:mod:`ast` --- Abstract Syntax Trees

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 4)

Unknown directive type "module".

```
.. module:: ast
   :synopsis: Abstract Syntax Tree classes and manipulation.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 7)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Martin v. LÃ¶wis <martin@v.loewis.de>
```

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

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Georg Brandl <georg@python.org>
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 10)

Unknown directive type "testsetup".

```
.. testsetup::

    import ast
```

Source code: :source:`Lib/ast.py`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 14); *backlink*

Unknown interpreted text role "source".

The :mod:`ast` module helps Python applications to process trees of the Python abstract syntax grammar. The abstract syntax itself might change with each Python release; this module helps to find out programmatically what the current grammar looks like.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 18); *backlink*

Unknown interpreted text role "mod".

An abstract syntax tree can be generated by passing :data:`ast.PyCF_ONLY_AST` as a flag to the :func:`compile` built-in function, or using the :func:`parse` helper provided in this module. The result will be a tree of objects whose classes all inherit from :class:`ast.AST`. An abstract syntax tree can be compiled into a Python code object using the built-in :func:`compile` function.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 23); *backlink*

Unknown interpreted text role "data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 23); *backlink*

Unknown interpreted text role "func".

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

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 23); backlink

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 23); backlink

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 23); backlink

Unknown interpreted text role "func".

Abstract Grammar

The abstract grammar is currently defined as follows:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 37)

Unknown directive type "literalinclude".

```
.. literalinclude:: ../../Parser/Python.asdl
   :language: asdl
```

Node classes

This is the base of all AST node classes. The actual node classes are derived from the :file:`Parser/Python.asdl` file, which is reproduced :ref:`below <abstract-grammar>`. They are defined in the :mod:`_ast` C module and re-exported in :mod:`ast`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 46); backlink

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 46); backlink

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 46); backlink

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 46); backlink

Unknown interpreted text role "mod".

There is one class defined for each left-hand side symbol in the abstract grammar (for example, :class:`ast.stmt` or :class:`ast.expr`). In addition, there is one class defined for each constructor on the right-hand side; these classes inherit from the classes for the left-hand side trees. For example, :class:`ast.BinOp` inherits from :class:`ast.expr`. For production rules with alternatives (aka "sums"), the left-hand side class is abstract: only instances of specific constructor nodes are ever created.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 51); backlink

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 51); backlink

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 51); backlink

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 51); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 59)

Unknown directive type "index".

```
.. index:: single: ? (question mark); in AST grammar
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 60)

Unknown directive type "index".

```
.. index:: single: * (asterisk); in AST grammar
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 62)

Unknown directive type "attribute".

```
.. attribute:: _fields

   Each concrete class has an attribute :attr:`_fields` which gives the names
   of all child nodes.

   Each instance of a concrete class has one attribute for each child node,
   of the type as defined in the grammar.  For example, :class:`ast.BinOp`
   instances have an attribute :attr:`left` of type :class:`ast.expr`.

   If these attributes are marked as optional in the grammar (using a
   question mark), the value might be ``None``.  If the attributes can have
   zero-or-more values (marked with an asterisk), the values are represented
   as Python lists.  All possible attributes must be present and have valid
   values when compiling an AST with :func:`compile`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 77)

Unknown directive type "attribute".

```
.. attribute:: lineno
               col_offset
               end_lineno
               end_col_offset

   Instances of :class:`ast.expr` and :class:`ast.stmt` subclasses have
   :attr:`lineno`, :attr:`col_offset`, :attr:`end_lineno`, and
   :attr:`end_col_offset` attributes.  The :attr:`lineno` and :attr:`end_lineno`
   are the first and last line numbers of source text span (1-indexed so the
   first line is line 1) and the :attr:`col_offset` and :attr:`end_col_offset`
   are the corresponding UTF-8 byte offsets of the first and last tokens that
   generated the node. The UTF-8 offset is recorded because the parser uses
   UTF-8 internally.

   Note that the end positions are not required by the compiler and are
   therefore optional. The end offset is *after* the last symbol, for example
   one can get the source segment of a one-line expression node using
   ``source_line[node.col_offset : node.end_col_offset]``.
```

The constructor of a class :class:`ast.T` parses its arguments as follows:

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

Unknown interpreted text role "class".

- If there are positional arguments, there must be as many as there are items in :attr:`T._fields`; they will be assigned as attributes of these names.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 98); *backlink*

Unknown interpreted text role "attr".

- If there are keyword arguments, they will set the attributes of the same names to the given values.

For example, to create and populate an :class:`ast.UnaryOp` node, you could use

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 103); *backlink*

Unknown interpreted text role "class".

```
node = ast.UnaryOp()
node.op = ast.USub()
node.operand = ast.Constant()
node.operand.value = 5
node.operand.lineno = 0
node.operand.col_offset = 0
node.lineno = 0
node.col_offset = 0
```

or the more compact

```
node = ast.UnaryOp(ast.USub(), ast.Constant(5, lineno=0, col_offset=0),
                   lineno=0, col_offset=0)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 120)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.8

   Class :class:`ast.Constant` is now used for all constants.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 124)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.9

   Simple indices are represented by their value, extended slices are
   represented as tuples.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 129)

Unknown directive type "deprecated".

```
.. deprecated:: 3.8

   Old classes :class:`ast.Num`, :class:`ast.Str`, :class:`ast.Bytes`,
   :class:`ast.NameConstant` and :class:`ast.Ellipsis` are still available,
   but they will be removed in future Python releases.  In the meantime,
   instantiating them will return an instance of a different class.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 136)

Unknown directive type "deprecated".

```
.. deprecated:: 3.9

   Old classes :class:`ast.Index` and :class:`ast.ExtSlice` are still
   available, but they will be removed in future Python releases.
   In the meantime, instantiating them will return an instance of
   a different class.
```

Note

The descriptions of the specific node classes displayed here were initially adapted from the fantastic Green Tree Snakes project and all its contributors.

Literals

A constant value. The `value` attribute of the `Constant` literal contains the Python object it represents. The values represented can be simple types such as a number, string or `None`, but also immutable container types (tuples and frozensets) if all of their elements

are constant.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 159)

Unknown directive type "doctest".

```
.. doctest::

     >>> print(ast.dump(ast.parse('123', mode='eval'), indent=4))
     Expression(
         body=Constant(value=123))
```

Node representing a single formatting field in an f-string. If the string contains a single formatting field and nothing else the node can be isolated otherwise it appears in :class:`JoinedStr`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 168); *backlink*

Unknown interpreted text role "class".

- `value` is any expression node (such as a literal, a variable, or a function call).
- `conversion` is an integer:
  - -1: no formatting
  - 115: `!s` string formatting
  - 114: `!r` repr formatting
  - 97: `!a` ascii formatting
- `format_spec` is a :class:`JoinedStr` node representing the formatting of the value, or `None` if no format was specified. Both `conversion` and `format_spec` can be set at the same time.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 181); *backlink*

Unknown interpreted text role "class".

An f-string, comprising a series of :class:`FormattedValue` and :class:`Constant` nodes.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 188); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 188); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 191)

Unknown directive type "doctest".

```
.. doctest::

     >>> print(ast.dump(ast.parse('f"sin({a}) is {sin(a):.3}"', mode='eval'), indent=4))
     Expression(
         body=JoinedStr(
             values=[
                 Constant(value='sin('),
                 FormattedValue(
                     value=Name(id='a', ctx=Load()),
                     conversion=-1),
                 Constant(value=') is '),
                 FormattedValue(
                     value=Call(
                         func=Name(id='sin', ctx=Load()),
                         args=[
                             Name(id='a', ctx=Load())],
                         keywords=[]),
                     conversion=-1,
                     format_spec=JoinedStr(
                         values=[
                             Constant(value='.3')]))])))
```

A list or tuple. `elts` holds a list of nodes representing the elements. `ctx` is :class:`Store` if the container is an assignment target (i.e. `(x,y)=something`), and :class:`Load` otherwise.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 217); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 217); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 221)

Unknown directive type "doctest".

