Data model

Objects, values and types

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

Unknown directive type "index".

```
.. index::
   single: object
   single: data
```

:dfn:`Objects` are Python's abstraction for data. All data in a Python program is represented by objects or by relations between objects. (In a sense, and in conformance to Von Neumann's model of a "stored program computer", code is also represented by objects.)

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

Unknown interpreted text role "dfn".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 23)

Unknown directive type "index".

```
.. index::
   builtin: id
   builtin: type
   single: identity of an object
   single: value of an object
   single: type of an object
   single: mutable object
   single: immutable object
```

Every object has an identity, a type and a value. An object's *identity* never changes once it has been created; you may think of it as the object's address in memory. The ':keyword:`is`' operator compares the identity of two objects; the :func:`id` function returns an integer representing its identity.

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

Unknown interpreted text role "keyword".

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

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 40)

Unknown directive type "impl-detail".

```
.. impl-detail::

   For CPython, ``id(x)`` is the memory address where ``x`` is stored.
```

An object's type determines the operations that the object supports (e.g., "does it have a length?") and also defines the possible values for objects of that type. The :func:`type` function returns an object's type (which is an object itself). Like its identity, an object's :dfn:`type` is also unchangeable. [1]

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 44); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 44); *backlink*

Unknown interpreted text role "dfn".

The *value* of some objects can change. Objects whose value can change are said to be *mutable*; objects whose value is unchangeable once they are created are called *immutable*. (The value of an immutable container object that contains a reference to a mutable object can change when the latter's value is changed; however the container is still considered immutable, because the collection of objects it contains cannot be changed. So, immutability is not strictly the same as having an unchangeable value, it is more subtle.) An object's mutability is determined by its type; for instance, numbers, strings and tuples are immutable, while dictionaries and lists are mutable.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 60)

Unknown directive type "index".

```
.. index::
   single: garbage collection
   single: reference counting
   single: unreachable object
```

Objects are never explicitly destroyed; however, when they become unreachable they may be garbage-collected. An implementation is allowed to postpone garbage collection or omit it altogether --- it is a matter of implementation quality how garbage collection is implemented, as long as no objects are collected that are still reachable.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 71)

Unknown directive type "impl-detail".

```
.. impl-detail::

   CPython currently uses a reference-counting scheme with (optional) delayed
   detection of cyclically linked garbage, which collects most objects as soon
   as they become unreachable, but is not guaranteed to collect garbage
   containing circular references.  See the documentation of the :mod:`gc`
   module for information on controlling the collection of cyclic garbage.
   Other implementations act differently and CPython may change.
   Do not depend on immediate finalization of objects when they become
   unreachable (so you should always close files explicitly).
```

Note that the use of the implementation's tracing or debugging facilities may keep objects alive that would normally be collectable. Also note that catching an exception with a ':keyword:`try`...:keyword:`except`' statement may keep objects alive.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 82); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 82); *backlink*

Unknown interpreted text role "keyword".

Some objects contain references to "external" resources such as open files or windows. It is understood that these resources are freed when the object is garbage-collected, but since garbage collection is not guaranteed to happen, such objects also provide an explicit way to release the external resource, usually a :meth:`close` method. Programs are strongly recommended to explicitly close such objects. The ':keyword:`try`...:keyword:`finally`' statement and the ':keyword:`with`' statement provide convenient ways to do this.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 87); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 87); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 87); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 87); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 95)

Unknown directive type "index".

```
.. index:: single: container
```

Some objects contain references to other objects; these are called *containers*. Examples of containers are tuples, lists and dictionaries. The references are part of a container's value. In most cases, when we talk about the value of a container, we imply the values, not the identities of the contained objects; however, when we talk about the mutability of a container, only the identities of the immediately contained objects are implied. So, if an immutable container (like a tuple) contains a reference to a mutable object, its value changes if that mutable object is changed.

Types affect almost all aspects of object behavior. Even the importance of object identity is affected in some sense: for immutable types, operations that compute new values may actually return a reference to any existing object with the same type and value, while for mutable objects this is not allowed. E.g., after `a = 1; b = 1`, `a` and `b` may or may not refer to the same object with the value one, depending on the implementation, but after `c = []; d = []`, `c` and `d` are guaranteed to refer to two different, unique, newly created empty lists. (Note that `c = d = []` assigns the same object to both `c` and `d`.)

The standard type hierarchy

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 122)

Unknown directive type "index".

```
.. index::
   single: type
   pair: data; type
   pair: type; hierarchy
   pair: extension; module
   pair: C; language
```

Below is a list of the types that are built into Python. Extension modules (written in C, Java, or other languages, depending on the implementation) can define additional types. Future versions of Python may add types to the type hierarchy (e.g., rational numbers, efficiently stored arrays of integers, etc.), although such additions will often be provided via the standard library instead.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 135)

Unknown directive type "index".

```
.. index::
   single: attribute
   pair: special; attribute
   triple: generic; special; attribute
```

Some of the type descriptions below contain a paragraph listing 'special attributes.' These are attributes that provide access to the implementation and are not intended for general use. Their definition may change in the future.

None

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 145)**
>
> Unknown directive type "index".
>
> ```
> .. index:: object: None
> ```
>
> This type has a single value. There is a single object with this value. This object is accessed through the built-in name `None`. It is used to signify the absence of a value in many situations, e.g., it is returned from functions that don't explicitly return anything. Its truth value is false.

NotImplemented

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

line 153)

Unknown directive type "index".

```
.. index:: object: NotImplemented
```

This type has a single value. There is a single object with this value. This object is accessed through the built-in name `NotImplemented`. Numeric methods and rich comparison methods should return this value if they do not implement the operation for the operands provided. (The interpreter will then try the reflected operation, or some other fallback, depending on the operator.) It should not be evaluated in a boolean context.

See :ref:`implementing-the-arithmetic-operations` for more details.

System Message: ERROR/3 (**`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`**, **line 162**); ***backlink***

Unknown interpreted text role "ref".

System Message: ERROR/3 (**`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`**, **line 166**)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.9
   Evaluating ``NotImplemented`` in a boolean context is deprecated. While
   it currently evaluates as true, it will emit a :exc:`DeprecationWarning`.
   It will raise a :exc:`TypeError` in a future version of Python.
```

Ellipsis

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

Unknown directive type "index".

