Simple statements

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 8)

Unknown directive type "index".

```
.. index:: pair: simple; statement
```

A simple statement is comprised within a single logical line. Several simple statements may occur on a single line separated by semicolons. The syntax for simple statements is:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 14)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   simple_stmt: `expression_stmt`
              : | `assert_stmt`
              : | `assignment_stmt`
              : | `augmented_assignment_stmt`
              : | `annotated_assignment_stmt`
              : | `pass_stmt`
              : | `del_stmt`
              : | `return_stmt`
              : | `yield_stmt`
              : | `raise_stmt`
              : | `break_stmt`
              : | `continue_stmt`
              : | `import_stmt`
              : | `future_stmt`
              : | `global_stmt`
              : | `nonlocal_stmt`
```

Expression statements

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 38)

Unknown directive type "index".

```
.. index::
   pair: expression; statement
   pair: expression; list
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 41)

Unknown directive type "index".

```
.. index:: pair: expression; list
```

Expression statements are used (mostly interactively) to compute and write a value, or (usually) to call a procedure (a function that returns no meaningful result; in Python, procedures return the value `None`). Other uses of expression statements are allowed and occasionally useful. The syntax for an expression statement is:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 49)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   expression_stmt: `starred_expression`
```

An expression statement evaluates the expression list (which may be a single expression).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 55)

Unknown directive type "index".

```
.. index::
   builtin: repr
   object: None
   pair: string; conversion
   single: output
   pair: standard; output
   pair: writing; values
   pair: procedure; call
```

In interactive mode, if the value is not `None`, it is converted to a string using the built-in :func:`repr` function and the resulting string is written to standard output on a line by itself (except if the result is `None`, so that procedure calls do not cause any output.)

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 64); *backlink*

Unknown interpreted text role "func".

Assignment statements

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 74)

Unknown directive type "index".

```
.. index::
   single: = (equals); assignment statement
   pair: assignment; statement
   pair: binding; name
   pair: rebinding; name
   object: mutable
   pair: attribute; assignment
```

Assignment statements are used to (re)bind names to values and to modify attributes or items of mutable objects:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 85)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   assignment_stmt: (`target_list` "=")+ (`starred_expression` | `yield_expression`)
   target_list: `target` ("," `target`)* [","]
   target: `identifier`
         : | "(" [`target_list`] ")"
         : | "[" [`target_list`] "]"
         : | `attributeref`
         : | `subscription`
         : | `slicing`
         : | "*" `target`
```

(See section :ref:`primaries` for the syntax definitions for *attributeref*, *subscription*, and *slicing*.)

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 96); *backlink*

Unknown interpreted text role "ref".

An assignment statement evaluates the expression list (remember that this can be a single expression or a comma-separated list, the latter yielding a tuple) and assigns the single resulting object to each of the target lists, from left to right.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 104)

Unknown directive type "index".

```
.. index::
   single: target
   pair: target; list
```

Assignment is defined recursively depending on the form of the target (list). When a target is part of a mutable object (an attribute reference, subscription or slicing), the mutable object must ultimately perform the assignment and decide about its validity, and may raise an exception if the assignment is unacceptable. The rules observed by various types and the exceptions raised are given with the definition of the object types (see section :ref:`types`).

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

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 115)

Unknown directive type "index".

```
.. index:: triple: target; list; assignment
   single: , (comma); in target list
   single: * (asterisk); in assignment target list
   single: [] (square brackets); in assignment target list
   single: () (parentheses); in assignment target list
```

Assignment of an object to a target list, optionally enclosed in parentheses or square brackets, is recursively defined as follows.

- If the target list is a single target with no trailing comma, optionally in parentheses, the object is assigned to that target.
- Else:
  - If the target list contains one target prefixed with an asterisk, called a "starred" target: The object must be an iterable with at least as many items as there are targets in the target list, minus one. The first items of the iterable are assigned, from left to right, to the targets before the starred target. The final items of the iterable are assigned to the targets after the starred target. A list of the remaining items in the iterable is then assigned to the starred target (the list can be empty).
  - Else: The object must be an iterable with the same number of items as there are targets in the target list, and the items are assigned, from left to right, to the corresponding targets.

Assignment of an object to a single target is recursively defined as follows.

- If the target is an identifier (name):
  - If the name does not occur in a :keyword:`global` or :keyword:`nonlocal` statement in the current code block: the name is bound to the object in the current local namespace.

    System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 145); *backlink*

    Unknown interpreted text role "keyword".

    System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 145); *backlink*

    Unknown interpreted text role "keyword".

  - Otherwise: the name is bound to the object in the global namespace or the outer namespace determined by :keyword:`nonlocal`, respectively.

