Built-in Functions

The Python interpreter has a number of functions and types built into it that are always available. They are listed here in alphabetical order.

Built-in Functions

A
:func:`abs`
:func:`aiter`
:func:`all`
:func:`any`
:func:`anext`
:func:`ascii`

B
:func:`bin`
:func:`bool`
:func:`breakpoint`
`bytearray()`
`bytes()`

C
:func:`callable`
:func:`chr`
:func:`classmethod`
:func:`compile`
:func:`complex`

D
:func:`delattr`
`dict()`
:func:`dir`
:func:`divmod`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 15); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 16); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 17); *backlink*

Unknown interpreted text role "func".

E
:func:`enumerate`
:func:`eval`
:func:`exec`

F
:func:`filter`
:func:`float`
:func:`format`
`frozenset()`

G
:func:`getattr`
:func:`globals`

H
:func:`hasattr`
:func:`hash`
:func:`help`
:func:`hex`

I
:func:`id`
:func:`input`
:func:`int`
:func:`isinstance`
:func:`issubclass`
:func:`iter`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 15); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 16); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 17); *backlink*

Unknown interpreted text role "func".

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

L
:func:`len`
`list()`
:func:`locals`

M
:func:`map`
:func:`max`
`memoryview()`
:func:`min`

N
:func:`next`

O
:func:`object`
:func:`oct`
:func:`open`
:func:`ord`

P
:func:`pow`
:func:`print`
:func:`property`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 15); *backlink*

Unknown interpreted text role "func".

R
`range()`
:func:`repr`
:func:`reversed`
:func:`round`

S
`set()`
:func:`setattr`
:func:`slice`
:func:`sorted`
:func:`staticmethod`
`str()`
:func:`sum`
:func:`super`

T
`tuple()`
:func:`type`

V
:func:`vars`

Z
:func:`zip`

_
:func:`__import__`

System Mes ERROR/3 (D:\onboard resources\s onboarding- resources\c main\Doc\li [cpython-ma [library]fu line 16); *bac*

Unknown inte role "func".

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

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 19); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 20); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.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]functions.rst, line 24); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 25); *backlink*

Unknown interpreted text role "func".

System Message:

[cpython-main][Doc][library]functions.rst, line 20); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 21); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 22); *backlink*

Unknown interpreted text role "func".

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

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 27); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 30); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 31); *backlink*

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 17); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 20); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 21); *backlink*

Unknown interpreted text role "func".

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

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 29); *backlink*

Unknown interpreted text role "func".

System Message:

System Mes ERROR/3 (D:\onboard resources\s onboarding- resources\c main\Doc\li [cpython-ma [library]fu line 17); *bac*

Unknown inte role "func".

System Mes ERROR/3 (D:\onboard resources\s onboarding- resources\c main\Doc\li [cpython-ma [library]fu line 18); *bac*

Unknown inte role "func".

System Mes ERROR/3 (D:\onboard resources\s onboarding- resources\c main\Doc\li [cpython-ma [library]fu line 22); *bac*

Unknown inte role "func".

System Mes ERROR/3 (D:\onboard resources\s onboarding- resources\c main\Doc\li [cpython-ma [library]fu line 23); *bac*

Unknown inte role "func".

System Mes ERROR/3 (D:\onboard resources\s onboarding- resources\c main\Doc\li [cpython-ma [library]fu line 24); *bac*

Unknown inte role "func".

System Mes ERROR/3 (D:\onboard resources\s onboarding- resources\c main\Doc\li [cpython-ma [library]fu line 25); *bac*

Unknown inte role "func".

System Mes

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 30); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 31); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 32); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 33); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 34); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 37); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 32); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 33); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 36); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 37); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 38); *backlink*

Unknown interpreted text role "func".

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

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 30); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 31); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 32); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 35); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 36); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 37); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 27); *backl*

Unknown inte role "func".

System Mes ERROR/3 (D:\onboard resources\s onboarding- resources\c main\Doc\li [cpython-ma [library]fu line 28); *bac*

Unknown inte role "func".

System Mes ERROR/3 (D:\onboard resources\s onboarding- resources\c main\Doc\li [cpython-ma [library]fu line 32); *bac*

Unknown inte role "func".

System Mes ERROR/3 (D:\onboard resources\s onboarding- resources\c main\Doc\li [cpython-ma [library]fu line 35); *bac*

Unknown inte role "func".

System Mes ERROR/3 (D:\onboard resources\s onboarding- resources\c main\Doc\li [cpython-ma [library]fu line 38); *bac*

Unknown inte role "func".

System Mes ERROR/3 (D:\onboard resources\s onboarding- resources\c main\Doc\li [cpython-ma [library]fu line 41); *bac*

Unknown inte role "func".

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

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 40); backlink

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 40); backlink

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 41); backlink

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 57)

Unknown directive type "function".

```
.. function:: abs(x)

   Return the absolute value of a number.  The argument may be an
   integer, a floating point number, or an object implementing :meth:`__abs__`.
   If the argument is a complex number, its magnitude is returned.
```

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

Unknown directive type "function".

```
.. function:: aiter(async_iterable)

   Return an :term:`asynchronous iterator` for an :term:`asynchronous iterable`.
   Equivalent to calling ``x.__aiter__()``.

   Note: Unlike :func:`iter`, :func:`aiter` has no 2-argument variant.

   .. versionadded:: 3.10
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 73)

Unknown directive type "function".

```
.. function:: all(iterable)

   Return ``True`` if all elements of the *iterable* are true (or if the iterable
   is empty).  Equivalent to::

      def all(iterable):
          for element in iterable:
              if not element:
                  return False
          return True
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 85)

Unknown directive type "awaitablefunction".

```
.. awaitablefunction:: anext(async_iterator[, default])

   When awaited, return the next item from the given :term:`asynchronous
   iterator`, or *default* if given and the iterator is exhausted.

   This is the async variant of the :func:`next` builtin, and behaves
   similarly.

   This calls the :meth:`~object.__anext__` method of *async_iterator*,
   returning an :term:`awaitable`. Awaiting this returns the next value of the
   iterator. If *default* is given, it is returned if the iterator is exhausted,
   otherwise :exc:`StopAsyncIteration` is raised.

   .. versionadded:: 3.10
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 100)

Unknown directive type "function".

```
.. function:: any(iterable)

   Return ``True`` if any element of the *iterable* is true.  If the iterable
   is empty, return ``False``.  Equivalent to::

      def any(iterable):
          for element in iterable:
              if element:
                  return True
          return False
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 112)

Unknown directive type "function".

```
.. function:: ascii(object)

   As :func:`repr`, return a string containing a printable representation of an
   object, but escape the non-ASCII characters in the string returned by
   :func:`repr` using ``\x``, ``\u``, or ``\U`` escapes.  This generates a string
   similar to that returned by :func:`repr` in Python 2.
```

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

Unknown directive type "function".

```
.. function:: bin(x)

   Convert an integer number to a binary string prefixed with "0b". The result
   is a valid Python expression. If *x* is not a Python :class:`int` object, it
   has to define an :meth:`__index__` method that returns an integer. Some
   examples:

      >>> bin(3)
      '0b11'
      >>> bin(-10)
      '-0b1010'

   If the prefix "0b" is desired or not, you can use either of the following ways.

      >>> format(14, '#b'), format(14, 'b')
      ('0b1110', '1110')
      >>> f'{14:#b}', f'{14:b}'
      ('0b1110', '1110')

   See also :func:`format` for more information.
```

Return a Boolean value, i.e. one of `True` or `False`. *x* is converted using the standard :ref:`truth testing procedure <truth>`. If *x* is false or omitted, this returns `False`; otherwise, it returns `True`. The :class:`bool` class is a subclass of :class:`int` (see :ref:`typesnumeric`). It cannot be subclassed further. Its only instances are `False` and `True` (see :ref:`bltin-boolean-values`).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 144); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 144); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 144); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 144); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 144); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 151)

Unknown directive type "index".

```
.. index:: pair: Boolean; type
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 153)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
   *x* is now a positional-only parameter.
```

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

Unknown directive type "function".

```
.. function:: breakpoint(*args, **kws)

   This function drops you into the debugger at the call site.  Specifically,
   it calls :func:`sys.breakpointhook`, passing ``args`` and ``kws`` straight
   through.  By default, ``sys.breakpointhook()`` calls
   :func:`pdb.set_trace()` expecting no arguments.  In this case, it is
   purely a convenience function so you don't have to explicitly import
   :mod:`pdb` or type as much code to enter the debugger.  However,
   :func:`sys.breakpointhook` can be set to some other function and
   :func:`breakpoint` will automatically call that, allowing you to drop into
   the debugger of choice.

   .. audit-event:: builtins.breakpoint breakpointhook breakpoint

   .. versionadded:: 3.7
```

Return a new array of bytes. The :class:`bytearray` class is a mutable sequence of integers in the range 0 <= x < 256. It has most of the usual methods of mutable sequences, described in :ref:`typesseq-mutable`, as well as most methods that the :class:`bytes` type has, see :ref:`bytes-methods`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 176); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 176); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 176); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 176); *backlink*

Unknown interpreted text role "ref".

The optional *source* parameter can be used to initialize the array in a few different ways:

- If it is a *string*, you must also give the *encoding* (and optionally, *errors*) parameters; :func:`bytearray` then converts the string to bytes using :meth:`str.encode`.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 184); *backlink*

  Unknown interpreted text role "func".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 184); *backlink*

  Unknown interpreted text role "meth".

- If it is an *integer*, the array will have that size and will be initialized with null bytes.
- If it is an object conforming to the :ref:`buffer interface <bufferobjects>`, a read-only buffer of the object will be used to initialize the bytes array.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 191); *backlink*

  Unknown interpreted text role "ref".