```
.. doctest::

     >>> print(ast.dump(ast.parse('[1, 2, 3]', mode='eval'), indent=4))
     Expression(
         body=List(
             elts=[
                 Constant(value=1),
                 Constant(value=2),
                 Constant(value=3)],
             ctx=Load()))
     >>> print(ast.dump(ast.parse('(1, 2, 3)', mode='eval'), indent=4))
     Expression(
         body=Tuple(
             elts=[
                 Constant(value=1),
                 Constant(value=2),
                 Constant(value=3)],
             ctx=Load()))
```

A set. `elts` holds a list of nodes representing the set's elements.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 245)

Unknown directive type "doctest".

```
.. doctest::

     >>> print(ast.dump(ast.parse('{1, 2, 3}', mode='eval'), indent=4))
     Expression(
         body=Set(
             elts=[
                 Constant(value=1),
                 Constant(value=2),
                 Constant(value=3)]))
```

A dictionary. `keys` and `values` hold lists of nodes representing the keys and the values respectively, in matching order (what would be returned when calling `dictionary.keys()` and `dictionary.values()`).

When doing dictionary unpacking using dictionary literals the expression to be expanded goes in the `values` list, with a `None` at the corresponding position in `keys`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 266)

Unknown directive type "doctest".

```
.. doctest::

     >>> print(ast.dump(ast.parse('{"a":1, **d}', mode='eval'), indent=4))
     Expression(
         body=Dict(
             keys=[
                 Constant(value='a'),
                 None],
             values=[
                 Constant(value=1),
                 Name(id='d', ctx=Load())]))
```

Variables

A variable name. `id` holds the name as a string, and `ctx` is one of the following types.

Variable references can be used to load the value of a variable, to assign a new value to it, or to delete it. Variable references are given a context to distinguish these cases.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 296)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('a'), indent=4))
    Module(
        body=[
            Expr(
                value=Name(id='a', ctx=Load()))],
        type_ignores=[])

    >>> print(ast.dump(ast.parse('a = 1'), indent=4))
    Module(
        body=[
            Assign(
                targets=[
                    Name(id='a', ctx=Store())],
                value=Constant(value=1))],
        type_ignores=[])

    >>> print(ast.dump(ast.parse('del a'), indent=4))
    Module(
        body=[
            Delete(
                targets=[
                    Name(id='a', ctx=Del())])],
        type_ignores=[])
```

A `*var` variable reference. `value` holds the variable, typically a :class:`Name` node. This type must be used when building a :class:`Call` node with `*args`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 325); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 325); *backlink*

Unknown interpreted text role "class".

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

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('a, *b = it'), indent=4))
    Module(
        body=[
            Assign(
                targets=[
                    Tuple(
                        elts=[
                            Name(id='a', ctx=Store()),
                            Starred(
                                value=Name(id='b', ctx=Store()),
                                ctx=Store())],
                        ctx=Store())],
                value=Name(id='it', ctx=Load()))],
        type_ignores=[])
```

Expressions

When an expression, such as a function call, appears as a statement by itself with its return value not used or stored, it is wrapped in this container. `value` holds one of the other nodes in this section, a :class:`Constant`, a :class:`Name`, a :class:`Lambda`, a :class:`Yield` or :class:`YieldFrom` node.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 352); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 352); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 352); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 352); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 352); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 357)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('-a'), indent=4))
    Module(
        body=[
            Expr(
                value=UnaryOp(
                    op=USub(),
                    operand=Name(id='a', ctx=Load())))],
        type_ignores=[])
```

A unary operation. `op` is the operator, and `operand` any expression node.

Unary operator tokens. :class:`Not` is the `not` keyword, :class:`Invert` is the `~` operator.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 380); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 380); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 383)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('not x', mode='eval'), indent=4))
    Expression(
        body=UnaryOp(
            op=Not(),
            operand=Name(id='x', ctx=Load())))
```

A binary operation (like addition or division). `op` is the operator, and `left` and `right` are any expression nodes.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 397)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('x + y', mode='eval'), indent=4))
    Expression(
```

```
            body=BinOp(
                left=Name(id='x', ctx=Load()),
                op=Add(),
                right=Name(id='y', ctx=Load())))
```

Binary operator tokens.

A boolean operation, 'or' or 'and'. `op` is :class:`Or` or :class:`And`. `values` are the values involved. Consecutive operations with the same operator, such as `a or b or c`, are collapsed into one node with several values.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 426); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 426); *backlink*

Unknown interpreted text role "class".

This doesn't include `not`, which is a :class:`UnaryOp`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 431); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 433)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('x or y', mode='eval'), indent=4))
    Expression(
        body=BoolOp(
            op=Or(),
            values=[
                Name(id='x', ctx=Load()),
                Name(id='y', ctx=Load())]))
```

Boolean operator tokens.

A comparison of two or more values. `left` is the first value in the comparison, `ops` the list of operators, and `comparators` the list of values after the first element in the comparison.

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

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('1 <= a < 10', mode='eval'), indent=4))
    Expression(
        body=Compare(
            left=Constant(value=1),
            ops=[
                LtE(),
                Lt()],
            comparators=[
                Name(id='a', ctx=Load()),
                Constant(value=10)]))
```

Comparison operator tokens.

A function call. `func` is the function, which will often be a :class:`Name` or :class:`Attribute` object. Of the arguments:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 486); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 486); *backlink*

Unknown interpreted text role "class".

- `args` holds a list of the arguments passed by position.
- `keywords` holds a list of :class:`keyword` objects representing arguments passed by keyword.

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

Unknown interpreted text role "class".

When creating a `Call` node, `args` and `keywords` are required, but they can be empty lists. `starargs` and `kwargs` are optional.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 496)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('func(a, b=c, *d, **e)', mode='eval'), indent=4))
    Expression(
        body=Call(
            func=Name(id='func', ctx=Load()),
            args=[
                Name(id='a', ctx=Load()),
                Starred(
                    value=Name(id='d', ctx=Load()),
                    ctx=Load())],
            keywords=[
                keyword(
                    arg='b',
                    value=Name(id='c', ctx=Load())),
                keyword(
                    value=Name(id='e', ctx=Load()))]))
```

A keyword argument to a function call or class definition. `arg` is a raw string of the parameter name, `value` is a node to pass in.

An expression such as `a if b else c`. Each field holds a single node, so in the following example, all three are :class:`Name` nodes.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 523); *backlink*

Unknown interpreted text role "class".

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

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('a if b else c', mode='eval'), indent=4))
    Expression(
        body=IfExp(
            test=Name(id='b', ctx=Load()),
            body=Name(id='a', ctx=Load()),
            orelse=Name(id='c', ctx=Load())))
```

Attribute access, e.g. `d.keys`. `value` is a node, typically a :class:`Name`. `attr` is a bare string giving the name of the attribute, and `ctx` is :class:`Load`, :class:`Store` or :class:`Del` according to how the attribute is acted on.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 538); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 538); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 538); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 538); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 543)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('snake.colour', mode='eval'), indent=4))
    Expression(
        body=Attribute(
            value=Name(id='snake', ctx=Load()),
            attr='colour',
            ctx=Load()))
```

A named expression. This AST node is produced by the assignment expressions operator (also known as the walrus operator). As opposed to the :class:`Assign` node in which the first argument can be multiple nodes, in this case both `target` and `value` must be single nodes.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 555); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 560)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('(x := 4)', mode='eval'), indent=4))
    Expression(
        body=NamedExpr(
            target=Name(id='x', ctx=Store()),
            value=Constant(value=4)))
```

Subscripting

A subscript, such as `l[1]`. `value` is the subscripted object (usually sequence or mapping). `slice` is an index, slice or key. It can be a :class:`Tuple` and contain a :class:`Slice`. `ctx` is :class:`Load`, :class:`Store` or :class:`Del` according to the action performed with the subscript.