    System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 149); *backlink*

    Unknown interpreted text role "keyword".

  **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc)`

(reference)simple_stmts.rst, line 152)

Unknown directive type "index".

```
.. index:: single: destructor
```

The name is rebound if it was already bound. This may cause the reference count for the object previously bound to the name to reach zero, causing the object to be deallocated and its destructor (if it has one) to be called.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst, line 158)

Unknown directive type "index".

```
.. index:: pair: attribute; assignment
```

- If the target is an attribute reference: The primary expression in the reference is evaluated. It should yield an object with assignable attributes; if this is not the case, :exc:`TypeError` is raised. That object is then asked to assign the assigned object to the given attribute; if it cannot perform the assignment, it raises an exception (usually but not necessarily :exc:`AttributeError`).

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

Unknown interpreted text role "exc".

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

Unknown interpreted text role "exc".

Note: If the object is a class instance and the attribute reference occurs on both sides of the assignment operator, the right-hand side expression, `a.x` can access either an instance attribute or (if no instance attribute exists) a class attribute. The left-hand side target `a.x` is always set as an instance attribute, creating it if necessary. Thus, the two occurrences of `a.x` do not necessarily refer to the same attribute: if the right-hand side expression refers to a class attribute, the left-hand side creates a new instance attribute as the target of the assignment:

```
class Cls:
    x = 3             # class variable
inst = Cls()
inst.x = inst.x + 1   # writes inst.x as 4 leaving Cls.x as 3
```

This description does not necessarily apply to descriptor attributes, such as properties created with :func:`property`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst, line 183); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst, line 186)

Unknown directive type "index".

```
.. index::
   pair: subscription; assignment
   object: mutable
```

- If the target is a subscription: The primary expression in the reference is evaluated. It should yield either a mutable sequence object (such as a list) or a mapping object (such as a dictionary). Next, the subscript expression is evaluated.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)

(reference)simple_stmts.rst, line 195)

Unknown directive type "index".

```
.. index::
   object: sequence
   object: list
```

If the primary is a mutable sequence object (such as a list), the subscript must yield an integer. If it is negative, the sequence's length is added to it. The resulting value must be a nonnegative integer less than the sequence's length, and the sequence is asked to assign the assigned object to its item with that index. If the index is out of range, :exc:`IndexError` is raised (assignment to a subscripted sequence cannot add new items to a list).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst, line 199); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst, line 206)

Unknown directive type "index".

```
.. index::
   object: mapping
   object: dictionary
```

If the primary is a mapping object (such as a dictionary), the subscript must have a type compatible with the mapping's key type, and the mapping is then asked to create a key/datum pair which maps the subscript to the assigned object. This can either replace an existing key/value pair with the same key value, or insert a new key/value pair (if no key with the same value existed).

For user-defined objects, the :meth:`__setitem__` method is called with appropriate arguments.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst, line 216); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst, line 219)

Unknown directive type "index".

```
.. index:: pair: slicing; assignment
```

- If the target is a slicing: The primary expression in the reference is evaluated. It should yield a mutable sequence object (such as a list). The assigned object should be a sequence object of the same type. Next, the lower and upper bound expressions are evaluated, insofar they are present; defaults are zero and the sequence's length. The bounds should evaluate to integers. If either bound is negative, the sequence's length is added to it. The resulting bounds are clipped to lie between zero and the sequence's length, inclusive. Finally, the sequence object is asked to replace the slice with the items of the assigned sequence. The length of the slice may be different from the length of the assigned sequence, thus changing the length of the target sequence, if the target sequence allows it.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst, line 233)

Unknown directive type "impl-detail".

```
.. impl-detail::

   In the current implementation, the syntax for targets is taken to be the same
   as for expressions, and invalid syntax is rejected during the code generation
   phase, causing less detailed error messages.
```

Although the definition of assignment implies that overlaps between the left-hand side and the right-hand side are 'simultaneous' (for example `a, b = b, a` swaps two variables), overlaps *within* the collection of assigned-to variables occur left-to-right, sometimes resulting in confusion. For instance, the following program prints `[0, 2]`:

```
x = [0, 1]
i = 0
i, x[i] = 1, 2         # i is updated, then x[i] is updated
print(x)
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst`, line 251)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`3132` - Extended Iterable Unpacking
      The specification for the ``*target`` feature.
```

Augmented assignment statements

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst`, line 262)

