The class name is bound to this class object in the original local namespace.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1358); *backlink***
>
> Unknown interpreted text role "ref".

The order in which attributes are defined in the class body is preserved in the new class's `__dict__`. Note that this is reliable only right after the class is created and only for classes that were defined using the definition syntax.

Class creation can be customized heavily using :ref:`metaclasses <metaclasses>`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1372); *backlink***
>
> Unknown interpreted text role "ref".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1374)**
>
> Unknown directive type "index".
>
> ```
> .. index::
>    single: @ (at); class definition
> ```

Classes can also be decorated: just like when decorating functions,

```
@f1(arg)
@f2
class Foo: pass
```

is roughly equivalent to

```
class Foo: pass
Foo = f1(arg)(f2(Foo))
```

The evaluation rules for the decorator expressions are the same as for function decorators. The result is then bound to the class name.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1391)**
>
> Unknown directive type "versionchanged".
>
> ```
> .. versionchanged:: 3.9
>    Classes may be decorated with any valid
>    :token:`~python-grammar:assignment_expression`. Previously, the grammar was
>    much more restrictive; see :pep:`614` for details.
> ```

Programmer's note: Variables defined in the class definition are class attributes; they are shared by instances. Instance attributes can be set in a method with `self.name = value`. Both class and instance attributes are accessible through the notation "`self.name`", and an instance attribute hides a class attribute with the same name when accessed in this way. Class attributes can be used as defaults for instance attributes, but using mutable values there can lead to unexpected results. :ref:`Descriptors <descriptors>` can be used to create instance variables with different implementation details.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1396); *backlink***
>
> Unknown interpreted text role "ref".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-**

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1406)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`3115` - Metaclasses in Python 3000
      The proposal that changed the declaration of metaclasses to the current
      syntax, and the semantics for how classes with metaclasses are
      constructed.

   :pep:`3129` - Class Decorators
      The proposal that added class decorators.  Function and method decorators
      were introduced in :pep:`318`.
```

Coroutines

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1423)

Unknown directive type "versionadded".

```
.. versionadded:: 3.5
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1425)

Unknown directive type "index".

```
.. index:: statement: async def
```

Coroutine function definition

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1431)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   async_funcdef: [`decorators`] "async" "def" `funcname` "(" [`parameter_list`] ")"
                : ["->" `expression`] ":" `suite`
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1435)

Unknown directive type "index".

```
.. index::
   keyword: async
   keyword: await
```

Execution of Python coroutines can be suspended and resumed at many points (see :term:`coroutine`). :keyword:`await` expressions, :keyword:`async for` and :keyword:`async with` can only be used in the body of a coroutine function.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1439); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1439); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1439); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1439); *backlink*

Unknown interpreted text role "keyword".

Functions defined with `async def` syntax are always coroutine functions, even if they do not contain `await` or `async` keywords.

It is a :exc:`SyntaxError` to use a `yield from` expression inside the body of a coroutine function.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1446); *backlink*

Unknown interpreted text role "exc".

An example of a coroutine function:

```
async def func(param1, param2):
    do_stuff()
    await some_coroutine()
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1455)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
   ``await`` and ``async`` are now keywords; previously they were only
   treated as such inside the body of a coroutine function.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1459)

Unknown directive type "index".

```
.. index:: statement: async for
```

The :keyword:`!async for` statement

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1462); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1465)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   async_for_stmt: "async" `for_stmt`
```

An :term:`asynchronous iterable` provides an `__aiter__` method that directly returns an :term:`asynchronous iterator`, which can call asynchronous code in its `__anext__` method.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1468); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1468); *backlink*

Unknown interpreted text role "term".

The `async for` statement allows convenient iteration over asynchronous iterables.