```
.. index::
   object: Ellipsis
   single: ...; ellipsis literal
```

This type has a single value. There is a single object with this value. This object is accessed through the literal `...` or the built-in name `Ellipsis`. Its truth value is true.

:class:`numbers.Number`

System Message: ERROR/3 (**`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`**, **line 266**); ***backlink***

Unknown interpreted text role "class".

System Message: ERROR/3 (**`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`**, **line 182**)

Unknown directive type "index".

```
.. index:: object: numeric
```

These are created by numeric literals and returned as results by arithmetic operators and arithmetic built-in functions. Numeric objects are immutable; once created their value never changes. Python numbers are of course strongly related to mathematical numbers, but subject to the limitations of numerical representation in computers.

The string representations of the numeric classes, computed by :meth:`~object.__repr__` and :meth:`~object.__str__`, have the following properties:

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

Unknown interpreted text role "meth".

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

Unknown interpreted text role "meth".

- They are valid numeric literals which, when passed to their class constructor, produce an object having the value of the original numeric.
- The representation is in base 10, when possible.
- Leading zeros, possibly excepting a single zero before a decimal point, are not shown.
- Trailing zeros, possibly excepting a single zero after a decimal point, are not shown.
- A sign is shown only when the number is negative.

Python distinguishes between integers, floating point numbers, and complex numbers:

:class:`numbers.Integral`

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 241); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 212)

Unknown directive type "index".

```
.. index:: object: integer
```

These represent elements from the mathematical set of integers (positive and negative).

There are two types of integers:

Integers (:class:`int`)

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 224); *backlink*

Unknown interpreted text role "class".

These represent numbers in an unlimited range, subject to available (virtual) memory only. For the purpose of shift and mask operations, a binary representation is assumed, and negative numbers are represented in a variant of 2's complement which gives the illusion of an infinite string of sign bits extending to the left.

Booleans (:class:`bool`)

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 236); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 227)

Unknown directive type "index".

```
.. index::
   object: Boolean
   single: False
   single: True
```

These represent the truth values False and True. The two objects representing the values `False` and `True` are the only Boolean objects. The Boolean type is a subtype of the integer type, and Boolean values behave like the values 0 and 1, respectively, in almost all contexts, the exception being that when converted to a string, the strings `"False"` or `"True"` are returned, respectively.

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

[reference]datamodel.rst, line 238)

Unknown directive type "index".

```
.. index:: pair: integer; representation
```

The rules for integer representation are intended to give the most meaningful interpretation of shift and mask operations involving negative integers.

:class:`numbers.Real` (:class:`float`)

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 244)

Unknown directive type "index".

```
.. index::
   object: floating point
   pair: floating point; number
   pair: C; language
   pair: Java; language
```

These represent machine-level double precision floating point numbers. You are at the mercy of the underlying machine architecture (and C or Java implementation) for the accepted range and handling of overflow. Python does not support single-precision floating point numbers; the savings in processor and memory usage that are usually the reason for using these are dwarfed by the overhead of using objects in Python, so there is no reason to complicate the language with two kinds of floating point numbers.

:class:`numbers.Complex` (:class:`complex`)

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 266); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 266); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 259)

Unknown directive type "index".

```
.. index::
   object: complex
   pair: complex; number
```

These represent complex numbers as a pair of machine-level double precision floating point numbers. The same caveats apply as for floating point numbers. The real and imaginary parts of a complex number `z` can be retrieved through the read-only attributes `z.real` and `z.imag`.

Sequences

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 269)

Unknown directive type "index".

```
.. index::
   builtin: len
   object: sequence
   single: index operation
   single: item selection
   single: subscription
```

These represent finite ordered sets indexed by non-negative numbers. The built-in function :func:`len` returns the number of items of a sequence. When the length of a sequence is *n*, the index set contains the numbers 0, 1, ..., *n*-1. Item *i* of sequence *a* is selected by `a[i]`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 276); *backlink*

Unknown interpreted text role "func".

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

Unknown directive type "index".

```
.. index:: single: slicing
```

Sequences also support slicing: `a[i:j]` selects all items with index *k* such that *i* <= *k* < *j*. When used as an expression, a slice is a sequence of the same type. This implies that the index set is renumbered so that it starts at 0.

Some sequences also support "extended slicing" with a third "step" parameter: `a[i:j:k]` selects all items of *a* with index *x* where `x = i + n*k`, *n* `>=` `0` and *i* <= *x* < *j*.

Sequences are distinguished according to their mutability:

Immutable sequences

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 295)

Unknown directive type "index".

```
.. index::
   object: immutable sequence
   object: immutable
```

An object of an immutable sequence type cannot change once it is created. (If the object contains references to other objects, these other objects may be mutable and may be changed; however, the collection of objects directly referenced by an immutable object cannot change.)

The following types are immutable sequences:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 306)

Unknown directive type "index".

```
.. index::
   single: string; immutable sequences
```

Strings

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 310)

Unknown directive type "index".

```
.. index::
   builtin: chr
   builtin: ord
   single: character
   single: integer
   single: Unicode
```

A string is a sequence of values that represent Unicode code points. All the code points in the range `U+0000 - U+10FFFF` can be represented in a string. Python doesn't have a :c:type:`char` type; instead, every code point in the string is represented as a string object with length `1`. The built-in function :func:`ord` converts a code point from its string form to an integer in the range `0 - 10FFFF`; :func:`chr` converts an integer in the range `0 - 10FFFF` to the corresponding length `1` string object. :meth:`str.encode` can be used to convert a :class:`str` to :class:`bytes` using the given text encoding, and :meth:`bytes.decode` can be used to achieve the opposite.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 317); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 317); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 317); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 317); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 317); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 317); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 317); *backlink*

Unknown interpreted text role "meth".

Tuples

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 330)

Unknown directive type "index".

```
.. index::
   object: tuple
   pair: singleton; tuple
   pair: empty; tuple
```

The items of a tuple are arbitrary Python objects. Tuples of two or more items are formed by comma-separated lists of expressions. A tuple of one item (a 'singleton') can be formed by affixing a comma to an expression (an expression by itself does not create a tuple, since parentheses must be usable for grouping of expressions). An empty tuple can be formed by an empty pair of parentheses.

Bytes

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 343)

Unknown directive type "index".