Unknown directive type "index".

```
.. index::
   pair: augmented; assignment
   single: statement; assignment, augmented
   single: +=; augmented assignment
   single: -=; augmented assignment
   single: *=; augmented assignment
   single: /=; augmented assignment
   single: %=; augmented assignment
   single: &=; augmented assignment
   single: ^=; augmented assignment
   single: |=; augmented assignment
   single: **=; augmented assignment
   single: //=; augmented assignment
   single: >>=; augmented assignment
   single: <<=; augmented assignment
```

Augmented assignment is the combination, in a single statement, of a binary operation and an assignment statement:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst`, line 281)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   augmented_assignment_stmt: `augtarget` `augop` (`expression_list` | `yield_expression`)
   augtarget: `identifier` | `attributeref` | `subscription` | `slicing`
   augop: "+=" | "-=" | "*=" | "@=" | "/=" | "//=" | "%=" | "**="
        : | ">>=" | "<<=" | "&=" | "^=" | "|="
```

(See section :ref:`primaries` for the syntax definitions of the last three symbols.)

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst`, line 287); *backlink*

Unknown interpreted text role "ref".

An augmented assignment evaluates the target (which, unlike normal assignment statements, cannot be an unpacking) and the expression list, performs the binary operation specific to the type of assignment on the two operands, and assigns the result to the original target. The target is only evaluated once.

An augmented assignment expression like `x += 1` can be rewritten as `x = x + 1` to achieve a similar, but not exactly equal effect. In the augmented version, `x` is only evaluated once. Also, when possible, the actual operation is performed *in-place*, meaning that rather than creating a new object and assigning that to the target, the old object is modified instead.

Unlike normal assignments, augmented assignments evaluate the left-hand side *before* evaluating the right-hand side. For example, `a[i] += f(x)` first looks-up `a[i]`, then it evaluates `f(x)` and performs the addition, and lastly, it writes the result back to `a[i]`.

With the exception of assigning to tuples and multiple targets in a single statement, the assignment done by augmented assignment statements is handled the same way as normal assignments. Similarly, with the exception of the possible *in-place* behavior, the binary operation performed by augmented assignment is the same as the normal binary operations.

For targets which are attribute references, the same :ref:`caveat about class and instance attributes <attr-target-note>` applies as for regular assignments.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 312); *backlink*

Unknown interpreted text role "ref".

Annotated assignment statements

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 321)

Unknown directive type "index".

```
.. index::
   pair: annotated; assignment
   single: statement; assignment, annotated
   single: : (colon); annotated variable
```

:term:`Annotation <variable annotation>` assignment is the combination, in a single statement, of a variable or attribute annotation and an optional assignment statement:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 326); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 329)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   annotated_assignment_stmt: `augtarget` ":" `expression`
                            : ["=" (`starred_expression` | `yield_expression`)]
```

The difference from normal :ref:`assignment` is that only single target is allowed.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 333); *backlink*

Unknown interpreted text role "ref".

For simple names as assignment targets, if in class or module scope, the annotations are evaluated and stored in a special class or module attribute :attr:`__annotations__` that is a dictionary mapping from variable names (mangled if private) to evaluated annotations. This attribute is writable and is automatically created at the start of class or module body execution, if annotations are found statically.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 335); *backlink*

Unknown interpreted text role "attr".

For expressions as assignment targets, the annotations are evaluated if in class or module scope, but not stored.

If a name is annotated in a function scope, then this name is local for that scope. Annotations are never evaluated and stored in function scopes.

If the right hand side is present, an annotated assignment performs the actual assignment before evaluating annotations (where applicable). If the right hand side is not present for an expression target, then the interpreter evaluates the target except for the last

:meth:`__setitem__` or :meth:`__setattr__` call.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 349); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 349); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 355)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`526` - Syntax for Variable Annotations
      The proposal that added syntax for annotating the types of variables
      (including class variables and instance variables), instead of expressing
      them through comments.

   :pep:`484` - Type hints
      The proposal that added the :mod:`typing` module to provide a standard
      syntax for type annotations that can be used in static analysis tools and
      IDEs.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 367)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.8
   Now annotated assignments allow same expressions in the right hand side as
   the regular assignments. Previously, some expressions (like un-parenthesized
   tuple expressions) caused a syntax error.
```

The :keyword:`!assert` statement

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 375); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 378)

Unknown directive type "index".

```
.. index::
   ! statement: assert
   pair: debugging; assertions
   single: , (comma); expression list
```

Assert statements are a convenient way to insert debugging assertions into a program:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 386)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   assert_stmt: "assert" `expression` ["," `expression`]
```

The simple form, `assert expression`, is equivalent to

```
if __debug__:
    if not expression: raise AssertionError
```

The extended form, `assert expression1, expression2`, is equivalent to

```
if __debug__:
    if not expression1: raise AssertionError(expression2)
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 399)

Unknown directive type "index".