The following code:

```
async for TARGET in ITER:
    SUITE
else:
    SUITE2
```

Is semantically equivalent to:

```
iter = (ITER)
iter = type(iter).__aiter__(iter)
running = True

while running:
    try:
        TARGET = await type(iter).__anext__(iter)
    except StopAsyncIteration:
        running = False
    else:
        SUITE
else:
    SUITE2
```

See also :meth:`__aiter__` and :meth:`__anext__` for details.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1498); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1498); *backlink*

Unknown interpreted text role "meth".

It is a :exc:`SyntaxError` to use an `async for` statement outside the body of a coroutine function.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1500); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1504)

Unknown directive type "index".

```
.. index:: statement: async with
```

The :keyword:`!async with` statement

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1507); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1510)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   async_with_stmt: "async" `with_stmt`
```

An :term:`asynchronous context manager` is a :term:`context manager` that is able to suspend execution in its *enter* and *exit* methods.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1513); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1513); *backlink*

Unknown interpreted text role "term".

The following code:

```
async with EXPRESSION as TARGET:
    SUITE
```

is semantically equivalent to:

```
manager = (EXPRESSION)
aenter = type(manager).__aenter__
aexit = type(manager).__aexit__
value = await aenter(manager)
hit_except = False

try:
    TARGET = value
    SUITE
except:
    hit_except = True
    if not await aexit(manager, *sys.exc_info()):
        raise
finally:
    if not hit_except:
        await aexit(manager, None, None, None)
```

See also :meth:`__aenter__` and :meth:`__aexit__` for details.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1540); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1540); *backlink*

Unknown interpreted text role "meth".

It is a :exc:`SyntaxError` to use an `async with` statement outside the body of a coroutine function.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1542); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1545)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`492` - Coroutines with async and await syntax
      The proposal that made coroutines a proper standalone concept in Python,
      and added supporting syntax.
```

Footnotes

[1] The exception is propagated to the invocation stack unless there is a :keyword:`finally` clause which happens to raise another exception. That new exception causes the old one to be lost.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1554); *backlink*

Unknown interpreted text role "keyword".

[2] In pattern matching, a sequence is defined as one of the following:

- a class that inherits from :class:`collections.abc.Sequence`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1560); backlink

Unknown interpreted text role "class".

- a Python class that has been registered as :class:`collections.abc.Sequence`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1561); backlink

Unknown interpreted text role "class".

- a builtin class that has its (CPython) :data:`Py_TPFLAGS_SEQUENCE` bit set

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1562); backlink

Unknown interpreted text role "data".

- a class that inherits from any of the above

The following standard library classes are sequences:

- :class:`array.array`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1567); backlink

Unknown interpreted text role "class".

- :class:`collections.deque`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1568); backlink

Unknown interpreted text role "class".

- :class:`list`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1569); backlink

Unknown interpreted text role "class".

- :class:`memoryview`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1570); backlink

Unknown interpreted text role "class".

- :class:`range`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1571); backlink

Unknown interpreted text role "class".

- :class:`tuple`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1572); backlink

Unknown interpreted text role "class".

Note

Subject values of type `str`, `bytes`, and `bytearray` do not match sequence patterns.

[3] In pattern matching, a mapping is defined as one of the following:

- a class that inherits from :class:`collections.abc.Mapping`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1579); backlink

Unknown interpreted text role "class".

- a Python class that has been registered as :class:`collections.abc.Mapping`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1580); backlink

Unknown interpreted text role "class".

- a builtin class that has its (CPython) :data:`Py_TPFLAGS_MAPPING` bit set

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1581); backlink

Unknown interpreted text role "data".

- a class that inherits from any of the above

The standard library classes :class:`dict` and :class:`types.MappingProxyType` are mappings.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1584); backlink

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1584); backlink

Unknown interpreted text role "class".

[4] A string literal appearing as the first statement in the function body is transformed into the function's `__doc__` attribute and therefore the function's :term:`docstring`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1587); backlink

Unknown interpreted text role "term".

[5] A string literal appearing as the first statement in the class body is transformed into the namespace's `__doc__` item and therefore the class's :term:`docstring`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]compound_stmts.rst, line 1591); backlink

Unknown interpreted text role "term".