- If it is an *iterable*, it must be an iterable of integers in the range `0 <= x < 256`, which are used as the initial contents of the array.

Without an argument, an array of size 0 is created.

See also :ref:`binaryseq` and :ref:`typebytearray`.

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

Unknown interpreted text role "ref".

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

Unknown interpreted text role "ref".

Return a new "bytes" object which is an immutable sequence of integers in the range `0 <= x < 256`. :class:`bytes` is an immutable version of :class:`bytearray` -- it has the same non-mutating methods and the same indexing and slicing behavior.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 206); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 206); *backlink*

Unknown interpreted text role "class".

Accordingly, constructor arguments are interpreted as for :func:`bytearray`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 211); *backlink*

Unknown interpreted text role "func".

Bytes objects can also be created with literals, see :ref:`strings`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 213); *backlink*

Unknown interpreted text role "ref".

See also :ref:`binaryseq`, :ref:`typebytes`, and :ref:`bytes-methods`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 215); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 215); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 215); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 218)

Unknown directive type "function".

```
.. function:: callable(object)

   Return :const:`True` if the *object* argument appears callable,
   :const:`False` if not.  If this returns ``True``, it is still possible that a
   call fails, but if it is ``False``, calling *object* will never succeed.
   Note that classes are callable (calling a class returns a new instance);
   instances are callable if their class has a :meth:`__call__` method.

   .. versionadded:: 3.2
      This function was first removed in Python 3.0 and then brought back
      in Python 3.2.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 231)

Unknown directive type "function".

```
.. function:: chr(i)

   Return the string representing a character whose Unicode code point is the
   integer *i*.  For example, ``chr(97)`` returns the string ``'a'``, while
   ``chr(8364)`` returns the string ``'â,¬'``. This is the inverse of :func:`ord`.

   The valid range for the argument is from 0 through 1,114,111 (0x10FFFF in
   base 16).  :exc:`ValueError` will be raised if *i* is outside that range.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 241)

Unknown directive type "decorator".

```
.. decorator:: classmethod
```

```
Transform a method into a class method.

A class method receives the class as an implicit first argument, just like an
instance method receives the instance. To declare a class method, use this
idiom::

   class C:
       @classmethod
       def f(cls, arg1, arg2): ...

The ``@classmethod`` form is a function :term:`decorator` -- see
:ref:`function` for details.

A class method can be called either on the class (such as ``C.f()``) or on an instance (such
as ``C().f()``).  The instance is ignored except for its class. If a class
method is called for a derived class, the derived class object is passed as the
implied first argument.

Class methods are different than C++ or Java static methods. If you want those,
see :func:`staticmethod` in this section.
For more information on class methods, see :ref:`types`.

.. versionchanged:: 3.9
   Class methods can now wrap other :term:`descriptors <descriptor>` such as
   :func:`property`.

.. versionchanged:: 3.10
   Class methods now inherit the method attributes (``__module__``,
   ``__name__``, ``__qualname__``, ``__doc__`` and ``__annotations__``) and
   have a new ``__wrapped__`` attribute.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 274)

Unknown directive type "function".

```
.. function:: compile(source, filename, mode, flags=0, dont_inherit=False, optimize=-1)

   Compile the *source* into a code or AST object.  Code objects can be executed
   by :func:`exec` or :func:`eval`.  *source* can either be a normal string, a
   byte string, or an AST object.  Refer to the :mod:`ast` module documentation
   for information on how to work with AST objects.

   The *filename* argument should give the file from which the code was read;
   pass some recognizable value if it wasn't read from a file (``'<string>'`` is
   commonly used).

   The *mode* argument specifies what kind of code must be compiled; it can be
   ``'exec'`` if *source* consists of a sequence of statements, ``'eval'`` if it
   consists of a single expression, or ``'single'`` if it consists of a single
   interactive statement (in the latter case, expression statements that
   evaluate to something other than ``None`` will be printed).

   The optional arguments *flags* and *dont_inherit* control which
   :ref:`compiler options <ast-compiler-flags>` should be activated
   and which :ref:`future features <future>` should be allowed. If neither
   is present (or both are zero) the code is compiled with the same flags that
   affect the code that is calling :func:`compile`. If the *flags*
   argument is given and *dont_inherit* is not (or is zero) then the compiler
   options and the future statements specified by the *flags* argument are used
   in addition to those that would be used anyway. If *dont_inherit* is a
   non-zero integer then the *flags* argument is it -- the flags (future
   features and compiler options) in the surrounding code are ignored.

   Compiler options and future statements are specified by bits which can be
   bitwise ORed together to specify multiple options. The bitfield required to
   specify a given future feature can be found as the
   :attr:`~__future__._Feature.compiler_flag` attribute on the
   :class:`~__future__._Feature` instance in the :mod:`__future__` module.
   :ref:`Compiler flags <ast-compiler-flags>` can be found in :mod:`ast`
   module, with ``PyCF_`` prefix.

   The argument *optimize* specifies the optimization level of the compiler; the
   default value of ``-1`` selects the optimization level of the interpreter as
   given by :option:`-O` options.  Explicit levels are ``0`` (no optimization;
   ``__debug__`` is true), ``1`` (asserts are removed, ``__debug__`` is false)
   or ``2`` (docstrings are removed too).

   This function raises :exc:`SyntaxError` if the compiled source is invalid,
   and :exc:`ValueError` if the source contains null bytes.

   If you want to parse Python code into its AST representation, see
   :func:`ast.parse`.

   .. audit-event:: compile source,filename compile

      Raises an :ref:`auditing event <auditing>` ``compile`` with arguments
      ``source`` and ``filename``. This event may also be raised by implicit
      compilation.

   .. note::

      When compiling a string with multi-line code in ``'single'`` or
      ``'eval'`` mode, input must be terminated by at least one newline
      character.  This is to facilitate detection of incomplete and complete
      statements in the :mod:`code` module.

   .. warning::

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

   .. versionchanged:: 3.2
      Allowed use of Windows and Mac newlines.  Also, input in ``'exec'`` mode
      does not have to end in a newline anymore.  Added the *optimize* parameter.
```

```
.. versionchanged:: 3.5
   Previously, :exc:`TypeError` was raised when null bytes were encountered
   in *source*.

.. versionadded:: 3.8
   ``ast.PyCF_ALLOW_TOP_LEVEL_AWAIT`` can now be passed in flags to enable
   support for top-level ``await``, ``async for``, and ``async with``.
```

Return a complex number with the value *real* + *imag**1j or convert a string or number to a complex number. If the first parameter is a string, it will be interpreted as a complex number and the function must be called without a second parameter. The second parameter can never be a string. Each argument may be any numeric type (including complex). If *imag* is omitted, it defaults to zero and the constructor serves as a numeric conversion like :class:`int` and :class:`float`. If both arguments are omitted, returns `0j`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 356); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 356); *backlink*

Unknown interpreted text role "class".

For a general Python object `x`, `complex(x)` delegates to `x.__complex__()`. If `__complex__()` is not defined then it falls back to :meth:`__float__`. If `__float__()` is not defined then it falls back to :meth:`__index__`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 365); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 365); *backlink*

Unknown interpreted text role "meth".

Note

When converting from a string, the string must not contain whitespace around the central `+` or `-` operator. For example, `complex('1+2j')` is fine, but `complex('1 + 2j')` raises :exc:`ValueError`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 372); *backlink*

Unknown interpreted text role "exc".

The complex type is described in :ref:`typesnumeric`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 377); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 379)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.6
   Grouping digits with underscores as in code literals is allowed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 382)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.8
   Falls back to :meth:`__index__` if :meth:`__complex__` and
   :meth:`__float__` are not defined.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 387)

Unknown directive type "function".

```
.. function:: delattr(object, name)

   This is a relative of :func:`setattr`.  The arguments are an object and a
   string.  The string must be the name of one of the object's attributes.  The
   function deletes the named attribute, provided the object allows it.  For
   example, ``delattr(x, 'foobar')`` is equivalent to ``del x.foobar``.
```

Create a new dictionary. The :class:`dict` object is the dictionary class. See :class:`dict` and :ref:`typesmapping` for documentation about this class.

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "ref".

For other containers see the built-in :class:`list`, :class:`set`, and :class:`tuple` classes, as well as the :mod:`collections` module.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 404); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 404); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 404); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 404); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 408)

Unknown directive type "function".

```
.. function:: dir([object])

   Without arguments, return the list of names in the current local scope.  With an
   argument, attempt to return a list of valid attributes for that object.

   If the object has a method named :meth:`__dir__`, this method will be called and
   must return the list of attributes. This allows objects that implement a custom
   :func:`__getattr__` or :func:`__getattribute__` function to customize the way
   :func:`dir` reports their attributes.

   If the object does not provide :meth:`__dir__`, the function tries its best to
   gather information from the object's :attr:`~object.__dict__` attribute, if defined, and
   from its type object.  The resulting list is not necessarily complete and may
   be inaccurate when the object has a custom :func:`__getattr__`.

   The default :func:`dir` mechanism behaves differently with different types of
   objects, as it attempts to produce the most relevant, rather than complete,
   information:

   * If the object is a module object, the list contains the names of the module's
     attributes.

   * If the object is a type or class object, the list contains the names of its
     attributes, and recursively of the attributes of its bases.

   * Otherwise, the list contains the object's attributes' names, the names of its
     class's attributes, and recursively of the attributes of its class's base
     classes.