```
.. index::
   single: __debug__
   exception: AssertionError
```

These equivalences assume that :const:`__debug__` and :exc:`AssertionError` refer to the built-in variables with those names. In the current implementation, the built-in variable :const:`__debug__` is `True` under normal circumstances, `False` when optimization is requested (command line option :option:`-O`). The current code generator emits no code for an assert statement when optimization is requested at compile time. Note that it is unnecessary to include the source code for the expression that failed in the error message; it will be displayed as part of the stack trace.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 403); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 403); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 403); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 403); *backlink*

Unknown interpreted text role "option".

Assignments to :const:`__debug__` are illegal. The value for the built-in variable is determined when the interpreter starts.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 412); *backlink*

Unknown interpreted text role "const".

The :keyword:`!pass` statement

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 418); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 421)

Unknown directive type "index".

```
.. index::
   statement: pass
   pair: null; operation
           pair: null; operation
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 426)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   pass_stmt: "pass"
```

:keyword:`pass` is a null operation --- when it is executed, nothing happens. It is useful as a placeholder when a statement is required syntactically, but no code needs to be executed, for example:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 429); *backlink*

Unknown interpreted text role "keyword".

```
def f(arg): pass    # a function that does nothing (yet)

class C: pass       # a class with no methods (yet)
```

The :keyword:`!del` statement

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 440); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 443)

Unknown directive type "index".

```
.. index::
   ! statement: del
   pair: deletion; target
   triple: deletion; target; list
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 448)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   del_stmt: "del" `target_list`
```

Deletion is recursively defined very similar to the way assignment is defined. Rather than spelling it out in full details, here are some hints.

Deletion of a target list recursively deletes each target, from left to right.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 456)

Unknown directive type "index".

```
.. index::
   statement: global
   pair: unbinding; name
```

Deletion of a name removes the binding of that name from the local or global namespace, depending on whether the name occurs in a :keyword:`global` statement in the same code block. If the name is unbound, a :exc:`NameError` exception will be raised.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 460); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 460); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 465)

Unknown directive type "index".

```
.. index:: pair: attribute; deletion
```

Deletion of attribute references, subscriptions and slicings is passed to the primary object involved; deletion of a slicing is in general equivalent to assignment of an empty slice of the right type (but even this is determined by the sliced object).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 472)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.2
   Previously it was illegal to delete a name from the local namespace if it
   occurs as a free variable in a nested block.
```

The :keyword:`!return` statement

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 479); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 482)

Unknown directive type "index".

```
.. index::
   ! statement: return
   pair: function; definition
   pair: class; definition
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 487)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   return_stmt: "return" [`expression_list`]
```

:keyword:`return` may only occur syntactically nested in a function definition, not within a nested class definition.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 490); *backlink*

Unknown interpreted text role "keyword".

If an expression list is present, it is evaluated, else `None` is substituted.

:keyword:`return` leaves the current function call with the expression list (or `None`) as return value.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 495); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 498)

Unknown directive type "index".

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

When :keyword:`return` passes control out of a :keyword:`try` statement with a :keyword:`finally` clause, that :keyword:`!finally` clause is executed before really leaving the function.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 500); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 500); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 500); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 500); *backlink*

Unknown interpreted text role "keyword".

In a generator function, the :keyword:`return` statement indicates that the generator is done and will cause :exc:`StopIteration` to be raised. The returned value (if any) is used as an argument to construct :exc:`StopIteration` and becomes the :attr:`StopIteration.value` attribute.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 504); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 504); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 504); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 504); *backlink*

Unknown interpreted text role "attr".

In an asynchronous generator function, an empty :keyword:`return` statement indicates that the asynchronous generator is done and will cause :exc:`StopAsyncIteration` to be raised. A non-empty :keyword:`!return` statement is a syntax error in an asynchronous generator function.

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

Unknown interpreted text role "keyword".

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

Unknown interpreted text role "exc".

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

Unknown interpreted text role "keyword".

The :keyword:`!yield` statement

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 516); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 519)

Unknown directive type "index".

```
.. index::
   statement: yield
   single: generator; function
   single: generator; iterator
   single: function; generator
   exception: StopIteration
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 526)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   yield_stmt: `yield_expression`
```

A :keyword:`yield` statement is semantically equivalent to a :ref:`yield expression <yieldexpr>`. The yield statement can be used to omit the parentheses that would otherwise be required in the equivalent yield expression statement. For example, the yield statements

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 529); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 529); *backlink*

Unknown interpreted text role "ref".

```
yield <expr>
yield from <expr>
```

are equivalent to the yield expression statements

```
(yield <expr>)
(yield from <expr>)
```

Yield expressions and statements are only used when defining a :term:`generator` function, and are only used in the body of the generator function. Using yield in a function definition is sufficient to cause that definition to create a generator function instead of a normal function.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 542); *backlink*

Unknown interpreted text role "term".