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 580)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('l[1:2, 3]', mode='eval'), indent=4))
    Expression(
        body=Subscript(
            value=Name(id='l', ctx=Load()),
            slice=Tuple(
                elts=[
                    Slice(
                        lower=Constant(value=1),
                        upper=Constant(value=2)),
                    Constant(value=3)],
                ctx=Load()),
            ctx=Load()))
```

Regular slicing (on the form `lower:upper` or `lower:upper:step`). Can occur only inside the *slice* field of :class:`Subscript`, either directly or as an element of :class:`Tuple`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 598); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 598); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 602)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('l[1:2]', mode='eval'), indent=4))
    Expression(
        body=Subscript(
            value=Name(id='l', ctx=Load()),
            slice=Slice(
                lower=Constant(value=1),
                upper=Constant(value=2)),
            ctx=Load()))
```

Comprehensions

List and set comprehensions, generator expressions, and dictionary comprehensions. `elt` (or `key` and `value`) is a single node representing the part that will be evaluated for each item.

`generators` is a list of :class:`comprehension` nodes.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 626); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 628)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('[x for x in numbers]', mode='eval'), indent=4))
    Expression(
        body=ListComp(
            elt=Name(id='x', ctx=Load()),
            generators=[
                comprehension(
                    target=Name(id='x', ctx=Store()),
                    iter=Name(id='numbers', ctx=Load()),
                    ifs=[],
                    is_async=0)]))
    >>> print(ast.dump(ast.parse('{x: x**2 for x in numbers}', mode='eval'), indent=4))
```

```
        Expression(
            body=DictComp(
                key=Name(id='x', ctx=Load()),
                value=BinOp(
                    left=Name(id='x', ctx=Load()),
                    op=Pow(),
                    right=Constant(value=2)),
                generators=[
                    comprehension(
                        target=Name(id='x', ctx=Store()),
                        iter=Name(id='numbers', ctx=Load()),
                        ifs=[],
                        is_async=0)]))
        >>> print(ast.dump(ast.parse('{x for x in numbers}', mode='eval'), indent=4))
        Expression(
            body=SetComp(
                elt=Name(id='x', ctx=Load()),
                generators=[
                    comprehension(
                        target=Name(id='x', ctx=Store()),
                        iter=Name(id='numbers', ctx=Load()),
                        ifs=[],
                        is_async=0)]))
```

One `for` clause in a comprehension. `target` is the reference to use for each element - typically a :class:`Name` or :class:`Tuple` node. `iter` is the object to iterate over. `ifs` is a list of test expressions: each `for` clause can have multiple `ifs`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 668); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 668); *backlink*

Unknown interpreted text role "class".

`is_async` indicates a comprehension is asynchronous (using an `async for` instead of `for`). The value is an integer (0 or 1).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 676)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('[ord(c) for line in file for c in line]', mode='eval'),
    ...                indent=4)) # Multiple comprehensions in one.
    Expression(
        body=ListComp(
            elt=Call(
                func=Name(id='ord', ctx=Load()),
                args=[
                    Name(id='c', ctx=Load())],
                keywords=[]),
            generators=[
                comprehension(
                    target=Name(id='line', ctx=Store()),
                    iter=Name(id='file', ctx=Load()),
                    ifs=[],
                    is_async=0),
                comprehension(
                    target=Name(id='c', ctx=Store()),
                    iter=Name(id='line', ctx=Load()),
                    ifs=[],
                    is_async=0)]))

    >>> print(ast.dump(ast.parse('(n**2 for n in it if n>5 if n<10)', mode='eval'),
    ...                indent=4)) # generator comprehension
    Expression(
        body=GeneratorExp(
            elt=BinOp(
                left=Name(id='n', ctx=Load()),
                op=Pow(),
                right=Constant(value=2)),
            generators=[
                comprehension(
                    target=Name(id='n', ctx=Store()),
                    iter=Name(id='it', ctx=Load()),
                    ifs=[
                        Compare(
                            left=Name(id='n', ctx=Load()),
                            ops=[
                                Gt()],
```

```
                            comparators=[
                                Constant(value=5)]),
                        Compare(
                            left=Name(id='n', ctx=Load()),
                            ops=[
                                Lt()],
                            comparators=[
                                Constant(value=10)])],
                    is_async=0)]))

    >>> print(ast.dump(ast.parse('[i async for i in soc]', mode='eval'),
    ...                indent=4)) # Async comprehension
    Expression(
        body=ListComp(
            elt=Name(id='i', ctx=Load()),
            generators=[
                comprehension(
                    target=Name(id='i', ctx=Store()),
                    iter=Name(id='soc', ctx=Load()),
                    ifs=[],
                    is_async=1)]))
```

Statements

An assignment. `targets` is a list of nodes, and `value` is a single node.

Multiple nodes in `targets` represents assigning the same value to each. Unpacking is represented by putting a :class:`Tuple` or :class:`List` within `targets`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 745); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 745); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 749)

Unknown directive type "attribute".

```
.. attribute:: type_comment

    ``type_comment`` is an optional string with the type annotation as a comment.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 753)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('a = b = 1'), indent=4)) # Multiple assignment
    Module(
        body=[
            Assign(
                targets=[
                    Name(id='a', ctx=Store()),
                    Name(id='b', ctx=Store())],
                value=Constant(value=1))],
        type_ignores=[])

    >>> print(ast.dump(ast.parse('a,b = c'), indent=4)) # Unpacking
    Module(
        body=[
            Assign(
                targets=[
                    Tuple(
                        elts=[
                            Name(id='a', ctx=Store()),
                            Name(id='b', ctx=Store())],
                        ctx=Store())],
                value=Name(id='c', ctx=Load()))],
        type_ignores=[])
```

An assignment with a type annotation. `target` is a single node and can be a :class:`Name`, a :class:`Attribute` or a :class:`Subscript`. `annotation` is the annotation, such as a :class:`Constant` or :class:`Name` node. `value` is a single optional node. `simple` is a

boolean integer set to True for a :class:`Name` node in `target` that do not appear in between parenthesis and are hence pure names and not expressions.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 781); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 781); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 781); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 781); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 781); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 781); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 788)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('c: int'), indent=4))
    Module(
        body=[
            AnnAssign(
                target=Name(id='c', ctx=Store()),
                annotation=Name(id='int', ctx=Load()),
                simple=1)],
        type_ignores=[])

    >>> print(ast.dump(ast.parse('(a): int = 1'), indent=4)) # Annotation with parenthesis
    Module(
        body=[
            AnnAssign(
                target=Name(id='a', ctx=Store()),
                annotation=Name(id='int', ctx=Load()),
                value=Constant(value=1),
                simple=0)],
        type_ignores=[])

    >>> print(ast.dump(ast.parse('a.b: int'), indent=4)) # Attribute annotation
    Module(
        body=[
            AnnAssign(
                target=Attribute(
                    value=Name(id='a', ctx=Load()),
                    attr='b',
                    ctx=Store()),
                annotation=Name(id='int', ctx=Load()),
                simple=0)],
        type_ignores=[])

    >>> print(ast.dump(ast.parse('a[1]: int'), indent=4)) # Subscript annotation
    Module(
        body=[
            AnnAssign(
                target=Subscript(
                    value=Name(id='a', ctx=Load()),
                    slice=Constant(value=1),
                    ctx=Store()),
                annotation=Name(id='int', ctx=Load()),
```

```
                simple=0)],
        type_ignores=[])
```

Augmented assignment, such as `a += 1`. In the following example, `target` is a :class:`Name` node for `x` (with the :class:`Store` context), `op` is :class:`Add`, and `value` is a :class:`Constant` with value for 1.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 836); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 836); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 836); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 836); *backlink*

Unknown interpreted text role "class".

The `target` attribute cannot be of class :class:`Tuple` or :class:`List`, unlike the targets of :class:`Assign`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 841); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 841); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 841); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 844)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('x += 2'), indent=4))
    Module(
        body=[
            AugAssign(
                target=Name(id='x', ctx=Store()),
                op=Add(),
                value=Constant(value=2))],
        type_ignores=[])
```

A `raise` statement. `exc` is the exception object to be raised, normally a :class:`Call` or :class:`Name`, or `None` for a standalone `raise`. `cause` is the optional part for `y` in `raise x from y`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 858); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 858); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 862)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('raise x from y'), indent=4))
    Module(
        body=[
            Raise(
                exc=Name(id='x', ctx=Load()),
                cause=Name(id='y', ctx=Load()))],
        type_ignores=[])
```

An assertion. `test` holds the condition, such as a :class:`Compare` node. `msg` holds the failure message.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 875); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 878)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('assert x,y'), indent=4))
    Module(
        body=[
            Assert(
                test=Name(id='x', ctx=Load()),
                msg=Name(id='y', ctx=Load()))],
        type_ignores=[])
```

Represents a `del` statement. `targets` is a list of nodes, such as :class:`Name`, :class:`Attribute` or :class:`Subscript` nodes.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 891); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 891); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 891); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 894)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('del x,y,z'), indent=4))
    Module(
        body=[
            Delete(
                targets=[
                    Name(id='x', ctx=Del()),
                    Name(id='y', ctx=Del()),
                    Name(id='z', ctx=Del())])],
        type_ignores=[])
```

A `pass` statement.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 911)

Unknown directive type "doctest".