```
.. index:: bytes, byte
```

A bytes object is an immutable array. The items are 8-bit bytes, represented by integers in the range 0 <= x < 256. Bytes literals (like `b'abc'`) and the built-in :func:`bytes()` constructor can be used to create bytes objects. Also, bytes objects can be decoded to strings via the :meth:`~bytes.decode` method.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 345); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 345); *backlink*

Unknown interpreted text role "meth".

Mutable sequences

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 352)

Unknown directive type "index".

```
.. index::
   object: mutable sequence
   object: mutable
   pair: assignment; statement
   single: subscription
   single: slicing
```

Mutable sequences can be changed after they are created. The subscription and slicing notations can be used as the target of assignment and :keyword:`del` (delete) statements.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 359); *backlink*

Unknown interpreted text role "keyword".

There are currently two intrinsic mutable sequence types:

Lists

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 366)

Unknown directive type "index".

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

The items of a list are arbitrary Python objects. Lists are formed by placing a comma-separated list of expressions in square brackets. (Note that there are no special cases needed to form lists of length 0 or 1.)

Byte Arrays

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 373)

Unknown directive type "index".

```
.. index:: bytearray
```

A bytearray object is a mutable array. They are created by the built-in :func:`bytearray` constructor. Aside from being mutable (and hence unhashable), byte arrays otherwise provide the same interface and functionality as immutable :class:`bytes` objects.

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

Unknown interpreted text role "func".

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

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 380)

Unknown directive type "index".

```
.. index:: module: array
```

The extension module :mod:`array` provides an additional example of a mutable sequence type, as does the :mod:`collections` module.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 382); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 382); *backlink*

Unknown interpreted text role "mod".

Set types

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

Unknown directive type "index".

```
.. index::
   builtin: len
   object: set type
```

These represent unordered, finite sets of unique, immutable objects. As such, they cannot be indexed by any subscript. However, they can be iterated over, and the built-in function :func:`len` returns the number of items in a set. Common uses for sets are fast membership testing, removing duplicates from a sequence, and computing mathematical operations such as intersection, union, difference, and symmetric difference.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 390); *backlink*

Unknown interpreted text role "func".

For set elements, the same immutability rules apply as for dictionary keys. Note that numeric types obey the normal rules for numeric comparison: if two numbers compare equal (e.g., `1` and `1.0`), only one of them can be contained in a set.

There are currently two intrinsic set types:

Sets

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 405)

Unknown directive type "index".

```
.. index:: object: set
```

These represent a mutable set. They are created by the built-in :func:`set` constructor and can be modified afterwards by several methods, such as :meth:`~set.add`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 407); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 407); *backlink*

Unknown interpreted text role "meth".

Frozen sets

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 412)

Unknown directive type "index".

```
.. index:: object: frozenset
```

These represent an immutable set. They are created by the built-in :func:`frozenset` constructor. As a frozenset is immutable and :term:`hashable`, it can be used again as an element of another set, or as a dictionary key.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 414); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 414); *backlink*

Unknown interpreted text role "term".

Mappings

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 420)

Unknown directive type "index".

```
.. index::
   builtin: len
   single: subscription
   object: mapping
```

These represent finite sets of objects indexed by arbitrary index sets. The subscript notation `a[k]` selects the item indexed by `k` from the mapping `a`; this can be used in expressions and as the target of assignments or :keyword:`del` statements. The built-in function :func:`len` returns the number of items in a mapping.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 425); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 425); *backlink*

Unknown interpreted text role "func".

There is currently a single intrinsic mapping type:

Dictionaries

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 434)**
>
> Unknown directive type "index".
>
> ```
> .. index:: object: dictionary
> ```

These represent finite sets of objects indexed by nearly arbitrary values. The only types of values not acceptable as keys are values containing lists or dictionaries or other mutable types that are compared by value rather than by object identity, the reason being that the efficient implementation of dictionaries requires a key's hash value to remain constant. Numeric types used for keys obey the normal rules for numeric comparison: if two numbers compare equal (e.g., `1` and `1.0`) then they can be used interchangeably to index the same dictionary entry.

Dictionaries preserve insertion order, meaning that keys will be produced in the same order they were added sequentially over the dictionary. Replacing an existing key does not change the order, however removing a key and re-inserting it will add it to the end instead of keeping its old place.

Dictionaries are mutable; they can be created by the `{...}` notation (see section :ref:`dict`).

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 450); *backlink***
>
> Unknown interpreted text role "ref".

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 453)**
>
> Unknown directive type "index".
>
> ```
> .. index::
>    module: dbm.ndbm
>    module: dbm.gnu
> ```

The extension modules :mod:`dbm.ndbm` and :mod:`dbm.gnu` provide additional examples of mapping types, as does the :mod:`collections` module.

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 457); *backlink***
>
> Unknown interpreted text role "mod".

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 457); *backlink***
>
> Unknown interpreted text role "mod".

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 457); *backlink***
>
> Unknown interpreted text role "mod".

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 461)**
>
> Unknown directive type "versionchanged".
>
> ```
> .. versionchanged:: 3.7
>    Dictionaries did not preserve insertion order in versions of Python before 3.6.
>    In CPython 3.6, insertion order was preserved, but it was considered
>    an implementation detail at that time rather than a language guarantee.
> ```

Callable types

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 467)

Unknown directive type "index".

```
.. index::
   object: callable
   pair: function; call
   single: invocation
   pair: function; argument
```

These are the types to which the function call operation (see section :ref:`calls`) can be applied:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 473); *backlink*

Unknown interpreted text role "ref".

User-defined functions

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 477)

Unknown directive type "index".

```
.. index::
   pair: user-defined; function
   object: function
   object: user-defined function
```

A user-defined function object is created by a function definition (see section :ref:`function`). It should be called with an argument list containing the same number of items as the function's formal parameter list.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 482); *backlink*

Unknown interpreted text role "ref".

Special attributes:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 489)

Unknown directive type "tabularcolumns".