For full details of :keyword:`yield` semantics, refer to the :ref:`yieldexpr` section.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 547); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 547); *backlink*

Unknown interpreted text role "ref".

The :keyword:`!raise` statement

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 552); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 555)

Unknown directive type "index".

```
.. index::
   ! statement: raise
   single: exception
   pair: raising; exception
   single: __traceback__ (exception attribute)
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 561)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   raise_stmt: "raise" [`expression` ["from" `expression`]]
```

If no expressions are present, :keyword:`raise` re-raises the exception that is currently being handled, which is also known as the *active exception*. If there isn't currently an active exception, a :exc:`RuntimeError` exception is raised indicating that this is an error.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 564); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 564); *backlink*

Unknown interpreted text role "exc".

Otherwise, :keyword:`raise` evaluates the first expression as the exception object. It must be either a subclass or an instance of :class:`BaseException`. If it is a class, the exception instance will be obtained when needed by instantiating the class with no arguments.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 569); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 569); *backlink*

Unknown interpreted text role "class".

The :dfn:`type` of the exception is the exception instance's class, the :dfn:`value` is the instance itself.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 574); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 574); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 577)

Unknown directive type "index".

```
.. index:: object: traceback
```

A traceback object is normally created automatically when an exception is raised and attached to it as the :attr:`__traceback__` attribute, which is writable. You can create an exception and set your own traceback in one step using the :meth:`~BaseException.with_traceback` exception method (which returns the same exception instance, with its traceback set to its argument), like so:

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

Unknown interpreted text role "attr".

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

Unknown interpreted text role "meth".

```
raise Exception("foo occurred").with_traceback(tracebackobj)
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 587)

Unknown directive type "index".

```
.. index:: pair: exception; chaining
           __cause__ (exception attribute)
           __context__ (exception attribute)
```

The `from` clause is used for exception chaining: if given, the second *expression* must be another exception class or instance. If the second expression is an exception instance, it will be attached to the raised exception as the :attr:`__cause__` attribute (which is writable). If the expression is an exception class, the class will be instantiated and the resulting exception instance will be attached to the raised exception as the :attr:`__cause__` attribute. If the raised exception is not handled, both exceptions will be printed:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 591); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 591); *backlink*

Unknown interpreted text role "attr".

```
>>> try:
...     print(1 / 0)
... except Exception as exc:
...     raise RuntimeError("Something bad happened") from exc
...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
```

```
ZeroDivisionError: division by zero

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "<stdin>", line 4, in <module>
RuntimeError: Something bad happened
```

A similar mechanism works implicitly if a new exception is raised when an exception is already being handled. An exception may be handled when an :keyword:`except` or :keyword:`finally` clause, or a :keyword:`with` statement, is used. The previous exception is then attached as the new exception's :attr:`__context__` attribute:

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 615); *backlink*
>
> Unknown interpreted text role "keyword".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 615); *backlink*
>
> Unknown interpreted text role "keyword".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 615); *backlink*
>
> Unknown interpreted text role "keyword".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 615); *backlink*
>
> Unknown interpreted text role "attr".

```
>>> try:
...     print(1 / 0)
... except:
...     raise RuntimeError("Something bad happened")
...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
ZeroDivisionError: division by zero

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "<stdin>", line 4, in <module>
RuntimeError: Something bad happened
```

Exception chaining can be explicitly suppressed by specifying :const:`None` in the `from` clause:

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 636); *backlink*
>
> Unknown interpreted text role "const".

```
>>> try:
...     print(1 / 0)
... except:
...     raise RuntimeError("Something bad happened") from None
...
Traceback (most recent call last):
  File "<stdin>", line 4, in <module>
RuntimeError: Something bad happened
```

Additional information on exceptions can be found in section :ref:`exceptions`, and information about handling exceptions is in section :ref:`try`.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 648); *backlink*
>
> Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 648); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 651)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.3
    :const:`None` is now permitted as ``Y`` in ``raise X from Y``.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 654)

Unknown directive type "versionadded".

```
.. versionadded:: 3.3
    The ``__suppress_context__`` attribute to suppress automatic display of the
    exception context.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 658)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.11
    If the traceback of the active exception is modified in an :keyword:`except`
    clause, a subsequent ``raise`` statement re-raises the exception with the
    modified traceback. Previously, the exception was re-raised with the
    traceback it had when it was caught.
```

The :keyword:`!break` statement

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 666); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 669)

Unknown directive type "index".