```
.. doctest::

     >>> print(ast.dump(ast.parse('pass'), indent=4))
     Module(
         body=[
             Pass()],
         type_ignores=[])
```

Other statements which are only applicable inside functions or loops are described in other sections.

Imports

An import statement. `names` is a list of :class:`alias` nodes.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 928); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 930)

Unknown directive type "doctest".

```
.. doctest::

     >>> print(ast.dump(ast.parse('import x,y,z'), indent=4))
     Module(
         body=[
             Import(
                 names=[
                     alias(name='x'),
                     alias(name='y'),
                     alias(name='z')])],
         type_ignores=[])
```

Represents `from x import y`. `module` is a raw string of the 'from' name, without any leading dots, or `None` for statements such as `from . import foo`. `level` is an integer holding the level of the relative import (0 means absolute import).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 950)

Unknown directive type "doctest".

```
.. doctest::

     >>> print(ast.dump(ast.parse('from y import x,y,z'), indent=4))
     Module(
         body=[
             ImportFrom(
                 module='y',
                 names=[
                     alias(name='x'),
                     alias(name='y'),
                     alias(name='z')],
                 level=0)],
         type_ignores=[])
```

Both parameters are raw strings of the names. `asname` can be `None` if the regular name is to be used.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 970)

Unknown directive type "doctest".

```
.. doctest::

     >>> print(ast.dump(ast.parse('from ..foo.bar import a as b, c'), indent=4))
     Module(
         body=[
             ImportFrom(
                 module='foo.bar',
                 names=[
                     alias(name='a', asname='b'),
                     alias(name='c')],
                 level=2)],
         type_ignores=[])
```

Control flow

Note

Optional clauses such as `else` are stored as an empty list if they're not present.

An `if` statement. `test` holds a single node, such as a :class:`Compare` node. `body` and `orelse` each hold a list of nodes.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 992); *backlink*

Unknown interpreted text role "class".

`elif` clauses don't have a special representation in the AST, but rather appear as extra :class:`If` nodes within the `orelse` section of the previous one.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 995); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 999)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse("""
    ... if x:
    ...     ...
    ... elif y:
    ...     ...
    ... else:
    ...     ...
    ... """), indent=4))
    Module(
        body=[
            If(
                test=Name(id='x', ctx=Load()),
                body=[
                    Expr(
                        value=Constant(value=Ellipsis))],
                orelse=[
                    If(
                        test=Name(id='y', ctx=Load()),
                        body=[
                            Expr(
                                value=Constant(value=Ellipsis))],
                        orelse=[
                            Expr(
                                value=Constant(value=Ellipsis))])])],
        type_ignores=[])
```

A `for` loop. `target` holds the variable(s) the loop assigns to, as a single :class:`Name`, :class:`Tuple` or :class:`List` node. `iter` holds the item to be looped over, again as a single node. `body` and `orelse` contain lists of nodes to execute. Those in `orelse` are executed if the loop finishes normally, rather than via a `break` statement.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1030); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1030); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1030); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1036)

Unknown directive type "attribute".

```
.. attribute:: type_comment
```

```
    ``type_comment`` is an optional string with the type annotation as a comment.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1040)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse("""
    ... for x in y:
    ...     ...
    ... else:
    ...     ...
    ... """), indent=4))
    Module(
        body=[
            For(
                target=Name(id='x', ctx=Store()),
                iter=Name(id='y', ctx=Load()),
                body=[
                    Expr(
                        value=Constant(value=Ellipsis))],
                orelse=[
                    Expr(
                        value=Constant(value=Ellipsis))])],
        type_ignores=[])
```

A `while` loop. `test` holds the condition, such as a :class:`Compare` node.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1064); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1067)

Unknown directive type "doctest".

```
.. doctest::

    >> print(ast.dump(ast.parse("""
    ... while x:
    ...     ...
    ... else:
    ...     ...
    ... """), indent=4))
    Module(
        body=[
            While(
                test=Name(id='x', ctx=Load()),
                body=[
                    Expr(
                        value=Constant(value=Ellipsis))],
                orelse=[
                    Expr(
                        value=Constant(value=Ellipsis))])],
        type_ignores=[])
```

The `break` and `continue` statements.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1093)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse("""\
    ... for a in b:
    ...     if a > 5:
    ...         break
    ...     else:
    ...         continue
    ...
    ... """), indent=4))
    Module(
        body=[
            For(
                target=Name(id='a', ctx=Store()),
```

```
                    iter=Name(id='b', ctx=Load()),
                    body=[
                        If(
                            test=Compare(
                                left=Name(id='a', ctx=Load()),
                                ops=[
                                    Gt()],
                                comparators=[
                                    Constant(value=5)]),
                            body=[
                                Break()],
                            orelse=[
                                Continue()])],
                    orelse=[])],
            type_ignores=[])
```

`try` blocks. All attributes are list of nodes to execute, except for `handlers`, which is a list of :class:`ExceptHandler` nodes.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1126); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1129)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse("""
    ... try:
    ...     ...
    ... except Exception:
    ...     ...
    ... except OtherException as e:
    ...     ...
    ... else:
    ...     ...
    ... finally:
    ...     ...
    ... """), indent=4))
    Module(
        body=[
            Try(
                body=[
                    Expr(
                        value=Constant(value=Ellipsis))],
                handlers=[
                    ExceptHandler(
                        type=Name(id='Exception', ctx=Load()),
                        body=[
                            Expr(
                                value=Constant(value=Ellipsis))]),
                    ExceptHandler(
                        type=Name(id='OtherException', ctx=Load()),
                        name='e',
                        body=[
                            Expr(
                                value=Constant(value=Ellipsis))])],
                orelse=[
                    Expr(
                        value=Constant(value=Ellipsis))],
                finalbody=[
                    Expr(
                        value=Constant(value=Ellipsis))])],
        type_ignores=[])
```

`try` blocks which are followed by `except*` clauses. The attributes are the same as for :class:`Try` but the :class:`ExceptHandler` nodes in `handlers` are interpreted as `except*` blocks rather then `except`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1172); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1172); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1176)

Unknown directive type "doctest".

```
.. doctest::

     >>> print(ast.dump(ast.parse("""
     ... try:
     ...     ...
     ... except* Exception:
     ...     ...
     ... """), indent=4))
     Module(
         body=[
             TryStar(
                 body=[
                     Expr(
                         value=Constant(value=Ellipsis))],
                 handlers=[
                     ExceptHandler(
                         type=Name(id='Exception', ctx=Load()),
                         body=[
                             Expr(
                                 value=Constant(value=Ellipsis))])],
                 orelse=[],
                 finalbody=[])],
         type_ignores=[])
```

A single `except` clause. `type` is the exception type it will match, typically a :class:`Name` node (or `None` for a catch-all `except:` clause). `name` is a raw string for the name to hold the exception, or `None` if the clause doesn't have `as foo`. `body` is a list of nodes.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1203); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1208)

Unknown directive type "doctest".

```
.. doctest::

     >>> print(ast.dump(ast.parse("""\
     ... try:
     ...     a + 1
     ... except TypeError:
     ...     pass
     ... """), indent=4))
     Module(
         body=[
             Try(
                 body=[
                     Expr(
                         value=BinOp(
                             left=Name(id='a', ctx=Load()),
                             op=Add(),
                             right=Constant(value=1)))],
                 handlers=[
                     ExceptHandler(
                         type=Name(id='TypeError', ctx=Load()),
                         body=[
                             Pass()])],
                 orelse=[],
                 finalbody=[])],
         type_ignores=[])
```