```
.. tabularcolumns:: |l|L|l|
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 491)

Unknown directive type "index".

```
.. index::
   single: __doc__ (function attribute)
   single: __name__ (function attribute)
   single: __module__ (function attribute)
   single: __dict__ (function attribute)
   single: __defaults__ (function attribute)
   single: __closure__ (function attribute)
   single: __code__ (function attribute)
   single: __globals__ (function attribute)
   single: __annotations__ (function attribute)
   single: __kwdefaults__ (function attribute)
   pair: global; namespace
```

Attribute	**Meaning**	

Attribute	Meaning	
:attr:`__doc__` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 508); *backlink* Unknown interpreted text role "attr".	The function's documentation string, or `None` if unavailable; not inherited by subclasses.	Writable
:attr:`~definition.\ __name__` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 513); *backlink* Unknown interpreted text role "attr".	The function's name.	Writable
:attr:`~definition.\ __qualname__` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 516); *backlink* Unknown interpreted text role "attr".	The function's :term:`qualified name`. **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 516); *backlink* Unknown interpreted text role "term". **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 519) Unknown directive type "versionadded". `.. versionadded:: 3.3`	Writable

Attribute	Meaning	
:attr:`__module__` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 521); *backlink* Unknown interpreted text role "attr".	The name of the module the function was defined in, or `None` if unavailable.	Writable
:attr:`__defaults__` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 525); *backlink* Unknown interpreted text role "attr".	A tuple containing default argument values for those arguments that have defaults, or `None` if no arguments have a default value.	Writable
:attr:`__code__` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 531); *backlink* Unknown interpreted text role "attr".	The code object representing the compiled function body.	Writable
:attr:`__globals__` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 534); *backlink* Unknown interpreted text role "attr".	A reference to the dictionary that holds the function's global variables --- the global namespace of the module in which the function was defined.	Read-only

Attribute	Meaning	
:attr:`~object.__dict__` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 541); *backlink* Unknown interpreted text role "attr".	The namespace supporting arbitrary function attributes.	Writable
:attr:`__closure__` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 545); *backlink* Unknown interpreted text role "attr".	`None` or a tuple of cells that contain bindings for the function's free variables. See below for information on the `cell_contents` attribute.	Read-only
:attr:`__annotations__` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 552); *backlink* Unknown interpreted text role "attr".	A dict containing annotations of parameters. The keys of the dict are the parameter names, and `'return'` for the return annotation, if provided. For more information on working with this attribute, see :ref:`annotations-howto`. **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 552); *backlink* Unknown interpreted text role "ref".	Writable
:attr:`__kwdefaults__` **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 562); *backlink* Unknown interpreted text role "attr".	A dict containing defaults for keyword-only parameters.	Writable

Most of the attributes labelled "Writable" check the type of the assigned value.

Function objects also support getting and setting arbitrary attributes, which can be used, for example, to attach metadata to functions. Regular attribute dot-notation is used to get and set such attributes. *Note that the current implementation only supports function attributes on user-defined functions. Function attributes on built-in functions may be supported in the future.*

A cell object has the attribute `cell_contents`. This can be used to get the value of the cell, as well as set the value.

Additional information about a function's definition can be retrieved from its code object; see the description of internal types below. The :data:`cell <types.CellType>` type can be accessed in the :mod:`types` module.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 576); *backlink*

Unknown interpreted text role "data".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 576); *backlink*

Unknown interpreted text role "mod".

Instance methods

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 582)

Unknown directive type "index".

```
.. index::
   object: method
   object: user-defined method
   pair: user-defined; method
```

An instance method object combines a class, a class instance and any callable object (normally a user-defined function).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 590)

Unknown directive type "index".

```
.. index::
   single: __func__ (method attribute)
   single: __self__ (method attribute)
   single: __doc__ (method attribute)
   single: __name__ (method attribute)
   single: __module__ (method attribute)
```

Special read-only attributes: :attr:`__self__` is the class instance object, :attr:`__func__` is the function object; :attr:`__doc__` is the method's documentation (same as `__func__.__doc__`); :attr:`~definition.__name__` is the method name (same as `__func__.__name__`); :attr:`__module__` is the name of the module the method was defined in, or `None` if unavailable.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 597); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 597); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 597); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 597); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 597); *backlink*

Unknown interpreted text role "attr".

Methods also support accessing (but not setting) the arbitrary function attributes on the underlying function object.

User-defined method objects may be created when getting an attribute of a class (perhaps via an instance of that class), if that attribute is a user-defined function object or a class method object.

When an instance method object is created by retrieving a user-defined function object from a class via one of its instances, its :attr:`__self__` attribute is the instance, and the method object is said to be bound. The new method's :attr:`__func__` attribute is the original function object.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 610); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 610); *backlink*

Unknown interpreted text role "attr".

When an instance method object is created by retrieving a class method object from a class or instance, its :attr:`__self__` attribute is the class itself, and its :attr:`__func__` attribute is the function object underlying the class method.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 616); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 616); *backlink*

Unknown interpreted text role "attr".

When an instance method object is called, the underlying function (:attr:`__func__`) is called, inserting the class instance (:attr:`__self__`) in front of the argument list. For instance, when :class:`C` is a class which contains a definition for a function :meth:`f`, and `x` is an instance of :class:`C`, calling `x.f(1)` is equivalent to calling `C.f(x, 1)`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 621); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 621); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 621); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 621); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 621); *backlink*

Unknown interpreted text role "class".

When an instance method object is derived from a class method object, the "class instance" stored in :attr:`__self__` will actually be the class itself, so that calling either `x.f(1)` or `C.f(1)` is equivalent to calling `f(C,1)` where `f` is the underlying function.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 628); *backlink*

Unknown interpreted text role "attr".

Note that the transformation from function object to instance method object happens each time the attribute is retrieved from the instance. In some cases, a fruitful optimization is to assign the attribute to a local variable and call that local variable. Also notice that this transformation only happens for user-defined functions; other callable objects (and all non-callable objects) are retrieved without transformation. It is also important to note that user-defined functions which are attributes of a class instance are not converted to bound methods; this *only* happens when the function is an attribute of the class.