```
.. index::
   ! statement: break
   statement: for
   statement: while
   pair: loop; statement
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 675)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   break_stmt: "break"
```

:keyword:`break` may only occur syntactically nested in a :keyword:`for` or :keyword:`while` loop, but not nested in a function or class definition within that loop.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 678); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 678); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 678); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 682)

Unknown directive type "index".

```
.. index:: keyword: else
           pair: loop control; target
```

It terminates the nearest enclosing loop, skipping the optional :keyword:`!else` clause if the loop has one.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 685); *backlink*

Unknown interpreted text role "keyword".

If a :keyword:`for` loop is terminated by :keyword:`break`, the loop control target keeps its current value.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 688); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 688); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 691)

Unknown directive type "index".

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

When :keyword:`break` passes control out of a :keyword:`try` statement with a :keyword:`finally` clause, that :keyword:`!finally` clause is executed before really leaving the loop.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 693); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 693); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 693); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 693); *backlink*

Unknown interpreted text role "keyword".

The :keyword:`!continue` statement

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 700); ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 703)

Unknown directive type "index".

```
.. index::
   ! statement: continue
   statement: for
   statement: while
   pair: loop; statement
   keyword: finally
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 710)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   continue_stmt: "continue"
```

:keyword:`continue` may only occur syntactically nested in a :keyword:`for` or :keyword:`while` loop, but not nested in a function or class definition within that loop. It continues with the next cycle of the nearest enclosing loop.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 713); ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 713); ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 713); ***backlink***

Unknown interpreted text role "keyword".

When :keyword:`continue` passes control out of a :keyword:`try` statement with a :keyword:`finally` clause, that :keyword:`!finally` clause is executed before really starting the next loop cycle.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 717); ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 717); ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 717); ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 717); *backlink*

Unknown interpreted text role "keyword".

The :keyword:`!import` statement

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 725); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 728)

Unknown directive type "index".

```
.. index::
   ! statement: import
   single: module; importing
   pair: name; binding
   keyword: from
   keyword: as
   exception: ImportError
   single: , (comma); import statement
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 737)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   import_stmt: "import" `module` ["as" `identifier`] ("," `module` ["as" `identifier`])*
              : | "from" `relative_module` "import" `identifier` ["as" `identifier`]
              : ("," `identifier` ["as" `identifier`])*
              : | "from" `relative_module` "import" "(" `identifier` ["as" `identifier`]
              : ("," `identifier` ["as" `identifier`])* [","] ")"
              : | "from" `relative_module` "import" "*"
   module: (`identifier` ".")* `identifier`
   relative_module: "."* `module` | "."+
```

The basic import statement (no :keyword:`from` clause) is executed in two steps:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 747); *backlink*

Unknown interpreted text role "keyword".

1. find a module, loading and initializing it if necessary
2. define a name or names in the local namespace for the scope where the :keyword:`import` statement occurs.

   System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 751); *backlink*

   Unknown interpreted text role "keyword".

When the statement contains multiple clauses (separated by commas) the two steps are carried out separately for each clause, just as though the clauses had been separated out into individual import statements.

The details of the first step, finding and loading modules are described in greater detail in the section on the :ref:`import system <importsystem>`, which also describes the various types of packages and modules that can be imported, as well as all the hooks that can be used to customize the import system. Note that failures in this step may indicate either that the module could not be located, *or* that an error occurred while initializing the module, which includes execution of the module's code.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 759); *backlink*

Unknown interpreted text role "ref".

If the requested module is retrieved successfully, it will be made available in the local namespace in one of three ways:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, **line 770**)

Unknown directive type "index".

```
.. index:: single: as; import statement
```

- If the module name is followed by :keyword:`!as`, then the name following :keyword:`!as` is bound directly to the imported module.

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

  Unknown interpreted text role "keyword".

- If no other name is specified, and the module being imported is a top level module, the module's name is bound in the local namespace as a reference to the imported module
- If the module being imported is *not* a top level module, then the name of the top level package that contains the module is bound in the local namespace as a reference to the top level package. The imported module must be accessed using its full qualified name rather than directly

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, **line 783**)

Unknown directive type "index".