A `with` block. `items` is a list of :class:`withitem` nodes representing the context managers, and `body` is the indented block inside the context.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1237); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1240)

Unknown directive type "attribute".

```
.. attribute:: type_comment

    ``type_comment`` is an optional string with the type annotation as a comment.
```

A single context manager in a `with` block. `context_expr` is the context manager, often a :class:`Call` node. `optional_vars` is a :class:`Name`, :class:`Tuple` or :class:`List` for the `as foo` part, or `None` if that isn't used.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 1247); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 1247); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 1247); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 1247); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 1252)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse("""\
    ... with a as b, c as d:
    ...    something(b, d)
    ... """), indent=4))
    Module(
        body=[
            With(
                items=[
                    withitem(
                        context_expr=Name(id='a', ctx=Load()),
                        optional_vars=Name(id='b', ctx=Store())),
                    withitem(
                        context_expr=Name(id='c', ctx=Load()),
                        optional_vars=Name(id='d', ctx=Store()))],
                body=[
                    Expr(
                        value=Call(
                            func=Name(id='something', ctx=Load()),
                            args=[
                                Name(id='b', ctx=Load()),
                                Name(id='d', ctx=Load())],
                            keywords=[]))])],
        type_ignores=[])
```

Pattern matching

A `match` statement. `subject` holds the subject of the match (the object that is being matched against the cases) and `cases` contains an iterable of :class:`match_case` nodes with the different cases.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 1285); *backlink*

Unknown interpreted text role "class".

A single case pattern in a `match` statement. `pattern` contains the match pattern that the subject will be matched against. Note that the :class:`AST` nodes produced for patterns differ from those produced for expressions, even when they share the same syntax.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)ast.rst`, line 1291); *backlink*

Unknown interpreted text role "class".

The `guard` attribute contains an expression that will be evaluated if the pattern matches the subject.

`body` contains a list of nodes to execute if the pattern matches and the result of evaluating the guard expression is true.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1302)

Unknown directive type "doctest".

```
.. doctest::

     >>> print(ast.dump(ast.parse("""
     ... match x:
     ...     case [x] if x>0:
     ...         ...
     ...     case tuple():
     ...         ...
     ... """), indent=4))
     Module(
         body=[
             Match(
                 subject=Name(id='x', ctx=Load()),
                 cases=[
                     match_case(
                         pattern=MatchSequence(
                             patterns=[
                                 MatchAs(name='x')]),
                         guard=Compare(
                             left=Name(id='x', ctx=Load()),
                             ops=[
                                 Gt()],
                             comparators=[
                                 Constant(value=0)]),
                         body=[
                             Expr(
                                 value=Constant(value=Ellipsis))]),
                     match_case(
                         pattern=MatchClass(
                             cls=Name(id='tuple', ctx=Load()),
                             patterns=[],
                             kwd_attrs=[],
                             kwd_patterns=[]),
                         body=[
                             Expr(
                                 value=Constant(value=Ellipsis))])])],
         type_ignores=[])
```

A match literal or value pattern that compares by equality. `value` is an expression node. Permitted value nodes are restricted as described in the match statement documentation. This pattern succeeds if the match subject is equal to the evaluated value.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1347)

Unknown directive type "doctest".

```
.. doctest::

     >>> print(ast.dump(ast.parse("""
     ... match x:
     ...     case "Relevant":
     ...         ...
     ... """), indent=4))
     Module(
         body=[
             Match(
                 subject=Name(id='x', ctx=Load()),
                 cases=[
                     match_case(
                         pattern=MatchValue(
                             value=Constant(value='Relevant')),
                         body=[
                             Expr(
                                 value=Constant(value=Ellipsis))])])],
         type_ignores=[])
```

A match literal pattern that compares by identity. `value` is the singleton to be compared against: `None`, `True`, or `False`. This pattern succeeds if the match subject is the given constant.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1373)

Unknown directive type "doctest".

```
.. doctest::
```

```
>>> print(ast.dump(ast.parse("""
... match x:
...     case None:
...         ...
... """), indent=4))
Module(
    body=[
        Match(
            subject=Name(id='x', ctx=Load()),
            cases=[
                match_case(
                    pattern=MatchSingleton(value=None),
                    body=[
                        Expr(
                            value=Constant(value=Ellipsis))])])],
    type_ignores=[])
```

A match sequence pattern. `patterns` contains the patterns to be matched against the subject elements if the subject is a sequence. Matches a variable length sequence if one of the subpatterns is a `MatchStar` node, otherwise matches a fixed length sequence.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1399)

Unknown directive type "doctest".

```
.. doctest::
```

```
>>> print(ast.dump(ast.parse("""
... match x:
...     case [1, 2]:
...         ...
... """), indent=4))
Module(
    body=[
        Match(
            subject=Name(id='x', ctx=Load()),
            cases=[
                match_case(
                    pattern=MatchSequence(
                        patterns=[
                            MatchValue(
                                value=Constant(value=1)),
                            MatchValue(
                                value=Constant(value=2))]),
                    body=[
                        Expr(
                            value=Constant(value=Ellipsis))])])],
    type_ignores=[])
```

Matches the rest of the sequence in a variable length match sequence pattern. If `name` is not `None`, a list containing the remaining sequence elements is bound to that name if the overall sequence pattern is successful.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1429)

Unknown directive type "doctest".

```
.. doctest::
```

```
>>> print(ast.dump(ast.parse("""
... match x:
...     case [1, 2, *rest]:
...         ...
...     case [*_]:
...         ...
... """), indent=4))
Module(
    body=[
        Match(
            subject=Name(id='x', ctx=Load()),
            cases=[
                match_case(
                    pattern=MatchSequence(
                        patterns=[
                            MatchValue(
                                value=Constant(value=1)),
                            MatchValue(
                                value=Constant(value=2)),
                            MatchStar(name='rest')]),
                    body=[
                        Expr(
                            value=Constant(value=Ellipsis))]),
                match_case(
                    pattern=MatchSequence(
```

```
                            patterns=[
                                MatchStar()]),
                        body=[
                            Expr(
                                value=Constant(value=Ellipsis))])])],
        type_ignores=[])
```

A match mapping pattern. `keys` is a sequence of expression nodes. `patterns` is a corresponding sequence of pattern nodes. `rest` is an optional name that can be specified to capture the remaining mapping elements. Permitted key expressions are restricted as described in the match statement documentation.

This pattern succeeds if the subject is a mapping, all evaluated key expressions are present in the mapping, and the value corresponding to each key matches the corresponding subpattern. If `rest` is not `None`, a dict containing the remaining mapping elements is bound to that name if the overall mapping pattern is successful.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1477)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse("""
    ... match x:
    ...     case {1: _, 2: _}:
    ...         ...
    ...     case {**rest}:
    ...         ...
    ... """), indent=4))
    Module(
        body=[
            Match(
                subject=Name(id='x', ctx=Load()),
                cases=[
                    match_case(
                        pattern=MatchMapping(
                            keys=[
                                Constant(value=1),
                                Constant(value=2)],
                            patterns=[
                                MatchAs(),
                                MatchAs()]),
                        body=[
                            Expr(
                                value=Constant(value=Ellipsis))]),
                    match_case(
                        pattern=MatchMapping(keys=[], patterns=[], rest='rest'),
                        body=[
                            Expr(
                                value=Constant(value=Ellipsis))])])],
        type_ignores=[])
```

A match class pattern. `cls` is an expression giving the nominal class to be matched. `patterns` is a sequence of pattern nodes to be matched against the class defined sequence of pattern matching attributes. `kwd_attrs` is a sequence of additional attributes to be matched (specified as keyword arguments in the class pattern), `kwd_patterns` are the corresponding patterns (specified as keyword values in the class pattern).

This pattern succeeds if the subject is an instance of the nominated class, all positional patterns match the corresponding class-defined attributes, and any specified keyword attributes match their corresponding pattern.