Generator functions

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 645)

Unknown directive type "index".

```
.. index::
   single: generator; function
   single: generator; iterator
```

A function or method which uses the :keyword:`yield` statement (see section :ref:`yield`) is called a :dfn:`generator function`. Such a function, when called, always returns an :term:`iterator` object which can be used to execute the body of the function: calling the iterator's :meth:`iterator.__next__` method will cause the function to execute until it provides a value using the :keyword:`!yield` statement. When the function executes a :keyword:`return` statement or falls off the end, a :exc:`StopIteration` exception is raised and the iterator will have reached the end of the set of values to be returned.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 649); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 649); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 649); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 649); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 649); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 649); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 649); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 649); *backlink*

Unknown interpreted text role "exc".

Coroutine functions

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 660)

Unknown directive type "index".

```
.. index::
   single: coroutine; function
```

A function or method which is defined using :keyword:`async def` is called a :dfn:`coroutine function`. Such a function, when called, returns a :term:`coroutine` object. It may contain :keyword:`await` expressions, as well as :keyword:`async with` and :keyword:`async for` statements. See also the :ref:`coroutine-objects` section.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 663); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 663); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 663); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 663); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 663); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 663); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 663);** ***backlink***

Unknown interpreted text role "ref".

Asynchronous generator functions

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 670)**

Unknown directive type "index".

```
.. index::
   single: asynchronous generator; function
   single: asynchronous generator; asynchronous iterator
```

A function or method which is defined using :keyword:`async def` and which uses the :keyword:`yield` statement is called a :dfn:`asynchronous generator function`. Such a function, when called, returns an :term:`asynchronous iterator` object which can be used in an :keyword:`async for` statement to execute the body of the function.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 674);** ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 674);** ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 674);** ***backlink***

Unknown interpreted text role "dfn".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 674);** ***backlink***

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 674);** ***backlink***

Unknown interpreted text role "keyword".

Calling the asynchronous iterator's :meth:`aiterator.__anext__ <object.__anext__>` method will return an :term:`awaitable` which when awaited will execute until it provides a value using the :keyword:`yield` expression. When the function executes an empty :keyword:`return` statement or falls off the end, a :exc:`StopAsyncIteration` exception is raised and the asynchronous iterator will have reached the end of the set of values to be yielded.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 680);** ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 680);** ***backlink***

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc]` [reference]datamodel.rst, line 680); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 680); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 680); *backlink*

Unknown interpreted text role "exc".

Built-in functions

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 690)

Unknown directive type "index".

```
.. index::
   object: built-in function
   object: function
   pair: C; language
```

A built-in function object is a wrapper around a C function. Examples of built-in functions are :func:`len` and :func:`math.sin` (:mod:`math` is a standard built-in module). The number and type of the arguments are determined by the C function. Special read-only attributes: :attr:`__doc__` is the function's documentation string, or `None` if unavailable; :attr:`~definition.__name__` is the function's name; :attr:`__self__` is set to `None` (but see the next item); :attr:`__module__` is the name of the module the function was defined in or `None` if unavailable.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 695); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 695); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 695); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 695); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 695); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 695); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 695**); *backlink*

Unknown interpreted text role "attr".

Built-in methods

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

Unknown directive type "index".

```
.. index::
   object: built-in method
   object: method
   pair: built-in; method
```

This is really a different disguise of a built-in function, this time containing an object passed to the C function as an implicit extra argument. An example of a built-in method is `alist.append()`, assuming *alist* is a list object. In this case, the special read-only attribute :attr:`__self__` is set to the object denoted by *alist*.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 710**); *backlink*

Unknown interpreted text role "attr".

Classes

Classes are callable. These objects normally act as factories for new instances of themselves, but variations are possible for class types that override :meth:`~object.__new__`. The arguments of the call are passed to :meth:`__new__` and, in the typical case, to :meth:`~object.__init__` to initialize the new instance.

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

Unknown interpreted text role "meth".

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

Unknown interpreted text role "meth".

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

Unknown interpreted text role "meth".

Class Instances

Instances of arbitrary classes can be made callable by defining a :meth:`~object.__call__` method in their class.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 724**); *backlink*

Unknown interpreted text role "meth".

Modules

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 729**)

Unknown directive type "index".

```
.. index::
   statement: import
   object: module
```

Modules are a basic organizational unit of Python code, and are created by the :ref:`import system <importsystem>` as invoked either by the :keyword:`import` statement, or by calling functions such as :func:`importlib.import_module` and built-in :func:`__import__`. A module object has a namespace implemented by a dictionary object (this is the dictionary referenced by the `__globals__` attribute of functions defined in the module). Attribute references are translated to lookups in this dictionary, e.g., `m.x` is equivalent to `m.__dict__["x"]`. A module object does not contain the code object used to initialize the module (since it isn't needed once the initialization is done).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 733); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 733); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 733); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 733); *backlink*

Unknown interpreted text role "func".

Attribute assignment updates the module's namespace dictionary, e.g., `m.x = 1` is equivalent to `m.__dict__["x"] = 1`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 748)

Unknown directive type "index".

```
.. index::
   single: __name__ (module attribute)
   single: __doc__ (module attribute)
   single: __file__ (module attribute)
   single: __annotations__ (module attribute)
   pair: module; namespace
```

Predefined (writable) attributes:

:attr:`__name__`

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 758); *backlink*

Unknown interpreted text role "attr".

The module's name.

:attr:`__doc__`

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 762); *backlink*

Unknown interpreted text role "attr".

The module's documentation string, or `None` if unavailable.

:attr:`__file__`

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

[Doc][reference]datamodel.rst, line 771); *backlink*

Unknown interpreted text role "attr".

The pathname of the file from which the module was loaded, if it was loaded from a file. The :attr:`__file__` attribute may be missing for certain types of modules, such as C modules that are statically linked into the interpreter. For extension modules loaded dynamically from a shared library, it's the pathname of the shared library file.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 765); *backlink*

Unknown interpreted text role "attr".

:attr:`__annotations__`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 777); *backlink*

Unknown interpreted text role "attr".