   The resulting list is sorted alphabetically.  For example:

      >>> import struct
      >>> dir()   # show the names in the module namespace  # doctest: +SKIP
      ['__builtins__', '__name__', 'struct']
      >>> dir(struct)   # show the names in the struct module # doctest: +SKIP
      ['Struct', '__all__', '__builtins__', '__cached__', '__doc__', '__file__',
       '__initializing__', '__loader__', '__name__', '__package__',
       '_clearcache', 'calcsize', 'error', 'pack', 'pack_into',
       'unpack', 'unpack_from']
      >>> class Shape:
      ...     def __dir__(self):
      ...         return ['area', 'perimeter', 'location']
      >>> s = Shape()
      >>> dir(s)
      ['area', 'location', 'perimeter']

   .. note::

      Because :func:`dir` is supplied primarily as a convenience for use at an
      interactive prompt, it tries to supply an interesting set of names more
      than it tries to supply a rigorously or consistently defined set of names,
      and its detailed behavior may change across releases.  For example,
      metaclass attributes are not in the result list when the argument is a
      class.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 464)

Unknown directive type "function".

```
.. function:: divmod(a, b)

   Take two (non-complex) numbers as arguments and return a pair of numbers
   consisting of their quotient and remainder when using integer division.  With
   mixed operand types, the rules for binary arithmetic operators apply.  For
   integers, the result is the same as ``(a // b, a % b)``. For floating point
   numbers the result is ``(q, a % b)``, where *q* is usually ``math.floor(a /
   b)`` but may be 1 less than that.  In any case ``q * b + a % b`` is very
   close to *a*, if ``a % b`` is non-zero it has the same sign as *b*, and ``0
   <= abs(a % b) < abs(b)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 476)

Unknown directive type "function".

```
.. function:: enumerate(iterable, start=0)

   Return an enumerate object. *iterable* must be a sequence, an
   :term:`iterator`, or some other object which supports iteration.
   The :meth:`~iterator.__next__` method of the iterator returned by
   :func:`enumerate` returns a tuple containing a count (from *start* which
   defaults to 0) and the values obtained from iterating over *iterable*.

      >>> seasons = ['Spring', 'Summer', 'Fall', 'Winter']
      >>> list(enumerate(seasons))
      [(0, 'Spring'), (1, 'Summer'), (2, 'Fall'), (3, 'Winter')]
      >>> list(enumerate(seasons, start=1))
      [(1, 'Spring'), (2, 'Summer'), (3, 'Fall'), (4, 'Winter')]

   Equivalent to::

      def enumerate(sequence, start=0):
          n = start
          for elem in sequence:
              yield n, elem
              n += 1
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 499)

Unknown directive type "function".

```
.. function:: eval(expression[, globals[, locals]])

   The arguments are a string and optional globals and locals.  If provided,
   *globals* must be a dictionary.  If provided, *locals* can be any mapping
   object.

   The *expression* argument is parsed and evaluated as a Python expression
   (technically speaking, a condition list) using the *globals* and *locals*
   dictionaries as global and local namespace.  If the *globals* dictionary is
   present and does not contain a value for the key ``__builtins__``, a
   reference to the dictionary of the built-in module :mod:`builtins` is
   inserted under that key before *expression* is parsed.  That way you can
   control what builtins are available to the executed code by inserting your
   own ``__builtins__`` dictionary into *globals* before passing it to
   :func:`eval`.  If the *locals* dictionary is omitted it defaults to the
   *globals* dictionary.  If both dictionaries are omitted, the expression is
   executed with the *globals* and *locals* in the environment where
   :func:`eval` is called.  Note, *eval()* does not have access to the
   :term:`nested scopes <nested scope>` (non-locals) in the enclosing
   environment.

   The return value is the result of
   the evaluated expression. Syntax errors are reported as exceptions.  Example:

      >>> x = 1
      >>> eval('x+1')
      2

   This function can also be used to execute arbitrary code objects (such as
   those created by :func:`compile`).  In this case, pass a code object instead
   of a string.  If the code object has been compiled with ``'exec'`` as the
   *mode* argument, :func:`eval`\'s return value will be ``None``.

   Hints: dynamic execution of statements is supported by the :func:`exec`
   function.  The :func:`globals` and :func:`locals` functions
   return the current global and local dictionary, respectively, which may be
   useful to pass around for use by :func:`eval` or :func:`exec`.

   If the given source is a string, then leading and trailing spaces and tabs
   are stripped.

   See :func:`ast.literal_eval` for a function that can safely evaluate strings
   with expressions containing only literals.

   .. audit-event:: exec code_object eval

      Raises an :ref:`auditing event <auditing>` ``exec`` with the code object
      as the argument. Code compilation events may also be raised.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 548)

Unknown directive type "index".

```
.. index:: builtin: exec
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 550)

Unknown directive type "function".

```
.. function:: exec(object[, globals[, locals]])

   This function supports dynamic execution of Python code. *object* must be
   either a string or a code object.  If it is a string, the string is parsed as
   a suite of Python statements which is then executed (unless a syntax error
   occurs). [#]_ If it is a code object, it is simply executed.  In all cases,
   the code that's executed is expected to be valid as file input (see the
   section :ref:`file-input` in the Reference Manual). Be aware that the
   :keyword:`nonlocal`, :keyword:`yield`,  and :keyword:`return`
   statements may not be used outside of
   function definitions even within the context of code passed to the
   :func:`exec` function. The return value is ``None``.

   In all cases, if the optional parts are omitted, the code is executed in the
   current scope.  If only *globals* is provided, it must be a dictionary
   (and not a subclass of dictionary), which
   will be used for both the global and the local variables.  If *globals* and
   *locals* are given, they are used for the global and local variables,
   respectively.  If provided, *locals* can be any mapping object.  Remember
   that at the module level, globals and locals are the same dictionary. If exec
   gets two separate objects as *globals* and *locals*, the code will be
   executed as if it were embedded in a class definition.

   If the *globals* dictionary does not contain a value for the key
   ``__builtins__``, a reference to the dictionary of the built-in module
   :mod:`builtins` is inserted under that key.  That way you can control what
   builtins are available to the executed code by inserting your own
   ``__builtins__`` dictionary into *globals* before passing it to :func:`exec`.

   .. audit-event:: exec code_object exec

      Raises an :ref:`auditing event <auditing>` ``exec`` with the code object
      as the argument. Code compilation events may also be raised.

   .. note::

      The built-in functions :func:`globals` and :func:`locals` return the current
      global and local dictionary, respectively, which may be useful to pass around
      for use as the second and third argument to :func:`exec`.

   .. note::

      The default *locals* act as described for function :func:`locals` below:
      modifications to the default *locals* dictionary should not be attempted.
      Pass an explicit *locals* dictionary if you need to see effects of the
      code on *locals* after function :func:`exec` returns.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 598)

Unknown directive type "function".

```
.. function:: filter(function, iterable)

   Construct an iterator from those elements of *iterable* for which *function*
   returns true.  *iterable* may be either a sequence, a container which
   supports iteration, or an iterator.  If *function* is ``None``, the identity
   function is assumed, that is, all elements of *iterable* that are false are
   removed.

   Note that ``filter(function, iterable)`` is equivalent to the generator
   expression ``(item for item in iterable if function(item))`` if function is
   not ``None`` and ``(item for item in iterable if item)`` if function is
   ``None``.

   See :func:`itertools.filterfalse` for the complementary function that returns
   elements of *iterable* for which *function* returns false.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 617)

Unknown directive type "index".

```
.. index::
   single: NaN
   single: Infinity
```

Return a floating point number constructed from a number or string *x*.

If the argument is a string, it should contain a decimal number, optionally preceded by a sign, and optionally embedded in whitespace. The optional sign may be `'+'` or `'-'`; a `'+'` sign has no effect on the value produced. The argument may also be a string representing a NaN (not-a-number), or positive or negative infinity. More precisely, the input must conform to the following grammar after leading and trailing whitespace characters are removed:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 631)

Unknown directive type "productionlist".

```
.. productionlist:: float
   sign: "+" | "-"
   infinity: "Infinity" | "inf"
   nan: "nan"
   numeric_value: `floatnumber` | `infinity` | `nan`
   numeric_string: [`sign`] `numeric_value`
```

Here `floatnumber` is the form of a Python floating-point literal, described in :ref:`floating`. Case is not significant, so, for example, "inf", "Inf", "INFINITY", and "iNfiNity" are all acceptable spellings for positive infinity.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 638); *backlink*

Unknown interpreted text role "ref".

Otherwise, if the argument is an integer or a floating point number, a floating point number with the same value (within Python's floating point precision) is returned. If the argument is outside the range of a Python float, an :exc:`OverflowError` will be raised.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 643); *backlink*

Unknown interpreted text role "exc".

For a general Python object `x`, `float(x)` delegates to `x.__float__()`. If `__float__()` is not defined then it falls back to :meth:`__index__`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 648); *backlink*

Unknown interpreted text role "meth".

If no argument is given, `0.0` is returned.

Examples:

```
>>> float('+1.23')
1.23
>>> float('   -12345\n')
-12345.0
>>> float('1e-003')
0.001
>>> float('+1E6')
1000000.0
>>> float('-Infinity')
-inf
```

The float type is described in :ref:`typesnumeric`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 667); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 669)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.6
   Grouping digits with underscores as in code literals is allowed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 672)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
   *x* is now a positional-only parameter.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 675)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.8
   Falls back to :meth:`__index__` if :meth:`__float__` is not defined.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 679)

Unknown directive type "index".

```
.. index::
   single: __format__
   single: string; format() (built-in function)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 683)

Unknown directive type "function".

```
.. function:: format(value[, format_spec])
```

```
Convert a *value* to a "formatted" representation, as controlled by
*format_spec*.  The interpretation of *format_spec* will depend on the type
of the *value* argument; however, there is a standard formatting syntax that
is used by most built-in types: :ref:`formatspec`.

The default *format_spec* is an empty string which usually gives the same
effect as calling :func:`str(value) <str>`.

A call to ``format(value, format_spec)`` is translated to
``type(value).__format__(value, format_spec)`` which bypasses the instance
dictionary when searching for the value's :meth:`__format__` method.  A
:exc:`TypeError` exception is raised if the method search reaches
:mod:`object` and the *format_spec* is non-empty, or if either the
*format_spec* or the return value are not strings.