Note: classes may define a property that returns self in order to match a pattern node against the instance being matched. Several builtin types are also matched that way, as described in the match statement documentation.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1526)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse("""
    ... match x:
    ...     case Point2D(0, 0):
    ...         ...
    ...     case Point3D(x=0, y=0, z=0):
    ...         ...
    ... """), indent=4))
    Module(
        body=[
            Match(
                subject=Name(id='x', ctx=Load()),
                cases=[
                    match_case(
                        pattern=MatchClass(
```

```
                            cls=Name(id='Point2D', ctx=Load()),
                            patterns=[
                                MatchValue(
                                    value=Constant(value=0)),
                                MatchValue(
                                    value=Constant(value=0))],
                            kwd_attrs=[],
                            kwd_patterns=[]),
                        body=[
                            Expr(
                                value=Constant(value=Ellipsis))]),
                    match_case(
                        pattern=MatchClass(
                            cls=Name(id='Point3D', ctx=Load()),
                            patterns=[],
                            kwd_attrs=[
                                'x',
                                'y',
                                'z'],
                            kwd_patterns=[
                                MatchValue(
                                    value=Constant(value=0)),
                                MatchValue(
                                    value=Constant(value=0)),
                                MatchValue(
                                    value=Constant(value=0))]),
                        body=[
                            Expr(
                                value=Constant(value=Ellipsis))])])],
            type_ignores=[])
```

A match "as-pattern", capture pattern or wildcard pattern. `pattern` contains the match pattern that the subject will be matched against. If the pattern is `None`, the node represents a capture pattern (i.e a bare name) and will always succeed.

The `name` attribute contains the name that will be bound if the pattern is successful. If `name` is `None`, `pattern` must also be `None` and the node represents the wildcard pattern.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1584)

Unknown directive type "doctest".

```
.. doctest::

     >>> print(ast.dump(ast.parse("""
     ... match x:
     ...     case [x] as y:
     ...         ...
     ...     case _:
     ...         ...
     ... """), indent=4))
     Module(
         body=[
             Match(
                 subject=Name(id='x', ctx=Load()),
                 cases=[
                     match_case(
                         pattern=MatchAs(
                             pattern=MatchSequence(
                                 patterns=[
                                     MatchAs(name='x')]),
                             name='y'),
                         body=[
                             Expr(
                                 value=Constant(value=Ellipsis))]),
                     match_case(
                         pattern=MatchAs(),
                         body=[
                             Expr(
                                 value=Constant(value=Ellipsis))])])],
         type_ignores=[])
```

A match "or-pattern". An or-pattern matches each of its subpatterns in turn to the subject, until one succeeds. The or-pattern is then deemed to succeed. If none of the subpatterns succeed the or-pattern fails. The `patterns` attribute contains a list of match pattern nodes that will be matched against the subject.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1622)

Unknown directive type "doctest".

```
.. doctest::

     >>> print(ast.dump(ast.parse("""
```

```
... match x:
...     case [x] | (y):
...         ...
... """), indent=4))
Module(
    body=[
        Match(
            subject=Name(id='x', ctx=Load()),
            cases=[
                match_case(
                    pattern=MatchOr(
                        patterns=[
                            MatchSequence(
                                patterns=[
                                    MatchAs(name='x')]),
                            MatchAs(name='y')]),
                    body=[
                        Expr(
                            value=Constant(value=Ellipsis))])])],
    type_ignores=[])
```

Function and class definitions

A function definition.

- `name` is a raw string of the function name.
- `args` is an :class:`arguments` node.

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1655); *backlink*
>
> Unknown interpreted text role "class".

- `body` is the list of nodes inside the function.
- `decorator_list` is the list of decorators to be applied, stored outermost first (i.e. the first in the list will be applied last).
- `returns` is the return annotation.

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1661)
>
> Unknown directive type "attribute".
>
> ```
> .. attribute:: type_comment
>
>     ``type_comment`` is an optional string with the type annotation as a comment.
> ```

`lambda` is a minimal function definition that can be used inside an expression. Unlike :class:`FunctionDef`, `body` holds a single node.

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1668); *backlink*
>
> Unknown interpreted text role "class".

> **System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1671)
>
> Unknown directive type "doctest".
>
> ```
> .. doctest::
>
>     >>> print(ast.dump(ast.parse('lambda x,y: ...'), indent=4))
>     Module(
>         body=[
>             Expr(
>                 value=Lambda(
>                     args=arguments(
>                         posonlyargs=[],
>                         args=[
>                             arg(arg='x'),
>                             arg(arg='y')],
>                         kwonlyargs=[],
>                         kw_defaults=[],
>                         defaults=[]),
>                     body=Constant(value=Ellipsis)))],
>         type_ignores=[])
> ```

The arguments for a function.

- `posonlyargs`, `args` and `kwonlyargs` are lists of :class:`arg` nodes.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1694); *backlink*

  Unknown interpreted text role "class".

- `vararg` and `kwarg` are single :class:`arg` nodes, referring to the `*args, **kwargs` parameters.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1695); *backlink*

  Unknown interpreted text role "class".

- `kw_defaults` is a list of default values for keyword-only arguments. If one is `None`, the corresponding argument is required.
- `defaults` is a list of default values for arguments that can be passed positionally. If there are fewer defaults, they correspond to the last n arguments.

A single argument in a list. `arg` is a raw string of the argument name, `annotation` is its annotation, such as a :class:`Str` or :class:`Name` node.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1706); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1706); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1710)

Unknown directive type "attribute".

```
.. attribute:: type_comment

    ``type_comment`` is an optional string with the type annotation as a comment
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1714)

Unknown directive type "doctest".

```
.. doctest::

     >>> print(ast.dump(ast.parse("""\
     ... @decorator1
     ... @decorator2
     ... def f(a: 'annotation', b=1, c=2, *d, e, f=3, **g) -> 'return annotation':
     ...     pass
     ... """), indent=4))
     Module(
         body=[
             FunctionDef(
                 name='f',
                 args=arguments(
                     posonlyargs=[],
                     args=[
                         arg(
                             arg='a',
                             annotation=Constant(value='annotation')),
                         arg(arg='b'),
                         arg(arg='c')],
                     vararg=arg(arg='d'),
                     kwonlyargs=[
                         arg(arg='e'),
                         arg(arg='f')],
                     kw_defaults=[
                         None,
                         Constant(value=3)],
                     kwarg=arg(arg='g'),
                     defaults=[
                         Constant(value=1),
                         Constant(value=2)]),
                 body=[
```

```
                    Pass()],
                decorator_list=[
                    Name(id='decorator1', ctx=Load()),
                    Name(id='decorator2', ctx=Load())],
                returns=Constant(value='return annotation'))],
            type_ignores=[])
```

A `return` statement.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1758)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('return 4'), indent=4))
    Module(
        body=[
            Return(
                value=Constant(value=4))],
        type_ignores=[])
```

A `yield` or `yield from` expression. Because these are expressions, they must be wrapped in a :class:`Expr` node if the value sent back is not used.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1771); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1774)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('yield x'), indent=4))
    Module(
        body=[
            Expr(
                value=Yield(
                    value=Name(id='x', ctx=Load()))))],
        type_ignores=[])

    >>> print(ast.dump(ast.parse('yield from x'), indent=4))
    Module(
        body=[
            Expr(
                value=YieldFrom(
                    value=Name(id='x', ctx=Load()))))],
        type_ignores=[])
```

`global` and `nonlocal` statements. `names` is a list of raw strings.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1798)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse('global x,y,z'), indent=4))
    Module(
        body=[
            Global(
                names=[
                    'x',
                    'y',
                    'z'])],
        type_ignores=[])

    >>> print(ast.dump(ast.parse('nonlocal x,y,z'), indent=4))
    Module(
        body=[
            Nonlocal(
                names=[
                    'x',
                    'y',
                    'z'])],
```