A dictionary containing :term:`variable annotations <variable annotation>` collected during module body execution. For best practices on working with :attr:`__annotations__`, please see :ref:`annotations-howto`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 774); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 774); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 774); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 779)

Unknown directive type "index".

```
.. index:: single: __dict__ (module attribute)
```

Special read-only attribute: :attr:`~object.__dict__` is the module's namespace as a dictionary object.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 781); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 784)

Unknown directive type "impl-detail".

```
.. impl-detail::

   Because of the way CPython clears module dictionaries, the module
   dictionary will be cleared when the module falls out of scope even if the
   dictionary still has live references.  To avoid this, copy the dictionary
   or keep the module around while using its dictionary directly.
```

Custom classes

Custom class types are typically created by class definitions (see section :ref:`class`). A class has a namespace implemented by a dictionary object. Class attribute references are translated to lookups in this dictionary, e.g., `C.x` is translated to `C.__dict__["x"]` (although there are a number of hooks which allow for other means of locating attributes). When the attribute name is not found there, the attribute search continues in the base classes. This search of the base classes uses the C3 method resolution order which behaves correctly even in the presence of 'diamond' inheritance structures where there are multiple inheritance paths leading back to a common ancestor. Additional details on the C3 MRO used by Python can be found in the documentation accompanying the 2.3 release at https://www.python.org/download/releases/2.3/mro/.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 792**); ***backlink***

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 807**)

Unknown directive type "index".

```
.. index::
   object: class
   object: class instance
   object: instance
   pair: class object; call
   single: container
   object: dictionary
   pair: class; attribute
```

When a class attribute reference (for class :class:`C`, say) would yield a class method object, it is transformed into an instance method object whose :attr:`__self__` attribute is :class:`C`. When it would yield a static method object, it is transformed into the object wrapped by the static method object. See section :ref:`descriptors` for another way in which attributes retrieved from a class may differ from those actually contained in its :attr:`~object.__dict__`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 816**); ***backlink***

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 816**); ***backlink***

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 816**); ***backlink***

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 816**); ***backlink***

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 816**); ***backlink***

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 824**)

Unknown directive type "index".

```
.. index:: triple: class; attribute; assignment
```

Class attribute assignments update the class's dictionary, never the dictionary of a base class.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 829)

Unknown directive type "index".

```
.. index:: pair: class object; call
```

A class object can be called (see above) to yield a class instance (see below).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 833)

Unknown directive type "index".

```
.. index::
   single: __name__ (class attribute)
   single: __module__ (class attribute)
   single: __dict__ (class attribute)
   single: __bases__ (class attribute)
   single: __doc__ (class attribute)
   single: __annotations__ (class attribute)
```

Special attributes:

:attr:`~definition.__name__`

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 844); *backlink***
>
> Unknown interpreted text role "attr".

The class name.

:attr:`__module__`

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 847); *backlink***
>
> Unknown interpreted text role "attr".

The name of the module in which the class was defined.

:attr:`~object.__dict__`

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 850); *backlink***
>
> Unknown interpreted text role "attr".

The dictionary containing the class's namespace.

:attr:`~class.__bases__`

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 854); *backlink***
>
> Unknown interpreted text role "attr".

A tuple containing the base classes, in the order of their occurrence in the base class list.

:attr:`__doc__`

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

`[Doc][reference]datamodel.rst`, line 857); *backlink*

Unknown interpreted text role "attr".

The class's documentation string, or `None` if undefined.

:attr:`__annotations__`

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 864); *backlink*

Unknown interpreted text role "attr".

A dictionary containing :term:`variable annotations <variable annotation>` collected during class body execution. For best practices on working with :attr:`__annotations__`, please see :ref:`annotations-howto`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 860); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 860); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 860); *backlink*

Unknown interpreted text role "ref".

Class instances

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 867)

Unknown directive type "index".

```
.. index::
   object: class instance
   object: instance
   pair: class; instance
   pair: class instance; attribute
```

A class instance is created by calling a class object (see above). A class instance has a namespace implemented as a dictionary which is the first place in which attribute references are searched. When an attribute is not found there, and the instance's class has an attribute by that name, the search continues with the class attributes. If a class attribute is found that is a user-defined function object, it is transformed into an instance method object whose :attr:`__self__` attribute is the instance. Static method and class method objects are also transformed; see above under "Classes". See section :ref:`descriptors` for another way in which attributes of a class retrieved via its instances may differ from the objects actually stored in the class's :attr:`~object.__dict__`. If no class attribute is found, and the object's class has a :meth:`~object.__getattr__` method, that is called to satisfy the lookup.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 873); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 873); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 873); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 873); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 887)

Unknown directive type "index".

```
.. index:: triple: class instance; attribute; assignment
```

Attribute assignments and deletions update the instance's dictionary, never a class's dictionary. If the class has a :meth:`~object.__setattr__` or :meth:`~object.__delattr__` method, this is called instead of updating the instance dictionary directly.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 889); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 889); *backlink*

Unknown interpreted text role "meth".

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

Unknown directive type "index".

```
.. index::
   object: numeric
   object: sequence
   object: mapping
```

Class instances can pretend to be numbers, sequences, or mappings if they have methods with certain special names. See section :ref:`specialnames`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 899); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 902)

Unknown directive type "index".

```
.. index::
   single: __dict__ (instance attribute)
   single: __class__ (instance attribute)
```

Special attributes: :attr:`~object.__dict__` is the attribute dictionary; :attr:`~instance.__class__` is the instance's class.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 906); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 906); *backlink*

Unknown interpreted text role "attr".

I/O objects (also known as file objects)

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 910)

Unknown directive type "index".

```
.. index::
   builtin: open
   module: io
   single: popen() (in module os)
   single: makefile() (socket method)
   single: sys.stdin
   single: sys.stdout
   single: sys.stderr
   single: stdio
   single: stdin (in module sys)
   single: stdout (in module sys)
   single: stderr (in module sys)
```

A :term:`file object` represents an open file. Various shortcuts are available to create file objects: the :func:`open` built-in function, and also :func:`os.popen`, :func:`os.fdopen`, and the :meth:`~socket.socket.makefile` method of socket objects (and perhaps by other functions or methods provided by extension modules).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 923); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 923); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 923); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 923); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 923); *backlink*

Unknown interpreted text role "meth".

The objects `sys.stdin`, `sys.stdout` and `sys.stderr` are initialized to file objects corresponding to the interpreter's standard input, output and error streams; they are all open in text mode and therefore follow the interface defined by the :class:`io.TextIOBase` abstract class.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 929); *backlink*

Unknown interpreted text role "class".