.. versionchanged:: 3.4
   ``object().__format__(format_spec)`` raises :exc:`TypeError`
   if *format_spec* is not an empty string.
```

Return a new :class:`frozenset` object, optionally with elements taken from *iterable*. `frozenset` is a built-in class. See :class:`frozenset` and :ref:`types-set` for documentation about this class.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst`, line 709); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst`, line 709); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst`, line 709); *backlink*

Unknown interpreted text role "ref".

For other containers see the built-in :class:`set`, :class:`list`, :class:`tuple`, and :class:`dict` classes, as well as the :mod:`collections` module.

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst`, line 718)

Unknown directive type "function".

```
.. function:: getattr(object, name[, default])

   Return the value of the named attribute of *object*.  *name* must be a string.
   If the string is the name of one of the object's attributes, the result is the
   value of that attribute.  For example, ``getattr(x, 'foobar')`` is equivalent to
   ``x.foobar``.  If the named attribute does not exist, *default* is returned if
   provided, otherwise :exc:`AttributeError` is raised.

   .. note::

      Since :ref:`private name mangling <private-name-mangling>` happens at
      compilation time, one must manually mangle a private attribute's
      (attributes with two leading underscores) name in order to retrieve it with
      :func:`getattr`.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst`, line 734)

Unknown directive type "function".

```
.. function:: globals()

   Return the dictionary implementing the current module namespace. For code within
   functions, this is set when the function is defined and remains the same
   regardless of where the function is called.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 741)

Unknown directive type "function".

```
.. function:: hasattr(object, name)

   The arguments are an object and a string.  The result is ``True`` if the
   string is the name of one of the object's attributes, ``False`` if not. (This
   is implemented by calling ``getattr(object, name)`` and seeing whether it
   raises an :exc:`AttributeError` or not.)
```

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

Unknown directive type "function".

```
.. function:: hash(object)

   Return the hash value of the object (if it has one).  Hash values are
   integers.  They are used to quickly compare dictionary keys during a
   dictionary lookup.  Numeric values that compare equal have the same hash
   value (even if they are of different types, as is the case for 1 and 1.0).

   .. note::

      For objects with custom :meth:`__hash__` methods, note that :func:`hash`
      truncates the return value based on the bit width of the host machine.
      See :meth:`__hash__` for details.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 762)

Unknown directive type "function".

```
.. function:: help([object])

   Invoke the built-in help system.  (This function is intended for interactive
   use.)  If no argument is given, the interactive help system starts on the
   interpreter console.  If the argument is a string, then the string is looked up
   as the name of a module, function, class, method, keyword, or documentation
   topic, and a help page is printed on the console.  If the argument is any other
   kind of object, a help page on the object is generated.

   Note that if a slash(/) appears in the parameter list of a function when
   invoking :func:`help`, it means that the parameters prior to the slash are
   positional-only. For more info, see
   :ref:`the FAQ entry on positional-only parameters <faq-positional-only-arguments>`.

   This function is added to the built-in namespace by the :mod:`site` module.

   .. versionchanged:: 3.4
      Changes to :mod:`pydoc` and :mod:`inspect` mean that the reported
      signatures for callables are now more comprehensive and consistent.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 783)

Unknown directive type "function".

```
.. function:: hex(x)

   Convert an integer number to a lowercase hexadecimal string prefixed with
   "0x". If *x* is not a Python :class:`int` object, it has to define an
   :meth:`__index__` method that returns an integer. Some examples:

      >>> hex(255)
      '0xff'
      >>> hex(-42)
      '-0x2a'

   If you want to convert an integer number to an uppercase or lower hexadecimal
   string with prefix or not, you can use either of the following ways:

     >>> '%#x' % 255, '%x' % 255, '%X' % 255
     ('0xff', 'ff', 'FF')
     >>> format(255, '#x'), format(255, 'x'), format(255, 'X')
     ('0xff', 'ff', 'FF')
     >>> f'{255:#x}', f'{255:x}', f'{255:X}'
     ('0xff', 'ff', 'FF')

   See also :func:`format` for more information.

   See also :func:`int` for converting a hexadecimal string to an
   integer using a base of 16.

   .. note::

      To obtain a hexadecimal string representation for a float, use the
      :meth:`float.hex` method.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 815)

Unknown directive type "function".

```
.. function:: id(object)

   Return the "identity" of an object.  This is an integer which
   is guaranteed to be unique and constant for this object during its lifetime.
   Two objects with non-overlapping lifetimes may have the same :func:`id`
   value.

   .. impl-detail:: This is the address of the object in memory.

   .. audit-event:: builtins.id id id
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 827)

Unknown directive type "function".

```
.. function:: input([prompt])

   If the *prompt* argument is present, it is written to standard output without
   a trailing newline.  The function then reads a line from input, converts it
   to a string (stripping a trailing newline), and returns that.  When EOF is
   read, :exc:`EOFError` is raised.  Example::

      >>> s = input('--> ')  # doctest: +SKIP
      --> Monty Python's Flying Circus
      >>> s  # doctest: +SKIP
      "Monty Python's Flying Circus"

   If the :mod:`readline` module was loaded, then :func:`input` will use it
   to provide elaborate line editing and history features.

   .. audit-event:: builtins.input prompt input

      Raises an :ref:`auditing event <auditing>` ``builtins.input`` with
      argument ``prompt`` before reading input

   .. audit-event:: builtins.input/result result input

      Raises an auditing event ``builtins.input/result`` with the result after
      successfully reading input.
```

Return an integer object constructed from a number or string *x*, or return `0` if no arguments are given. If *x* defines :meth:`__int__`, `int(x)` returns `x.__int__()`. If *x* defines :meth:`__index__`, it returns `x.__index__()`. If *x* defines :meth:`__trunc__`, it returns `x.__trunc__()`. For floating point numbers, this truncates towards zero.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 856); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 856); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 856); *backlink*

Unknown interpreted text role "meth".

If *x* is not a number or if *base* is given, then *x* must be a string, :class:`bytes`, or :class:`bytearray` instance representing an :ref:`integer literal <integers>` in radix *base*. Optionally, the literal can be preceded by `+` or `-` (with no space in between) and surrounded by whitespace. A base-n literal consists of the digits 0 to n-1, with `a` to `z` (or `A` to `Z`) having values 10 to 35. The default *base* is 10. The allowed values are 0 and 2--36. Base-2, -8, and -16 literals can be optionally prefixed with `0b`/`0B`, `0o`/`0O`, or `0x`/`0X`, as with integer literals in code. Base 0 means to interpret exactly as a code literal, so that the actual base is 2, 8, 10, or 16, and so that `int('010', 0)` is not legal, while `int('010')` is, as well as `int('010', 8)`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 863); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 863); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 863); *backlink*

Unknown interpreted text role "ref".

The integer type is described in :ref:`typesnumeric`.

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

main\Doc\library\[cpython-main][Doc][library]functions.rst, line 876); *backlink*

Unknown interpreted text role "ref".

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

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.4
   If *base* is not an instance of :class:`int` and the *base* object has a
   :meth:`base.__index__ <object.__index__>` method, that method is called
   to obtain an integer for the base.  Previous versions used
   :meth:`base.__int__ <object.__int__>` instead of :meth:`base.__index__
   <object.__index__>`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 885)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.6
   Grouping digits with underscores as in code literals is allowed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 888)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
   *x* is now a positional-only parameter.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 891)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.8
   Falls back to :meth:`__index__` if :meth:`__int__` is not defined.
```

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

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.11
   The delegation to :meth:`__trunc__` is deprecated.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 898)

Unknown directive type "function".

```
.. function:: isinstance(object, classinfo)

   Return ``True`` if the *object* argument is an instance of the *classinfo*
   argument, or of a (direct, indirect, or :term:`virtual <abstract base
   class>`) subclass thereof.  If *object* is not
   an object of the given type, the function always returns ``False``.
   If *classinfo* is a tuple of type objects (or recursively, other such
   tuples) or a :ref:`types-union` of multiple types, return ``True`` if
   *object* is an instance of any of the types.
   If *classinfo* is not a type or tuple of types and such tuples,
   a :exc:`TypeError` exception is raised.

   .. versionchanged:: 3.10
      *classinfo* can be a :ref:`types-union`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 914)

Unknown directive type "function".

```
.. function:: issubclass(class, classinfo)

   Return ``True`` if *class* is a subclass (direct, indirect, or :term:`virtual
   <abstract base class>`) of *classinfo*.  A
   class is considered a subclass of itself. *classinfo* may be a tuple of class
   objects or a :ref:`types-union`, in which case return ``True`` if *class* is a
   subclass of any entry in *classinfo*.  In any other case, a :exc:`TypeError`
   exception is raised.

   .. versionchanged:: 3.10
      *classinfo* can be a :ref:`types-union`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 927)

Unknown directive type "function".

```
.. function:: iter(object[, sentinel])
```

```
Return an :term:`iterator` object.  The first argument is interpreted very
differently depending on the presence of the second argument. Without a
second argument, *object* must be a collection object which supports the
:term:`iterable` protocol (the :meth:`__iter__` method), or it must support
the sequence protocol (the :meth:`__getitem__` method with integer arguments
starting at ``0``).  If it does not support either of those protocols,
:exc:`TypeError` is raised. If the second argument, *sentinel*, is given,
then *object* must be a callable object.  The iterator created in this case
will call *object* with no arguments for each call to its
:meth:`~iterator.__next__` method; if the value returned is equal to
*sentinel*, :exc:`StopIteration` will be raised, otherwise the value will
be returned.

See also :ref:`typeiter`.

One useful application of the second form of :func:`iter` is to build a
block-reader. For example, reading fixed-width blocks from a binary
database file until the end of file is reached::

   from functools import partial
   with open('mydata.db', 'rb') as f:
       for block in iter(partial(f.read, 64), b''):
           process_block(block)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 954)

Unknown directive type "function".