```
            type_ignores=[])
```

A class definition.

- `name` is a raw string for the class name
- `bases` is a list of nodes for explicitly specified base classes.
- `keywords` is a list of :class:`keyword` nodes, principally for 'metaclass'. Other keywords will be passed to the metaclass, as per PEP-3115.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1827); *backlink*

  Unknown interpreted text role "class".

- `starargs` and `kwargs` are each a single node, as in a function call. starargs will be expanded to join the list of base classes, and kwargs will be passed to the metaclass.
- `body` is a list of nodes representing the code within the class definition.
- `decorator_list` is a list of nodes, as in :class:`FunctionDef`.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1835); *backlink*

  Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1837)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse("""\
    ... @decorator1
    ... @decorator2
    ... class Foo(base1, base2, metaclass=meta):
    ...     pass
    ... """), indent=4))
    Module(
        body=[
            ClassDef(
                name='Foo',
                bases=[
                    Name(id='base1', ctx=Load()),
                    Name(id='base2', ctx=Load())],
                keywords=[
                    keyword(
                        arg='metaclass',
                        value=Name(id='meta', ctx=Load()))],
                body=[
                    Pass()],
                decorator_list=[
                    Name(id='decorator1', ctx=Load()),
                    Name(id='decorator2', ctx=Load())])],
        type_ignores=[])
```

Async and await

An `async def` function definition. Has the same fields as :class:`FunctionDef`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1868); *backlink*

Unknown interpreted text role "class".

An `await` expression. `value` is what it waits for. Only valid in the body of an :class:`AsyncFunctionDef`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1874); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1877)

Unknown directive type "doctest".

```
.. doctest::

    >>> print(ast.dump(ast.parse("""\
    ... async def f():
    ...     await other_func()
    ... """), indent=4))
    Module(
        body=[
            AsyncFunctionDef(
                name='f',
                args=arguments(
                    posonlyargs=[],
                    args=[],
                    kwonlyargs=[],
                    kw_defaults=[],
                    defaults=[]),
                body=[
                    Expr(
                        value=Await(
                            value=Call(
                                func=Name(id='other_func', ctx=Load()),
                                args=[],
                                keywords=[]))))],
                decorator_list=[])],
        type_ignores=[])
```

`async for` loops and `async with` context managers. They have the same fields as :class:`For` and :class:`With`, respectively. Only valid in the body of an :class:`AsyncFunctionDef`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1907); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1907); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1907); *backlink*

Unknown interpreted text role "class".

Note

When a string is parsed by :func:`ast.parse`, operator nodes (subclasses of :class:`ast.operator`, :class:`ast.unaryop`, :class:`ast.cmpop`, :class:`ast.boolop` and :class:`ast.expr_context`) on the returned tree will be singletons. Changes to one will be reflected in all other occurrences of the same value (e.g. :class:`ast.Add`).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1912); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1912); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1912); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1912); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1912); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1912); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1912); *backlink*

Unknown interpreted text role "class".

:mod:`ast` Helpers

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1919); *backlink*

Unknown interpreted text role "mod".

Apart from the node classes, the :mod:`ast` module defines these utility functions and classes for traversing abstract syntax trees:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1922); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1925)

Unknown directive type "function".

```
.. function:: parse(source, filename='<unknown>', mode='exec', *, type_comments=False, feature_version=

   Parse the source into an AST node.  Equivalent to ``compile(source,
   filename, mode, ast.PyCF_ONLY_AST)``.

   If ``type_comments=True`` is given, the parser is modified to check
   and return type comments as specified by :pep:`484` and :pep:`526`.
   This is equivalent to adding :data:`ast.PyCF_TYPE_COMMENTS` to the
   flags passed to :func:`compile()`.  This will report syntax errors
   for misplaced type comments.  Without this flag, type comments will
   be ignored, and the ``type_comment`` field on selected AST nodes
   will always be ``None``.  In addition, the locations of ``# type:
   ignore`` comments will be returned as the ``type_ignores``
   attribute of :class:`Module` (otherwise it is always an empty list).

   In addition, if ``mode`` is ``'func_type'``, the input syntax is
   modified to correspond to :pep:`484` "signature type comments",
   e.g. ``(str, int) -> List[str]``.

   Also, setting ``feature_version`` to a tuple ``(major, minor)``
   will attempt to parse using that Python version's grammar.
   Currently ``major`` must equal to ``3``.  For example, setting
   ``feature_version=(3, 4)`` will allow the use of ``async`` and
   ``await`` as variable names.  The lowest supported version is
   ``(3, 4)``; the highest is ``sys.version_info[0:2]``.

   If source contains a null character ('\0'), :exc:`ValueError` is raised.

    .. warning::
      Note that successfully parsing source code into an AST object doesn't
      guarantee that the source code provided is valid Python code that can
      be executed as the compilation step can raise further :exc:`SyntaxError`
      exceptions. For instance, the source ``return 42`` generates a valid
      AST node for a return statement, but it cannot be compiled alone (it needs
      to be inside a function node).

      In particular, :func:`ast.parse` won't do any scoping checks, which the
      compilation step does.

   .. warning::
      It is possible to crash the Python interpreter with a
```

```
      sufficiently large/complex string due to stack depth limitations
      in Python's AST compiler.

   .. versionchanged:: 3.8
      Added ``type_comments``, ``mode='func_type'`` and ``feature_version``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1973)

Unknown directive type "function".

```
.. function:: unparse(ast_obj)

   Unparse an :class:`ast.AST` object and generate a string with code
   that would produce an equivalent :class:`ast.AST` object if parsed
   back with :func:`ast.parse`.

   .. warning::
      The produced code string will not necessarily be equal to the original
      code that generated the :class:`ast.AST` object (without any compiler
      optimizations, such as constant tuples/frozensets).

   .. warning::
      Trying to unparse a highly complex expression would result with
      :exc:`RecursionError`.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 1991)

Unknown directive type "function".

```
.. function:: literal_eval(node_or_string)

   Safely evaluate an expression node or a string containing a Python literal or
   container display.  The string or node provided may only consist of the
   following Python literal structures: strings, bytes, numbers, tuples, lists,
   dicts, sets, booleans, ``None`` and ``Ellipsis``.

   This can be used for safely evaluating strings containing Python values from
   untrusted sources without the need to parse the values oneself.  It is not
   capable of evaluating arbitrarily complex expressions, for example involving
   operators or indexing.

   .. warning::
      It is possible to crash the Python interpreter with a
      sufficiently large/complex string due to stack depth limitations
      in Python's AST compiler.

      It can raise :exc:`ValueError`, :exc:`TypeError`, :exc:`SyntaxError`,
      :exc:`MemoryError` and :exc:`RecursionError` depending on the malformed
      input.

   .. versionchanged:: 3.2
      Now allows bytes and set literals.

   .. versionchanged:: 3.9
      Now supports creating empty sets with ``'set()'``.

   .. versionchanged:: 3.10
      For string inputs, leading spaces and tabs are now stripped.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2022)

Unknown directive type "function".

```
.. function:: get_docstring(node, clean=True)

   Return the docstring of the given *node* (which must be a
   :class:`FunctionDef`, :class:`AsyncFunctionDef`, :class:`ClassDef`,
   or :class:`Module` node), or ``None`` if it has no docstring.
   If *clean* is true, clean up the docstring's indentation with
   :func:`inspect.cleandoc`.

   .. versionchanged:: 3.5
      :class:`AsyncFunctionDef` is now supported.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2034)

Unknown directive type "function".

```
.. function:: get_source_segment(source, node, *, padded=False)

   Get source code segment of the *source* that generated *node*.
   If some location information (:attr:`lineno`, :attr:`end_lineno`,
   :attr:`col_offset`, or :attr:`end_col_offset`) is missing, return ``None``.

   If *padded* is ``True``, the first line of a multi-line statement will
   be padded with spaces to match its original position.

   .. versionadded:: 3.8
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2046)

Unknown directive type "function".

```
.. function:: fix_missing_locations(node)

   When you compile a node tree with :func:`compile`, the compiler expects
   :attr:`lineno` and :attr:`col_offset` attributes for every node that supports
   them.  This is rather tedious to fill in for generated nodes, so this helper
   adds these attributes recursively where not already set, by setting them to
   the values of the parent node.  It works recursively starting at *node*.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2055)

Unknown directive type "function".

```
.. function:: increment_lineno(node, n=1)

   Increment the line number and end line number of each node in the tree
   starting at *node* by *n*. This is useful to "move code" to a different
   location in a file.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2062)

Unknown directive type "function".