Internal types

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 936)

Unknown directive type "index".

```
.. index::
   single: internal type
   single: types, internal
```

A few types used internally by the interpreter are exposed to the user. Their definitions may change with future versions of the interpreter, but they are mentioned here for completeness.

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

Unknown directive type "index".

```
.. index:: bytecode, object; code, code object
```

Code objects

Code objects represent *byte-compiled* executable Python code, or :term:`bytecode`. The difference between a code object and a function object is that the function object contains an explicit reference to the function's globals (the module in which it was defined), while a code object contains no context; also the default argument values are stored in the function object, not in the code object (because they represent values calculated at run-time). Unlike function objects, code objects are immutable and contain no references (directly or indirectly) to mutable objects.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 947); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 956)

Unknown directive type "index".

```
.. index::
   single: co_argcount (code object attribute)
   single: co_posonlyargcount (code object attribute)
   single: co_kwonlyargcount (code object attribute)
   single: co_code (code object attribute)
   single: co_consts (code object attribute)
   single: co_filename (code object attribute)
   single: co_firstlineno (code object attribute)
   single: co_flags (code object attribute)
   single: co_lnotab (code object attribute)
   single: co_name (code object attribute)
   single: co_names (code object attribute)
   single: co_nlocals (code object attribute)
   single: co_stacksize (code object attribute)
   single: co_varnames (code object attribute)
   single: co_cellvars (code object attribute)
   single: co_freevars (code object attribute)
   single: co_qualname (code object attribute)
```

Special read-only attributes: :attr:`co_name` gives the function name; :attr:`co_qualname` gives the fully qualified function name; :attr:`co_argcount` is the total number of positional arguments (including positional-only arguments and arguments with default values); :attr:`co_posonlyargcount` is the number of positional-only arguments (including arguments with default values); :attr:`co_kwonlyargcount` is the number of keyword-only arguments (including arguments with default values); :attr:`co_nlocals` is the number of local variables used by the function (including arguments); :attr:`co_varnames` is a tuple containing the names of the local variables (starting with the argument names); :attr:`co_cellvars` is a tuple containing the names of local variables that are referenced by nested functions; :attr:`co_freevars` is a tuple containing the names of free variables; :attr:`co_code` is a string representing the sequence of bytecode instructions; :attr:`co_consts` is a tuple containing the literals used by the bytecode; :attr:`co_names` is a tuple containing the names used by the bytecode; :attr:`co_filename` is the filename from which the code was compiled; :attr:`co_firstlineno` is the first line number of the function; :attr:`co_lnotab` is a string encoding the mapping from bytecode offsets to line numbers (for details see the source code of the interpreter); :attr:`co_stacksize` is the required stack size; :attr:`co_flags` is an integer encoding a number of flags for the interpreter.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 975); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 975); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 975); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 975); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 975); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 975); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 975); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 975); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 975); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 975); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 975); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 975); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 975); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 975); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 975); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 975); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 975); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 998)

Unknown directive type "index".

```
.. index:: object: generator
```

The following flag bits are defined for :attr:`co_flags`: bit `0x04` is set if the function uses the `*arguments` syntax to accept an arbitrary number of positional arguments; bit `0x08` is set if the function uses the `**keywords` syntax to accept arbitrary keyword arguments; bit `0x20` is set if the function is a generator.

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

Unknown interpreted text role "attr".

Future feature declarations (`from __future__ import division`) also use bits in :attr:`co_flags` to indicate whether a code object was compiled with a particular feature enabled: bit `0x2000` is set if the function was compiled with future division enabled; bits `0x10` and `0x1000` were used in earlier versions of Python.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1006); *backlink*

Unknown interpreted text role "attr".

Other bits in :attr:`co_flags` are reserved for internal use.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1012); *backlink*

Unknown interpreted text role "attr".

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

[reference]datamodel.rst, line 1014)

Unknown directive type "index".

```
.. index:: single: documentation string
```

If a code object represents a function, the first item in :attr:`co_consts` is the documentation string of the function, or `None` if undefined.

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

Unknown interpreted text role "attr".

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

Unknown directive type "method".

```
.. method:: codeobject.co_positions()

   Returns an iterable over the source code positions of each bytecode
   instruction in the code object.

   The iterator returns tuples containing the ``(start_line, end_line,
   start_column, end_column)``. The *i-th* tuple corresponds to the
   position of the source code that compiled to the *i-th* instruction.
   Column information is 0-indexed utf-8 byte offsets on the given source
   line.

   This positional information can be missing. A non-exhaustive lists of
   cases where this may happen:

   - Running the interpreter with :option:`-X` ``no_debug_ranges``.
   - Loading a pyc file compiled while using :option:`-X` ``no_debug_ranges``.
   - Position tuples corresponding to artificial instructions.
   - Line and column numbers that can't be represented due to
     implementation specific limitations.

   When this occurs, some or all of the tuple elements can be
   :const:`None`.

   .. versionadded:: 3.11

   .. note::
      This feature requires storing column positions in code objects which may
      result in a small increase of disk usage of compiled Python files or
      interpreter memory usage. To avoid storing the extra information and/or
      deactivate printing the extra traceback information, the
      :option:`-X` ``no_debug_ranges`` command line flag or the :envvar:`PYTHONNODEBUGRA
      environment variable can be used.
```

Frame objects

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

Unknown directive type "index".

```
.. index:: object: frame
```

Frame objects represent execution frames. They may occur in traceback objects (see below), and are also passed to registered trace functions.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 1060)

Unknown directive type "index".

```
.. index::
   single: f_back (frame attribute)
   single: f_code (frame attribute)
   single: f_globals (frame attribute)
   single: f_locals (frame attribute)
   single: f_lasti (frame attribute)
```

```
    single: f_builtins (frame attribute)
```

Special read-only attributes: :attr:`f_back` is to the previous stack frame (towards the caller), or `None` if this is the bottom stack frame; :attr:`f_code` is the code object being executed in this frame; :attr:`f_locals` is the dictionary used to look up local variables; :attr:`f_globals` is used for global variables; :attr:`f_builtins` is used for built-in (intrinsic) names; :attr:`f_lasti` gives the precise instruction (this is an index into the bytecode string of the code object).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1068); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1068); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1068); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1068); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1068); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1068); *backlink*

Unknown interpreted text role "attr".