```
.. function:: len(s)

   Return the length (the number of items) of an object.  The argument may be a
   sequence (such as a string, bytes, tuple, list, or range) or a collection
   (such as a dictionary, set, or frozen set).

   .. impl-detail::

      ``len`` raises :exc:`OverflowError` on lengths larger than
      :data:`sys.maxsize`, such as :class:`range(2 ** 100) <range>`.
```

Rather than being a function, :class:`list` is actually a mutable sequence type, as documented in :ref:`typesseq-list` and :ref:`typesseq`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 970); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 970); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 970); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 974)

Unknown directive type "function".

```
.. function:: locals()

   Update and return a dictionary representing the current local symbol table.
   Free variables are returned by :func:`locals` when it is called in function
   blocks, but not in class blocks. Note that at the module level, :func:`locals`
   and :func:`globals` are the same dictionary.

   .. note::
      The contents of this dictionary should not be modified; changes may not
      affect the values of local and free variables used by the interpreter.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 985)

Unknown directive type "function".

```
.. function:: map(function, iterable, ...)

   Return an iterator that applies *function* to every item of *iterable*,
   yielding the results.  If additional *iterable* arguments are passed,
   *function* must take that many arguments and is applied to the items from all
   iterables in parallel.  With multiple iterables, the iterator stops when the
   shortest iterable is exhausted.  For cases where the function inputs are
   already arranged into argument tuples, see :func:`itertools.starmap`\.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 995)

Unknown directive type "function".

```
.. function:: max(iterable, *[, key, default])
```

```
              max(arg1, arg2, *args[, key])

      Return the largest item in an iterable or the largest of two or more
      arguments.

      If one positional argument is provided, it should be an :term:`iterable`.
      The largest item in the iterable is returned.  If two or more positional
      arguments are provided, the largest of the positional arguments is
      returned.

      There are two optional keyword-only arguments. The *key* argument specifies
      a one-argument ordering function like that used for :meth:`list.sort`. The
      *default* argument specifies an object to return if the provided iterable is
      empty. If the iterable is empty and *default* is not provided, a
      :exc:`ValueError` is raised.

      If multiple items are maximal, the function returns the first one
      encountered.  This is consistent with other sort-stability preserving tools
      such as ``sorted(iterable, key=keyfunc, reverse=True)[0]`` and
      ``heapq.nlargest(1, iterable, key=keyfunc)``.

      .. versionadded:: 3.4
         The *default* keyword-only argument.

      .. versionchanged:: 3.8
         The *key* can be ``None``.
```

Return a "memory view" object created from the given argument. See :ref:`typememoryview` for more information.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1028); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1032)

Unknown directive type "function".

```
   .. function:: min(iterable, *[, key, default])
                 min(arg1, arg2, *args[, key])

      Return the smallest item in an iterable or the smallest of two or more
      arguments.

      If one positional argument is provided, it should be an :term:`iterable`.
      The smallest item in the iterable is returned.  If two or more positional
      arguments are provided, the smallest of the positional arguments is
      returned.

      There are two optional keyword-only arguments. The *key* argument specifies
      a one-argument ordering function like that used for :meth:`list.sort`. The
      *default* argument specifies an object to return if the provided iterable is
      empty. If the iterable is empty and *default* is not provided, a
      :exc:`ValueError` is raised.

      If multiple items are minimal, the function returns the first one
      encountered.  This is consistent with other sort-stability preserving tools
      such as ``sorted(iterable, key=keyfunc)[0]`` and ``heapq.nsmallest(1,
      iterable, key=keyfunc)``.

      .. versionadded:: 3.4
         The *default* keyword-only argument.

      .. versionchanged:: 3.8
         The *key* can be ``None``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1061)

Unknown directive type "function".

```
   .. function:: next(iterator[, default])

      Retrieve the next item from the :term:`iterator` by calling its
      :meth:`~iterator.__next__` method.  If *default* is given, it is returned
      if the iterator is exhausted, otherwise :exc:`StopIteration` is raised.
```

Return a new featureless object. :class:`object` is a base for all classes. It has methods that are common to all instances of Python classes. This function does not accept any arguments.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1070); *backlink*

Unknown interpreted text role "class".

Note

:class:`object` does *not* have a :attr:`~object.__dict__`, so you can't assign arbitrary attributes to an instance of the :class:`object` class.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1076); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1076); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1076); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1080)

Unknown directive type "function".

```
.. function:: oct(x)

   Convert an integer number to an octal string prefixed with "0o".  The result
   is a valid Python expression. If *x* is not a Python :class:`int` object, it
   has to define an :meth:`__index__` method that returns an integer. For
   example:

       >>> oct(8)
       '0o10'
       >>> oct(-56)
       '-0o70'

   If you want to convert an integer number to an octal string either with the prefix
   "0o" or not, you can use either of the following ways.

       >>> '%#o' % 10, '%o' % 10
       ('0o12', '12')
       >>> format(10, '#o'), format(10, 'o')
       ('0o12', '12')
       >>> f'{10:#o}', f'{10:o}'
       ('0o12', '12')

   See also :func:`format` for more information.

    .. index::
       single: file object; open() built-in function
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1107)

Unknown directive type "function".

```
.. function:: open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None

   Open *file* and return a corresponding :term:`file object`.  If the file
   cannot be opened, an :exc:`OSError` is raised. See
   :ref:`tut-files` for more examples of how to use this function.

   *file* is a :term:`path-like object` giving the pathname (absolute or
   relative to the current working directory) of the file to be opened or an
   integer file descriptor of the file to be wrapped.  (If a file descriptor is
   given, it is closed when the returned I/O object is closed unless *closefd*
   is set to ``False``.)

   *mode* is an optional string that specifies the mode in which the file is
   opened.  It defaults to ```'r'``` which means open for reading in text mode.
   Other common values are ```'w'``` for writing (truncating the file if it
   already exists), ```'x'``` for exclusive creation, and ```'a'``` for appending
   (which on *some* Unix systems, means that *all* writes append to the end of
   the file regardless of the current seek position).  In text mode, if
   *encoding* is not specified the encoding used is platform-dependent:
   ``locale.getpreferredencoding(False)`` is called to get the current locale
   encoding. (For reading and writing raw bytes use binary mode and leave
   *encoding* unspecified.)  The available modes are:

   .. _filemodes:

   .. index::
      pair: file; modes

   ========= ===========================================================
   Character Meaning
   ========= ===========================================================
   ```'r'```   open for reading (default)
   ```'w'```   open for writing, truncating the file first
   ```'x'```   open for exclusive creation, failing if the file already exists
   ```'a'```   open for writing, appending to the end of file if it exists
   ```'b'```   binary mode
   ```'t'```   text mode (default)
   ```'+'```   open for updating (reading and writing)
   ========= ===

   The default mode is ```'r'``` (open for reading text, a synonym of ```'rt'```).
   Modes ```'w+'``` and ```'w+b'``` open and truncate the file. Modes ```'r+'```
   and ```'r+b'``` open the file with no truncation.

   As mentioned in the :ref:`io-overview`, Python distinguishes between binary
   and text I/O. Files opened in binary mode (including ```'b'``` in the *mode*
   argument) return contents as :class:`bytes` objects without any decoding. In
   text mode (the default, or when ```'t'``` is included in the *mode* argument),
   the contents of the file are returned as :class:`str`, the bytes having been
```
```

```
first decoded using a platform-dependent encoding or using the specified
*encoding* if given.

.. note::

   Python doesn't depend on the underlying operating system's notion of text
   files; all the processing is done by Python itself, and is therefore
   platform-independent.

*buffering* is an optional integer used to set the buffering policy.  Pass 0
to switch buffering off (only allowed in binary mode), 1 to select line
buffering (only usable in text mode), and an integer > 1 to indicate the size
in bytes of a fixed-size chunk buffer.  When no *buffering* argument is
given, the default buffering policy works as follows:

* Binary files are buffered in fixed-size chunks; the size of the buffer is
  chosen using a heuristic trying to determine the underlying device's "block
  size" and falling back on :attr:`io.DEFAULT_BUFFER_SIZE`.  On many systems,
  the buffer will typically be 4096 or 8192 bytes long.

* "Interactive" text files (files for which :meth:`~io.IOBase.isatty`
  returns ``True``) use line buffering.  Other text files use the policy
  described above for binary files.

*encoding* is the name of the encoding used to decode or encode the file.
This should only be used in text mode.  The default encoding is platform
dependent (whatever :func:`locale.getpreferredencoding` returns), but any
:term:`text encoding` supported by Python
can be used.  See the :mod:`codecs` module for
the list of supported encodings.

*errors* is an optional string that specifies how encoding and decoding
errors are to be handledâ€”this cannot be used in binary mode.
A variety of standard error handlers are available
(listed under :ref:`error-handlers`), though any
error handling name that has been registered with
:func:`codecs.register_error` is also valid.  The standard names
include:

* ``'strict'`` to raise a :exc:`ValueError` exception if there is
  an encoding error.  The default value of ``None`` has the same
  effect.

* ``'ignore'`` ignores errors.  Note that ignoring encoding errors
  can lead to data loss.

* ``'replace'`` causes a replacement marker (such as ``'?'``) to be inserted
  where there is malformed data.

* ``'surrogateescape'`` will represent any incorrect bytes as low
  surrogate code units ranging from U+DC80 to U+DCFF.
  These surrogate code units will then be turned back into
  the same bytes when the ``surrogateescape`` error handler is used
  when writing data.  This is useful for processing files in an
  unknown encoding.