```
.. function:: copy_location(new_node, old_node)

   Copy source location (:attr:`lineno`, :attr:`col_offset`, :attr:`end_lineno`,
   and :attr:`end_col_offset`) from *old_node* to *new_node* if possible,
   and return *new_node*.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2069)

Unknown directive type "function".

```
.. function:: iter_fields(node)

   Yield a tuple of ``(fieldname, value)`` for each field in ``node._fields``
   that is present on *node*.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2075)

Unknown directive type "function".

```
.. function:: iter_child_nodes(node)

   Yield all direct child nodes of *node*, that is, all fields that are nodes
   and all items of fields that are lists of nodes.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2081)

Unknown directive type "function".

```
.. function:: walk(node)

   Recursively yield all descendant nodes in the tree starting at *node*
   (including *node* itself), in no specified order.  This is useful if you only
   want to modify nodes in place and don't care about the context.
```

A node visitor base class that walks the abstract syntax tree and calls a visitor function for every node found. This function may return a value which is forwarded by the :meth:`visit` method.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2090); *backlink*

Unknown interpreted text role "meth".

This class is meant to be subclassed, with the subclass adding visitor methods.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2097)

Unknown directive type "method".

```
.. method:: visit(node)

   Visit a node.  The default implementation calls the method called
   :samp:`self.visit_{classname}` where *classname* is the name of the node
   class, or :meth:`generic_visit` if that method doesn't exist.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2103)

Unknown directive type "method".

```
.. method:: generic_visit(node)

   This visitor calls :meth:`visit` on all children of the node.

   Note that child nodes of nodes that have a custom visitor method won't be
   visited unless the visitor calls :meth:`generic_visit` or visits them
   itself.
```

Don't use the :class:`NodeVisitor` if you want to apply changes to nodes during traversal. For this a special visitor exists (:class:`NodeTransformer`) that allows modifications.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2111); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2111); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2115)

Unknown directive type "deprecated".

```
.. deprecated:: 3.8

   Methods :meth:`visit_Num`, :meth:`visit_Str`, :meth:`visit_Bytes`,
   :meth:`visit_NameConstant` and :meth:`visit_Ellipsis` are deprecated
   now and will not be called in future Python versions.  Add the
   :meth:`visit_Constant` method to handle all constant nodes.
```

A :class:`NodeVisitor` subclass that walks the abstract syntax tree and allows modification of nodes.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2125); *backlink*

Unknown interpreted text role "class".

The :class:`NodeTransformer` will walk the AST and use the return value of the visitor methods to replace or remove the old node. If the return value of the visitor method is `None`, the node will be removed from its location, otherwise it is replaced with the return value. The return value may be the original node in which case no replacement takes place.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2128); *backlink*

Unknown interpreted text role "class".

Here is an example transformer that rewrites all occurrences of name lookups (`foo`) to `data['foo']`:

```
class RewriteName(NodeTransformer):

    def visit_Name(self, node):
        return Subscript(
            value=Name(id='data', ctx=Load()),
            slice=Constant(value=node.id),
            ctx=node.ctx
        )
```

Keep in mind that if the node you're operating on has child nodes you must either transform the child nodes yourself or call the :meth:`generic_visit` method for the node first.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2146); *backlink*

Unknown interpreted text role "meth".

For nodes that were part of a collection of statements (that applies to all statement nodes), the visitor may also return a list of nodes rather than just a single node.

If :class:`NodeTransformer` introduces new nodes (that weren't part of original tree) without giving them location information (such as :attr:`lineno`), :func:`fix_missing_locations` should be called with the new sub-tree to recalculate the location information:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2154); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2154); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2154); *backlink*

Unknown interpreted text role "func".

```
tree = ast.parse('foo', mode='eval')
new_tree = fix_missing_locations(RewriteName().visit(tree))
```

Usually you use the transformer like this:

```
node = YourTransformer().visit(node)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2167)

Unknown directive type "function".

```
.. function:: dump(node, annotate_fields=True, include_attributes=False, *, indent=None)

   Return a formatted dump of the tree in *node*.  This is mainly useful for
   debugging purposes.  If *annotate_fields* is true (by default),
   the returned string will show the names and the values for fields.
   If *annotate_fields* is false, the result string will be more compact by
   omitting unambiguous field names.  Attributes such as line
   numbers and column offsets are not dumped by default.  If this is wanted,
   *include_attributes* can be set to true.

   If *indent* is a non-negative integer or string, then the tree will be
   pretty-printed with that indent level.  An indent level
   of 0, negative, or ``""`` will only insert newlines.  ``None`` (the default)
   selects the single line representation. Using a positive integer indent
   indents that many spaces per level.  If *indent* is a string (such as ``"\t"``),
   that string is used to indent each level.
```

```
.. versionchanged:: 3.9
   Added the *indent* option.
```

Compiler Flags

The following flags may be passed to :func:`compile` in order to change effects on the compilation of a program:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2193); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2196)

Unknown directive type "data".

```
.. data:: PyCF_ALLOW_TOP_LEVEL_AWAIT

   Enables support for top-level ``await``, ``async for``, ``async with``
   and async comprehensions.

   .. versionadded:: 3.8
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2203)

Unknown directive type "data".

```
.. data:: PyCF_ONLY_AST

   Generates and returns an abstract syntax tree instead of returning a
   compiled code object.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2208)

Unknown directive type "data".

```
.. data:: PyCF_TYPE_COMMENTS

   Enables support for :pep:`484` and :pep:`526` style type comments
   (``# type: <type>``, ``# type: ignore <stuff>``).

   .. versionadded:: 3.8
```

Command-Line Usage

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2221)

Unknown directive type "versionadded".

```
.. versionadded:: 3.9
```

The :mod:`ast` module can be executed as a script from the command line. It is as simple as:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2223); *backlink*

Unknown interpreted text role "mod".

```
python -m ast [-m <mode>] [-a] [infile]
```

The following options are accepted:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2232)

Unknown directive type "program".

```
.. program:: ast
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2234)

Unknown directive type "cmdoption".

```
.. cmdoption:: -h, --help

   Show the help message and exit.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2238)

Unknown directive type "cmdoption".

```
.. cmdoption:: -m <mode>
               --mode <mode>

   Specify what kind of code must be compiled, like the *mode* argument
   in :func:`parse`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2244)

Unknown directive type "cmdoption".

```
.. cmdoption:: --no-type-comments

   Don't parse type comments.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2248)

Unknown directive type "cmdoption".

```
.. cmdoption:: -a, --include-attributes

   Include attributes such as line numbers and column offsets.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2252)

Unknown directive type "cmdoption".

```
.. cmdoption:: -i <indent>
               --indent <indent>

   Indentation of nodes in AST (number of spaces).
```

If :file:`infile` is specified its contents are parsed to AST and dumped to stdout. Otherwise, the content is read from stdin.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2257); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)ast.rst, line 2261)

Unknown directive type "seealso".

```
.. seealso::

   `Green Tree Snakes <https://greentreesnakes.readthedocs.io/>`_, an external
   documentation resource, has good details on working with Python ASTs.

   `ASTTokens <https://asttokens.readthedocs.io/en/latest/user-guide.html>`_
   annotates Python ASTs with the positions of tokens and text in the source
   code that generated them. This is helpful for tools that make source code
   transformations.

   `leoAst.py <http://leoeditor.com/appendices.html#leoast-py>`_ unifies the
   token-based and parse-tree-based views of python programs by inserting
   two-way links between tokens and ast nodes.

   `LibCST <https://libcst.readthedocs.io/>`_ parses code as a Concrete Syntax
```

```
Tree that looks like an ast tree and keeps all formatting details. It's
useful for building automated refactoring (codemod) applications and
linters.

`Parso <https://parso.readthedocs.io>`_ is a Python parser that supports
error recovery and round-trip parsing for different Python versions (in
multiple Python versions). Parso is also able to list multiple syntax errors
in your python file.
```