:mod:`codeop` --- Compile Python code

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

Unknown directive type "module".

```
.. module:: codeop
   :synopsis: Compile (possibly incomplete) Python code.
```

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

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Moshe Zadka <moshez@zadka.site.co.il>
```

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

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Michael Hudson <mwh@python.net>
```

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

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

Unknown interpreted text role "source".

The :mod:`codeop` module provides utilities upon which the Python read-eval-print loop can be emulated, as is done in the :mod:`code` module. As a result, you probably don't want to use the module directly; if you want to include such a loop in your program you probably want to use the :mod:`code` module instead.

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

Unknown interpreted text role "mod".

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

Unknown interpreted text role "mod".

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

Unknown interpreted text role "mod".

There are two parts to this job:

1. Being able to tell if a line of input completes a Python statement: in short, telling whether to print '>>>' or '...' next.
2. Remembering which future statements the user has entered, so subsequent input can be compiled with these in effect.

The :mod:`codeop` module provides a way of doing each of these things, and a way of doing them both.

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)codeop.rst, line 28); *backlink*

Unknown interpreted text role "mod".

To do just the former:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)codeop.rst, line 33)

Unknown directive type "function".

```
.. function:: compile_command(source, filename="<input>", symbol="single")

   Tries to compile *source*, which should be a string of Python code and return a
   code object if *source* is valid Python code. In that case, the filename
   attribute of the code object will be *filename*, which defaults to
   ``'<input>'``. Returns ``None`` if *source* is *not* valid Python code, but is a
   prefix of valid Python code.

   If there is a problem with *source*, an exception will be raised.
   :exc:`SyntaxError` is raised if there is invalid Python syntax, and
   :exc:`OverflowError` or :exc:`ValueError` if there is an invalid literal.

   The *symbol* argument determines whether *source* is compiled as a statement
   (``'single'``, the default), as a sequence of statements (``'exec'``) or
   as an :term:`expression` (``'eval'``).  Any other value will
   cause :exc:`ValueError` to  be raised.

   .. note::

      It is possible (but not likely) that the parser stops parsing with a
      successful outcome before reaching the end of the source; in this case,
      trailing symbols may be ignored instead of causing an error.  For example,
      a backslash followed by two newlines may be followed by arbitrary garbage.
      This will be fixed once the API for the parser is better.
```

Instances of this class have :meth:`__call__` methods identical in signature to the built-in function :func:`compile`, but with the difference that if the instance compiles program text containing a :mod:`__future__` statement, the instance 'remembers' and compiles all subsequent program texts with the statement in force.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)codeop.rst, line 61); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)codeop.rst, line 61); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)codeop.rst, line 61); *backlink*

Unknown interpreted text role "mod".

Instances of this class have :meth:`__call__` methods identical in signature to :func:`compile_command`; the difference is that if the instance compiles program text containing a `__future__` statement, the instance 'remembers' and compiles all subsequent program texts with the statement in force.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)codeop.rst, line 70); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)codeop.rst, line 70); *backlink*

Unknown interpreted text role "func".