* ``'xmlcharrefreplace'`` is only supported when writing to a file.
  Characters not supported by the encoding are replaced with the
  appropriate XML character reference ``&#nnn;``.

* ``'backslashreplace'`` replaces malformed data by Python's backslashed
  escape sequences.

* ``'namereplace'`` (also only supported when writing)
  replaces unsupported characters with ``\N{...}`` escape sequences.

.. index::
   single: universal newlines; open() built-in function

.. _open-newline-parameter:

*newline* controls how :term:`universal newlines` mode works (it only
applies to text mode).  It can be ``None``, ``''``, ``'\n'``, ``'\r'``, and
``'\r\n'``.  It works as follows:

* When reading input from the stream, if *newline* is ``None``, universal
  newlines mode is enabled.  Lines in the input can end in ``'\n'``,
  ``'\r'``, or ``'\r\n'``, and these are translated into ``'\n'`` before
  being returned to the caller.  If it is ``''``, universal newlines mode is
  enabled, but line endings are returned to the caller untranslated.  If it
  has any of the other legal values, input lines are only terminated by the
  given string, and the line ending is returned to the caller untranslated.

* When writing output to the stream, if *newline* is ``None``, any ``'\n'``
  characters written are translated to the system default line separator,
  :data:`os.linesep`.  If *newline* is ``''`` or ``'\n'``, no translation
  takes place.  If *newline* is any of the other legal values, any ``'\n'``
  characters written are translated to the given string.

If *closefd* is ``False`` and a file descriptor rather than a filename was
given, the underlying file descriptor will be kept open when the file is
closed.  If a filename is given *closefd* must be ``True`` (the default);
otherwise, an error will be raised.

A custom opener can be used by passing a callable as *opener*. The underlying
file descriptor for the file object is then obtained by calling *opener* with
(*file*, *flags*). *opener* must return an open file descriptor (passing
:mod:`os.open` as *opener* results in functionality similar to passing
``None``).

The newly created file is :ref:`non-inheritable <fd_inheritance>`.

The following example uses the :ref:`dir_fd <dir_fd>` parameter of the
:func:`os.open` function to open a file relative to a given directory::

   >>> import os
   >>> dir_fd = os.open('somedir', os.O_RDONLY)
   >>> def opener(path, flags):
   ...     return os.open(path, flags, dir_fd=dir_fd)
```

```
   ...
   >>> with open('spamspam.txt', 'w', opener=opener) as f:
   ...     print('This will be written to somedir/spamspam.txt', file=f)
   ...
   >>> os.close(dir_fd)  # don't leak a file descriptor

The type of :term:`file object` returned by the :func:`open` function
depends on the mode.  When :func:`open` is used to open a file in a text
mode (``'w'``, ``'r'``, ``'wt'``, ``'rt'``, etc.), it returns a subclass of
:class:`io.TextIOBase` (specifically :class:`io.TextIOWrapper`).  When used
to open a file in a binary mode with buffering, the returned class is a
subclass of :class:`io.BufferedIOBase`.  The exact class varies: in read
binary mode, it returns an :class:`io.BufferedReader`; in write binary and
append binary modes, it returns an :class:`io.BufferedWriter`, and in
read/write mode, it returns an :class:`io.BufferedRandom`.  When buffering is
disabled, the raw stream, a subclass of :class:`io.RawIOBase`,
:class:`io.FileIO`, is returned.

.. index::
   single: line-buffered I/O
   single: unbuffered I/O
   single: buffer size, I/O
   single: I/O control; buffering
   single: binary mode
   single: text mode
   module: sys

See also the file handling modules, such as :mod:`fileinput`, :mod:`io`
(where :func:`open` is declared), :mod:`os`, :mod:`os.path`, :mod:`tempfile`,
and :mod:`shutil`.

.. audit-event:: open file,mode,flags open

The ``mode`` and ``flags`` arguments may have been modified or inferred from
the original call.

.. versionchanged:: 3.3

      * The *opener* parameter was added.
      * The ``'x'`` mode was added.
      * :exc:`IOError` used to be raised, it is now an alias of :exc:`OSError`.
      * :exc:`FileExistsError` is now raised if the file opened in exclusive
        creation mode (``'x'``) already exists.

.. versionchanged:: 3.4

      * The file is now non-inheritable.

.. versionchanged:: 3.5

      * If the system call is interrupted and the signal handler does not raise an
        exception, the function now retries the system call instead of raising an
        :exc:`InterruptedError` exception (see :pep:`475` for the rationale).
      * The ``'namereplace'`` error handler was added.

.. versionchanged:: 3.6

      * Support added to accept objects implementing :class:`os.PathLike`.
      * On Windows, opening a console buffer may return a subclass of
        :class:`io.RawIOBase` other than :class:`io.FileIO`.

.. versionchanged:: 3.11
   The ``'U'`` mode has been removed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1329)

Unknown directive type "function".

```
.. function:: ord(c)

   Given a string representing one Unicode character, return an integer
   representing the Unicode code point of that character.  For example,
   ``ord('a')`` returns the integer ``97`` and ``ord('â,¬')`` (Euro sign)
   returns ``8364``.  This is the inverse of :func:`chr`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1337)

Unknown directive type "function".

```
.. function:: pow(base, exp[, mod])

   Return *base* to the power *exp*; if *mod* is present, return *base* to the
   power *exp*, modulo *mod* (computed more efficiently than
   ``pow(base, exp) % mod``). The two-argument form ``pow(base, exp)`` is
   equivalent to using the power operator: ``base**exp``.

   The arguments must have numeric types.  With mixed operand types, the
   coercion rules for binary arithmetic operators apply.  For :class:`int`
   operands, the result has the same type as the operands (after coercion)
   unless the second argument is negative; in that case, all arguments are
   converted to float and a float result is delivered.  For example, ``pow(10, 2)``
   returns ``100``, but ``pow(10, -2)`` returns ``0.01``.  For a negative base of
   type :class:`int` or :class:`float` and a non-integral exponent, a complex
   result is delivered.  For example, ``pow(-9, 0.5)`` returns a value close
   to ``3j``.

   For :class:`int` operands *base* and *exp*, if *mod* is present, *mod* must
   also be of integer type and *mod* must be nonzero. If *mod* is present and
   *exp* is negative, *base* must be relatively prime to *mod*. In that case,
   ``pow(inv_base, -exp, mod)`` is returned, where *inv_base* is an inverse to
   *base* modulo *mod*.
```

```
      Here's an example of computing an inverse for ``38`` modulo ``97``::

         >>> pow(38, -1, mod=97)
         23
         >>> 23 * 38 % 97 == 1
         True

      .. versionchanged:: 3.8
         For :class:`int` operands, the three-argument form of ``pow`` now allows
         the second argument to be negative, permitting computation of modular
         inverses.

      .. versionchanged:: 3.8
         Allow keyword arguments.  Formerly, only positional arguments were
         supported.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1377)

Unknown directive type "function".

```
   .. function:: print(*objects, sep=' ', end='\\n', file=sys.stdout, flush=False)

      Print *objects* to the text stream *file*, separated by *sep* and followed
      by *end*.  *sep*, *end*, *file*, and *flush*, if present, must be given as keyword
      arguments.

      All non-keyword arguments are converted to strings like :func:`str` does and
      written to the stream, separated by *sep* and followed by *end*.  Both *sep*
      and *end* must be strings; they can also be ``None``, which means to use the
      default values.  If no *objects* are given, :func:`print` will just write
      *end*.

      The *file* argument must be an object with a ``write(string)`` method; if it
      is not present or ``None``, :data:`sys.stdout` will be used.  Since printed
      arguments are converted to text strings, :func:`print` cannot be used with
      binary mode file objects.  For these, use ``file.write(...)`` instead.

      Whether the output is buffered is usually determined by *file*, but if the
      *flush* keyword argument is true, the stream is forcibly flushed.

      .. versionchanged:: 3.3
         Added the *flush* keyword argument.
```

Return a property attribute.

fget is a function for getting an attribute value. *fset* is a function for setting an attribute value. *fdel* is a function for deleting an attribute value. And *doc* creates a docstring for the attribute.

A typical use is to define a managed attribute `x`:

```
class C:
    def __init__(self):
        self._x = None

    def getx(self):
        return self._x

    def setx(self, value):
        self._x = value

    def delx(self):
        del self._x

    x = property(getx, setx, delx, "I'm the 'x' property.")
```

If *c* is an instance of *C*, `c.x` will invoke the getter, `c.x = value` will invoke the setter, and `del c.x` the deleter.

If given, *doc* will be the docstring of the property attribute. Otherwise, the property will copy *fget*'s docstring (if it exists). This makes it possible to create read-only properties easily using :func:`property` as a :term:`decorator`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1429); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1429); *backlink*

Unknown interpreted text role "term".

```
class Parrot:
    def __init__(self):
        self._voltage = 100000

    @property
    def voltage(self):
        """Get the current voltage."""
        return self._voltage
```

The `@property` decorator turns the :meth:`voltage` method into a "getter" for a read-only attribute with the same name, and it sets the docstring for *voltage* to "Get the current voltage."

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1442); *backlink*

Unknown interpreted text role "meth".

A property object has :attr:`~property.getter`, :attr:`~property.setter`, and :attr:`~property.deleter` methods usable as decorators that

create a copy of the property with the corresponding accessor function set to the decorated function. This is best explained with an example:

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

Unknown interpreted text role "attr".

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

Unknown interpreted text role "attr".

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

Unknown interpreted text role "attr".

```
class C:
    def __init__(self):
        self._x = None

    @property
    def x(self):
        """I'm the 'x' property."""
        return self._x

    @x.setter
    def x(self, value):
        self._x = value

    @x.deleter
    def x(self):
        del self._x
```

This code is exactly equivalent to the first example. Be sure to give the additional functions the same name as the original property (`x` in this case.)