```
.. index::
   pair: name; binding
   single: from; import statement
```

The :keyword:`from` form uses a slightly more complex process:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, **line 787**); *backlink*

Unknown interpreted text role "keyword".

1. find the module specified in the :keyword:`from` clause, loading and initializing it if necessary;

   System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, **line 789**); *backlink*

   Unknown interpreted text role "keyword".

2. for each of the identifiers specified in the :keyword:`import` clauses:

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

   Unknown interpreted text role "keyword".

   1. check if the imported module has an attribute by that name

2. if not, attempt to import a submodule with that name and then check the imported module again for that attribute
3. if the attribute is not found, :exc:`ImportError` is raised.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 796); *backlink*

Unknown interpreted text role "exc".

4. otherwise, a reference to that value is stored in the local namespace, using the name in the :keyword:`!as` clause if it is present, otherwise using the attribute name

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 797); *backlink*

Unknown interpreted text role "keyword".

Examples:

```
import foo                 # foo imported and bound locally
import foo.bar.baz         # foo.bar.baz imported, foo bound locally
import foo.bar.baz as fbb  # foo.bar.baz imported and bound as fbb
from foo.bar import baz    # foo.bar.baz imported and bound as baz
from foo import attr       # foo imported and foo.attr bound as attr
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 809)

Unknown directive type "index".

```
.. index:: single: * (asterisk); import statement
```

If the list of identifiers is replaced by a star (`'*'`), all public names defined in the module are bound in the local namespace for the scope where the :keyword:`import` statement occurs.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 811); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 815)

Unknown directive type "index".

```
.. index:: single: __all__ (optional module attribute)
```

The *public names* defined by a module are determined by checking the module's namespace for a variable named `__all__`; if defined, it must be a sequence of strings which are names defined or imported by that module. The names given in `__all__` are all considered public and are required to exist. If `__all__` is not defined, the set of public names includes all names found in the module's namespace which do not begin with an underscore character (`'_'`). `__all__` should contain the entire public API. It is intended to avoid accidentally exporting items that are not part of the API (such as library modules which were imported and used within the module).

The wild card form of import --- `from module import *` --- is only allowed at the module level. Attempting to use it in class or function definitions will raise a :exc:`SyntaxError`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 827); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference)simple_stmts.rst`, line 831)

Unknown directive type "index".

```
.. index::
    single: relative; import
```

When specifying what module to import you do not have to specify the absolute name of the module. When a module or package is contained within another package it is possible to make a relative import within the same top package without having to mention the package name. By using leading dots in the specified module or package after :keyword:`from` you can specify how high to traverse up the current package hierarchy without specifying exact names. One leading dot means the current package where the module making the import exists. Two dots means up one package level. Three dots is up two levels, etc. So if you execute `from . import mod` from a module in the `pkg` package then you will end up importing `pkg.mod`. If you execute `from ..subpkg2 import mod` from within `pkg.subpkg1` you will import `pkg.subpkg2.mod`. The specification for relative imports is contained in the :ref:`relativeimports` section.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_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) simple_stmts.rst`, line 834); *backlink*

Unknown interpreted text role "ref".

:func:`importlib.import_module` is provided to support applications that determine dynamically the modules to be loaded.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 848); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 851)

Unknown directive type "audit-event".

```
.. audit-event:: import module,filename,sys.path,sys.meta_path,sys.path_hooks import
```

Future statements

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 858)

Unknown directive type "index".

```
.. index::
   pair: future; statement
   single: __future__; future statement
```

A :dfn:`future statement` is a directive to the compiler that a particular module should be compiled using syntax or semantics that will be available in a specified future release of Python where the feature becomes standard.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 862); *backlink*

Unknown interpreted text role "dfn".

The future statement is intended to ease migration to future versions of Python that introduce incompatible changes to the language. It allows use of the new features on a per-module basis before the release in which the feature becomes standard.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 871)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   future_stmt: "from" "__future__" "import" `feature` ["as" `identifier`]
              : ("," `feature` ["as" `identifier`])*
```

```
                  : | "from" "__future__" "import" "(" `feature` ["as" `identifier`]
                  : ("," `feature` ["as" `identifier`])* [","] ")"
    feature: `identifier`
```

A future statement must appear near the top of the module. The only lines that can appear before a future statement are:

- the module docstring (if any),
- comments,
- blank lines, and
- other future statements.

The only feature that requires using the future statement is `annotations` (see PEP 563).

All historical features enabled by the future statement are still recognized by Python 3. The list includes `absolute_import`, `division`, `generators`, `generator_stop`, `unicode_literals`, `print_function`, `nested_scopes` and `with_statement`. They are all redundant because they are always enabled, and only kept for backwards compatibility.

A future statement is recognized and treated specially at compile time: Changes to the semantics of core constructs are often implemented by generating different code. It may even be the case that a new feature introduces new incompatible syntax (such as a new reserved word), in which case the compiler may need to parse the module differently. Such decisions cannot be pushed off until runtime.

For any given release, the compiler knows which feature names have been defined, and raises a compile-time error if a future statement contains a feature not known to it.

The direct runtime semantics are the same as for any import statement: there is a standard module :mod:`__future__`, described later, and it will be imported in the usual way at the time the future statement is executed.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 907); *backlink*

Unknown interpreted text role "mod".

The interesting runtime semantics depend on the specific feature enabled by the future statement.

Note that there is nothing special about the statement:

```
import __future__ [as name]
```

That is not a future statement; it's an ordinary import statement with no special semantics or syntax restrictions.

Code compiled by calls to the built-in functions :func:`exec` and :func:`compile` that occur in a module :mod:`M` containing a future statement will, by default, use the new syntax or semantics associated with the future statement. This can be controlled by optional arguments to :func:`compile` --- see the documentation of that function for details.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 921); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 921); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 921); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 921); *backlink*

Unknown interpreted text role "func".

A future statement typed at an interactive interpreter prompt will take effect for the rest of the interpreter session. If an interpreter is started with the :option:`-i` option, is passed a script name to execute, and the script includes a future statement, it will be in effect in the interactive session started after the script is executed.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst, line 927); *backlink*

Unknown interpreted text role "option".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst, line 933)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`236` - Back to the __future__
      The original proposal for the __future__ mechanism.
```

The :keyword:`!global` statement

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst, line 941); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst, line 944)

Unknown directive type "index".

```
.. index::
   ! statement: global
   triple: global; name; binding
   single: , (comma); identifier list
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst, line 949)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   global_stmt: "global" `identifier` ("," `identifier`)*
```

The :keyword:`global` statement is a declaration which holds for the entire current code block. It means that the listed identifiers are to be interpreted as globals. It would be impossible to assign to a global variable without :keyword:`!global`, although free variables may refer to globals without being declared global.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst, line 952); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst, line 952); *backlink*

Unknown interpreted text role "keyword".

Names listed in a :keyword:`global` statement must not be used in the same code block textually preceding that :keyword:`!global` statement.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst, line 958); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst, line 958); *backlink*

Unknown interpreted text role "keyword".

Names listed in a :keyword:`global` statement must not be defined as formal parameters, or as targets in :keyword:`with` statements or :keyword:`except` clauses, or in a :keyword:`for` target list, :keyword:`class` definition, function definition, :keyword:`import` statement, or variable annotation.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 961); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 961); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 961); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 961); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 961); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 961); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 966)

Unknown directive type "impl-detail".

```
.. impl-detail::

   The current implementation does not enforce some of these restrictions, but
   programs should not abuse this freedom, as future implementations may enforce
   them or silently change the meaning of the program.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 972)

Unknown directive type "index".

```
.. index::
   builtin: exec
   builtin: eval
   builtin: compile
```

Programmer's note: :keyword:`global` is a directive to the parser. It applies only to code parsed at the same time as the :keyword:`!global` statement. In particular, a :keyword:`!global` statement contained in a string or code object supplied to the built-in :func:`exec` function does not affect the code block *containing* the function call, and code contained in such a string is unaffected by :keyword:`!global` statements in the code containing the function call. The same applies to the :func:`eval` and :func:`compile` functions.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 977); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst`, line 977); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst`, line 977); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst`, line 977); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst`, line 977); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst`, line 977); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst`, line 977); *backlink*

Unknown interpreted text role "func".

The :keyword:`!nonlocal` statement

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst`, line 988); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst`, line 991)

Unknown directive type "index".

```
.. index:: statement: nonlocal
   single: , (comma); identifier list
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst`, line 994)

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   nonlocal_stmt: "nonlocal" `identifier` ("," `identifier`)*
```

The :keyword:`nonlocal` statement causes the listed identifiers to refer to previously bound variables in the nearest enclosing scope excluding globals. This is important because the default behavior for binding is to search the local namespace first. The statement allows encapsulated code to rebind variables outside of the local scope besides the global (module) scope.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main)(Doc)(reference)simple_stmts.rst`, line 1001); *backlink*

Unknown interpreted text role "keyword".

Names listed in a :keyword:`nonlocal` statement, unlike those listed in a :keyword:`global` statement, must refer to pre-existing bindings in an enclosing scope (the scope in which a new binding should be created cannot be determined unambiguously).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 1011); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 1011); *backlink*

Unknown interpreted text role "keyword".

Names listed in a :keyword:`nonlocal` statement must not collide with pre-existing bindings in the local scope.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 1016); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\(cpython-main) (Doc) (reference) simple_stmts.rst`, line 1019)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`3104` - Access to Names in Outer Scopes
      The specification for the :keyword:`nonlocal` statement.
```