The returned property object also has the attributes `fget`, `fset`, and `fdel` corresponding to the constructor arguments.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1475)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.5
   The docstrings of property objects are now writeable.
```

Rather than being a function, :class:`range` is actually an immutable sequence type, as documented in :ref:`typesseq-range` and :ref:`typesseq`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1484); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1484); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1484); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1488)

Unknown directive type "function".

```
.. function:: repr(object)

   Return a string containing a printable representation of an object.  For many
   types, this function makes an attempt to return a string that would yield an
   object with the same value when passed to :func:`eval`; otherwise, the
   representation is a string enclosed in angle brackets that contains the name
   of the type of the object together with additional information often
   including the name and address of the object.  A class can control what this
   function returns for its instances by defining a :meth:`__repr__` method.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1499)

Unknown directive type "function".

```
.. function:: reversed(seq)

   Return a reverse :term:`iterator`.  *seq* must be an object which has
   a :meth:`__reversed__` method or supports the sequence protocol (the
   :meth:`__len__` method and the :meth:`__getitem__` method with integer
   arguments starting at ``0``).
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]functions.rst`, line 1507)

Unknown directive type "function".

```
.. function:: round(number[, ndigits])

   Return *number* rounded to *ndigits* precision after the decimal
   point.  If *ndigits* is omitted or is ``None``, it returns the
   nearest integer to its input.

   For the built-in types supporting :func:`round`, values are rounded to the
   closest multiple of 10 to the power minus *ndigits*; if two multiples are
   equally close, rounding is done toward the even choice (so, for example,
   both ``round(0.5)`` and ``round(-0.5)`` are ``0``, and ``round(1.5)`` is
   ``2``).  Any integer value is valid for *ndigits* (positive, zero, or
   negative).  The return value is an integer if *ndigits* is omitted or
   ``None``.
   Otherwise, the return value has the same type as *number*.

   For a general Python object ``number``, ``round`` delegates to
   ``number.__round__``.

   .. note::

      The behavior of :func:`round` for floats can be surprising: for example,
      ``round(2.675, 2)`` gives ``2.67`` instead of the expected ``2.68``.
      This is not a bug: it's a result of the fact that most decimal fractions
      can't be represented exactly as a float.  See :ref:`tut-fp-issues` for
      more information.
```

Return a new :class:`set` object, optionally with elements taken from *iterable*. `set` is a built-in class. See :class:`set` and :ref:`types-set` for documentation about this class.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]functions.rst`, line 1538); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]functions.rst`, line 1538); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]functions.rst`, line 1538); *backlink*

Unknown interpreted text role "ref".

For other containers see the built-in :class:`frozenset`, :class:`list`, :class:`tuple`, and :class:`dict` classes, as well as the :mod:`collections` module.

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]functions.rst`, line 1547)

Unknown directive type "function".

```
.. function:: setattr(object, name, value)

   This is the counterpart of :func:`getattr`.  The arguments are an object, a
   string, and an arbitrary value.  The string may name an existing attribute or a
   new attribute.  The function assigns the value to the attribute, provided the
   object allows it.  For example, ``setattr(x, 'foobar', 123)`` is equivalent to
   ``x.foobar = 123``.

   .. note::
```

```
Since :ref:`private name mangling <private-name-mangling>` happens at
compilation time, one must manually mangle a private attribute's
(attributes with two leading underscores) name in order to set it with
:func:`setattr`.
```

Return a :term:`slice` object representing the set of indices specified by `range(start, stop, step)`. The *start* and *step* arguments default to `None`. Slice objects have read-only data attributes :attr:`~slice.start`, :attr:`~slice.stop`, and :attr:`~slice.step` which merely return the argument values (or their default). They have no other explicit functionality; however, they are used by NumPy and other third-party packages. Slice objects are also generated when extended indexing syntax is used. For example: `a[start:stop:step]` or `a[start:stop, i]`. See :func:`itertools.islice` for an alternate version that returns an iterator.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1566); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1566); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1566); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1566); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1566); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1577)

Unknown directive type "function".

```
.. function:: sorted(iterable, /, *, key=None, reverse=False)

   Return a new sorted list from the items in *iterable*.

   Has two optional arguments which must be specified as keyword arguments.

   *key* specifies a function of one argument that is used to extract a comparison
   key from each element in *iterable* (for example, ``key=str.lower``).  The
   default value is ``None`` (compare the elements directly).

   *reverse* is a boolean value.  If set to ``True``, then the list elements are
   sorted as if each comparison were reversed.

   Use :func:`functools.cmp_to_key` to convert an old-style *cmp* function to a
   *key* function.

   The built-in :func:`sorted` function is guaranteed to be stable. A sort is
   stable if it guarantees not to change the relative order of elements that
   compare equal --- this is helpful for sorting in multiple passes (for
   example, sort by department, then by salary grade).

   The sort algorithm uses only ``<`` comparisons between items.  While
   defining an :meth:`~object.__lt__` method will suffice for sorting,
   :PEP:`8` recommends that all six :ref:`rich comparisons
   <comparisons>` be implemented.  This will help avoid bugs when using
   the same data with other ordering tools such as :func:`max` that rely
   on a different underlying method.  Implementing all six comparisons
   also helps avoid confusion for mixed type comparisons which can call
   reflected the :meth:`~object.__gt__` method.

   For sorting examples and a brief sorting tutorial, see :ref:`sortinghowto`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1609)

Unknown directive type "decorator".

```
.. decorator:: staticmethod

   Transform a method into a static method.

   A static method does not receive an implicit first argument. To declare a static
   method, use this idiom::

      class C:
          @staticmethod
          def f(arg1, arg2, ...): ...

   The ``@staticmethod`` form is a function :term:`decorator` -- see
   :ref:`function` for details.

   A static method can be called either on the class (such as ``C.f()``) or on
   an instance (such as ``C().f()``). Moreover, they can be called as regular
   functions (such as ``f()``).
```

```
Static methods in Python are similar to those found in Java or C++. Also, see
:func:`classmethod` for a variant that is useful for creating alternate class
constructors.

Like all decorators, it is also possible to call ``staticmethod`` as
a regular function and do something with its result.  This is needed
in some cases where you need a reference to a function from a class
body and you want to avoid the automatic transformation to instance
method.  For these cases, use this idiom::

   def regular_function():
       ...

   class C:
       method = staticmethod(regular_function)

For more information on static methods, see :ref:`types`.

.. versionchanged:: 3.10
   Static methods now inherit the method attributes (``__module__``,
   ``__name__``, ``__qualname__``, ``__doc__`` and ``__annotations__``),
   have a new ``__wrapped__`` attribute, and are now callable as regular
   functions.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1652)

Unknown directive type "index".

```
.. index::
   single: string; str() (built-in function)
```

Return a :class:`str` version of *object*. See :func:`str` for details.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1660); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1660); *backlink*

Unknown interpreted text role "func".

`str` is the built-in string :term:`class`. For general information about strings, see :ref:`textseq`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1662); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1662); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1666)

Unknown directive type "function".

```
.. function:: sum(iterable, /, start=0)

   Sums *start* and the items of an *iterable* from left to right and returns the
   total.  The *iterable*'s items are normally numbers, and the start value is not
   allowed to be a string.

   For some use cases, there are good alternatives to :func:`sum`.
   The preferred, fast way to concatenate a sequence of strings is by calling
   ``''.join(sequence)``.  To add floating point values with extended precision,
   see :func:`math.fsum`\.  To concatenate a series of iterables, consider using
   :func:`itertools.chain`.

   .. versionchanged:: 3.8
      The *start* parameter can be specified as a keyword argument.
```

Return a proxy object that delegates method calls to a parent or sibling class of *type*. This is useful for accessing inherited methods that have been overridden in a class.

The *object-or-type* determines the :term:`method resolution order` to be searched. The search starts from the class right after the *type*.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1687); *backlink*

Unknown interpreted text role "term".

For example, if :attr:`~class.__mro__` of *object-or-type* is `D -> B -> C -> A -> object` and the value of *type* is `B`, then :func:`super` searches `C -> A -> object`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1691); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1691); backlink

Unknown interpreted text role "func".

The :attr:`~class.__mro__` attribute of the *object-or-type* lists the method resolution search order used by both :func:`getattr` and :func:`super`. The attribute is dynamic and can change whenever the inheritance hierarchy is updated.

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

Unknown interpreted text role "attr".

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

Unknown interpreted text role "func".

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

Unknown interpreted text role "func".

If the second argument is omitted, the super object returned is unbound. If the second argument is an object, `isinstance(obj, type)` must be true. If the second argument is a type, `issubclass(type2, type)` must be true (this is useful for classmethods).

There are two typical use cases for *super*. In a class hierarchy with single inheritance, *super* can be used to refer to parent classes without naming them explicitly, thus making the code more maintainable. This use closely parallels the use of *super* in other programming languages.

The second use case is to support cooperative multiple inheritance in a dynamic execution environment. This use case is unique to Python and is not found in statically compiled languages or languages that only support single inheritance. This makes it possible to implement "diamond diagrams" where multiple base classes implement the same method. Good design dictates that such implementations have the same calling signature in every case (because the order of calls is determined at runtime, because that order adapts to changes in the class hierarchy, and because that order can include sibling classes that are unknown prior to runtime).

For both use cases, a typical superclass call looks like this:

```
class C(B):
    def method(self, arg):
        super().method(arg)    # This does the same thing as:
                               # super(C, self).method(arg)
```

In addition to method lookups, :func:`super` also works for attribute lookups. One possible use case for this is calling :term:`descriptors <descriptor>` in a parent or sibling class.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1727); backlink

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1727); backlink

Unknown interpreted text role "term".

Note that :func:`super` is implemented as part of the binding process for explicit dotted attribute lookups such as `super().__getitem__(name)`. It does so by implementing its own :meth:`__getattribute__` method for searching classes in a predictable order that supports cooperative multiple inheritance. Accordingly, :func:`super` is undefined for implicit lookups using statements or operators such as `super()[name]`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1731); backlink

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1731); backlink

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1731); backlink

Unknown interpreted text role "func".

Also note that, aside from the zero argument form, :func:`super` is not limited to use inside methods. The two argument form specifies the arguments exactly and makes the appropriate references. The zero argument form only works inside a class definition, as the compiler fills in the necessary details to correctly retrieve the class being defined, as well as accessing the current instance for ordinary methods.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1738); backlink

Unknown interpreted text role "func".

For practical suggestions on how to design cooperative classes using :func:`super`, see guide to using super().

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1745); *backlink*

Unknown interpreted text role "func".

Rather than being a function, :class:`tuple` is actually an immutable sequence type, as documented in :ref:`typesseq-tuple` and :ref:`typesseq`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1754); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1754); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1754); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1761)

Unknown directive type "index".

```
.. index:: object: type
```

With one argument, return the type of an *object*. The return value is a type object and generally the same object as returned by :attr:`object.__class__ <instance.__class__>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1763); *backlink*

Unknown interpreted text role "attr".

The :func:`isinstance` built-in function is recommended for testing the type of an object, because it takes subclasses into account.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1767); *backlink*

Unknown interpreted text role "func".

With three arguments, return a new type object. This is essentially a dynamic form of the :keyword:`class` statement. The *name* string is the class name and becomes the :attr:`~definition.__name__` attribute. The *bases* tuple contains the base classes and becomes the :attr:`~class.__bases__` attribute; if empty, :class:`object`, the ultimate base of all classes, is added. The *dict* dictionary contains attribute and method definitions for the class body; it may be copied or wrapped before becoming the :attr:`~object.__dict__` attribute. The following two statements create identical :class:`type` objects:

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

Unknown interpreted text role "keyword".

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

Unknown interpreted text role "attr".

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

Unknown interpreted text role "attr".

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

Unknown interpreted text role "attr".

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

Unknown interpreted text role "class".

```
>>> class X:
...     a = 1
...
>>> X = type('X', (), dict(a=1))
```

See also :ref:`bltin-type-objects`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1786); *backlink*

Unknown interpreted text role "ref".

Keyword arguments provided to the three argument form are passed to the appropriate metaclass machinery (usually :meth:`~object.__init_subclass__`) in the same way that keywords in a class definition (besides *metaclass*) would.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1788); *backlink*

Unknown interpreted text role "meth".

See also :ref:`class-customization`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1793); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1795)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.6
   Subclasses of :class:`type` which don't override ``type.__new__`` may no
   longer use the one-argument form to get the type of an object.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1799)

Unknown directive type "function".

```
.. function:: vars([object])

   Return the :attr:`~object.__dict__` attribute for a module, class, instance,
   or any other object with a :attr:`~object.__dict__` attribute.

   Objects such as modules and instances have an updateable :attr:`~object.__dict__`
   attribute; however, other objects may have write restrictions on their
   :attr:`~object.__dict__` attributes (for example, classes use a
   :class:`types.MappingProxyType` to prevent direct dictionary updates).

   Without an argument, :func:`vars` acts like :func:`locals`.  Note, the
   locals dictionary is only useful for reads since updates to the locals
   dictionary are ignored.

   A :exc:`TypeError` exception is raised if an object is specified but
   it doesn't have a :attr:`~object.__dict__` attribute (for example, if
   its class defines the :attr:`~object.__slots__` attribute).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1817)

Unknown directive type "function".

```
.. function:: zip(*iterables, strict=False)

   Iterate over several iterables in parallel, producing tuples with an item
   from each one.

   Example::

      >>> for item in zip([1, 2, 3], ['sugar', 'spice', 'everything nice']):
      ...     print(item)
      ...
      (1, 'sugar')
      (2, 'spice')
      (3, 'everything nice')

   More formally: :func:`zip` returns an iterator of tuples, where the *i*-th
   tuple contains the *i*-th element from each of the argument iterables.

   Another way to think of :func:`zip` is that it turns rows into columns, and
   columns into rows.  This is similar to `transposing a matrix
   <https://en.wikipedia.org/wiki/Transpose>`_.

   :func:`zip` is lazy: The elements won't be processed until the iterable is
   iterated on, e.g. by a :keyword:`!for` loop or by wrapping in a
   :class:`list`.

   One thing to consider is that the iterables passed to :func:`zip` could have
   different lengths; sometimes by design, and sometimes because of a bug in
   the code that prepared these iterables.  Python offers three different
   approaches to dealing with this issue:

   * By default, :func:`zip` stops when the shortest iterable is exhausted.
     It will ignore the remaining items in the longer iterables, cutting off
     the result to the length of the shortest iterable::

         >>> list(zip(range(3), ['fee', 'fi', 'fo', 'fum']))
         [(0, 'fee'), (1, 'fi'), (2, 'fo')]

   * :func:`zip` is often used in cases where the iterables are assumed to be
     of equal length.  In such cases, it's recommended to use the ``strict=True``
     option. Its output is the same as regular :func:`zip`::

         >>> list(zip(('a', 'b', 'c'), (1, 2, 3), strict=True))
```

```
      [('a', 1), ('b', 2), ('c', 3)]

   Unlike the default behavior, it checks that the lengths of iterables are
   identical, raising a :exc:`ValueError` if they aren't:

      >>> list(zip(range(3), ['fee', 'fi', 'fo', 'fum'], strict=True))
      Traceback (most recent call last):
        ...
      ValueError: zip() argument 2 is longer than argument 1

   Without the ``strict=True`` argument, any bug that results in iterables of
   different lengths will be silenced, possibly manifesting as a hard-to-find
   bug in another part of the program.

* Shorter iterables can be padded with a constant value to make all the
  iterables have the same length.  This is done by
  :func:`itertools.zip_longest`.

Edge cases: With a single iterable argument, :func:`zip` returns an
iterator of 1-tuples.  With no arguments, it returns an empty iterator.

Tips and tricks:

* The left-to-right evaluation order of the iterables is guaranteed. This
  makes possible an idiom for clustering a data series into n-length groups
  using ``zip(*[iter(s)]*n, strict=True)``.  This repeats the *same* iterator
  ``n`` times so that each output tuple has the result of ``n`` calls to the
  iterator. This has the effect of dividing the input into n-length chunks.

* :func:`zip` in conjunction with the ``*`` operator can be used to unzip a
  list::

      >>> x = [1, 2, 3]
      >>> y = [4, 5, 6]
      >>> list(zip(x, y))
      [(1, 4), (2, 5), (3, 6)]
      >>> x2, y2 = zip(*zip(x, y))
      >>> x == list(x2) and y == list(y2)
      True

.. versionchanged:: 3.10
   Added the ``strict`` argument.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]functions.rst, line 1903)

Unknown directive type "function".

```
.. function:: __import__(name, globals=None, locals=None, fromlist=(), level=0)

   .. index::
      statement: import
      module: imp

   .. note::

      This is an advanced function that is not needed in everyday Python
      programming, unlike :func:`importlib.import_module`.

   This function is invoked by the :keyword:`import` statement.  It can be
   replaced (by importing the :mod:`builtins` module and assigning to
   ``builtins.__import__``) in order to change semantics of the
   :keyword:`!import` statement, but doing so is **strongly** discouraged as it
   is usually simpler to use import hooks (see :pep:`302`) to attain the same
   goals and does not cause issues with code which assumes the default import
   implementation is in use.  Direct use of :func:`__import__` is also
   discouraged in favor of :func:`importlib.import_module`.

   The function imports the module *name*, potentially using the given *globals*
   and *locals* to determine how to interpret the name in a package context.
   The *fromlist* gives the names of objects or submodules that should be
   imported from the module given by *name*.  The standard implementation does
   not use its *locals* argument at all and uses its *globals* only to
   determine the package context of the :keyword:`import` statement.

   *level* specifies whether to use absolute or relative imports. ``0`` (the
   default) means only perform absolute imports.  Positive values for
   *level* indicate the number of parent directories to search relative to the
   directory of the module calling :func:`__import__` (see :pep:`328` for the
   details).

   When the *name* variable is of the form ``package.module``, normally, the
   top-level package (the name up till the first dot) is returned, *not* the
   module named by *name*.  However, when a non-empty *fromlist* argument is
   given, the module named by *name* is returned.

   For example, the statement ``import spam`` results in bytecode resembling the
   following code::

      spam = __import__('spam', globals(), locals(), [], 0)

   The statement ``import spam.ham`` results in this call::

      spam = __import__('spam.ham', globals(), locals(), [], 0)

   Note how :func:`__import__` returns the toplevel module here because this is
   the object that is bound to a name by the :keyword:`import` statement.

   On the other hand, the statement ``from spam.ham import eggs, sausage as
   saus`` results in ::

      _temp = __import__('spam.ham', globals(), locals(), ['eggs', 'sausage'], 0)
      eggs = _temp.eggs
      saus = _temp.sausage

   Here, the ``spam.ham`` module is returned from :func:`__import__`.  From this
```

```
object, the names to import are retrieved and assigned to their respective
names.

If you simply want to import a module (potentially within a package) by name,
use :func:`importlib.import_module`.

.. versionchanged:: 3.3
   Negative values for *level* are no longer supported (which also changes
   the default value to 0).

.. versionchanged:: 3.9
   When the command line options :option:`-E` or :option:`-I` are being used,
   the environment variable :envvar:`PYTHONCASEOK` is now ignored.
```

Footnotes

[1] Note that the parser only accepts the Unix-style end of line convention. If you are reading the code from a file, make sure to use newline conversion mode to convert Windows or Mac-style newlines.