Accessing `f_code` raises an :ref:`auditing event <auditing>` `object.__getattr__` with arguments `obj` and `"f_code"`.

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

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1079)

Unknown directive type "index".

```
.. index::
   single: f_trace (frame attribute)
   single: f_trace_lines (frame attribute)
   single: f_trace_opcodes (frame attribute)
   single: f_lineno (frame attribute)
```

Special writable attributes: :attr:`f_trace`, if not `None`, is a function called for various events during code execution (this is used by the debugger). Normally an event is triggered for each new source line - this can be disabled by setting :attr:`f_trace_lines` to :const:`False`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 1085); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 1085); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 1085); *backlink*

Unknown interpreted text role "const".

Implementations *may* allow per-opcode events to be requested by setting :attr:`f_trace_opcodes` to :const:`True`. Note that this may lead to undefined interpreter behaviour if exceptions raised by the trace function escape to the function being traced.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 1090); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 1090); *backlink*

Unknown interpreted text role "const".

:attr:`f_lineno` is the current line number of the frame --- writing to this from within a trace function jumps to the given line (only for the bottom-most frame). A debugger can implement a Jump command (aka Set Next Statement) by writing to f_lineno.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 1095); *backlink*

Unknown interpreted text role "attr".

Frame objects support one method:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 1102)

Unknown directive type "method".

```
.. method:: frame.clear()

   This method clears all references to local variables held by the
   frame.  Also, if the frame belonged to a generator, the generator
   is finalized.  This helps break reference cycles involving frame
   objects (for example when catching an exception and storing its
   traceback for later use).

   :exc:`RuntimeError` is raised if the frame is currently executing.

   .. versionadded:: 3.4
```

Traceback objects

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

Unknown directive type "index".

```
.. index::
   object: traceback
   pair: stack; trace
   pair: exception; handler
   pair: execution; stack
   single: exc_info (in module sys)
   single: last_traceback (in module sys)
   single: sys.exc_info
   single: sys.last_traceback
```

Traceback objects represent a stack trace of an exception. A traceback object is implicitly created when an exception occurs, and may also be explicitly created by calling :class:`types.TracebackType`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1127); ***backlink***

Unknown interpreted text role "class".

For implicitly created tracebacks, when the search for an exception handler unwinds the execution stack, at each unwound level a traceback object is inserted in front of the current traceback. When an exception handler is entered, the stack trace is made available to the program. (See section :ref:`try`.) It is accessible as the third item of the tuple returned by `sys.exc_info()`, and as the `__traceback__` attribute of the caught exception.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1131); ***backlink***

Unknown interpreted text role "ref".

When the program contains no suitable handler, the stack trace is written (nicely formatted) to the standard error stream; if the interpreter is interactive, it is also made available to the user as `sys.last_traceback`.

For explicitly created tracebacks, it is up to the creator of the traceback to determine how the `tb_next` attributes should be linked to form a full stack trace.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1148)

Unknown directive type "index".

```
.. index::
   single: tb_frame (traceback attribute)
   single: tb_lineno (traceback attribute)
   single: tb_lasti (traceback attribute)
   statement: try
```

Special read-only attributes: :attr:`tb_frame` points to the execution frame of the current level; :attr:`tb_lineno` gives the line number where the exception occurred; :attr:`tb_lasti` indicates the precise instruction. The line number and last instruction in the traceback may differ from the line number of its frame object if the exception occurred in a :keyword:`try` statement with no matching except clause or with a finally clause.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1154); ***backlink***

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1154); ***backlink***

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1154); ***backlink***

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1154); ***backlink***

Unknown interpreted text role "keyword".

Accessing `tb_frame` raises an :ref:`auditing event <auditing>` `object.__getattr__` with arguments `obj` and `"tb_frame"`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1163); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1166)

Unknown directive type "index".

```
.. index::
   single: tb_next (traceback attribute)
```

Special writable attribute: :attr:`tb_next` is the next level in the stack trace (towards the frame where the exception occurred), or `None` if there is no next level.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1169); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1173)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
   Traceback objects can now be explicitly instantiated from Python code,
   and the ``tb_next`` attribute of existing instances can be updated.
```

Slice objects

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1178)

Unknown directive type "index".

```
.. index:: builtin: slice
```

Slice objects are used to represent slices for :meth:`~object.__getitem__` methods. They are also created by the built-in :func:`slice` function.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1180); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1180); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1184)

Unknown directive type "index".

```
.. index::
   single: start (slice object attribute)
   single: stop (slice object attribute)
   single: step (slice object attribute)
```

Special read-only attributes: :attr:`~slice.start` is the lower bound; :attr:`~slice.stop` is the upper bound; :attr:`~slice.step` is the step value; each is `None` if omitted. These attributes can have any type.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1189); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1189); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1189); *backlink*

Unknown interpreted text role "attr".

Slice objects support one method:

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

Unknown directive type "method".

```
.. method:: slice.indices(self, length)

   This method takes a single integer argument *length* and computes
   information about the slice that the slice object would describe if
   applied to a sequence of *length* items.  It returns a tuple of three
   integers; respectively these are the *start* and *stop* indices and the
   *step* or stride length of the slice. Missing or out-of-bounds indices
   are handled in a manner consistent with regular slices.
```

Static method objects

Static method objects provide a way of defeating the transformation of function objects to method objects described above. A static method object is a wrapper around any other object, usually a user-defined method object. When a static method object is retrieved from a class or a class instance, the object actually returned is the wrapped object, which is not subject to any further transformation. Static method objects are also callable. Static method objects are created by the built-in :func:`staticmethod` constructor.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1205); *backlink*

Unknown interpreted text role "func".

Class method objects

A class method object, like a static method object, is a wrapper around another object that alters the way in which that object is retrieved from classes and class instances. The behaviour of class method objects upon such retrieval is described above, under "User-defined methods". Class method objects are created by the built-in :func:`classmethod` constructor.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1214); *backlink*

Unknown interpreted text role "func".

Special method names

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1226)

Unknown directive type "index".

```
.. index::
   pair: operator; overloading
```

```
single: __getitem__() (mapping object method)
```

A class can implement certain operations that are invoked by special syntax (such as arithmetic operations or subscripting and slicing) by defining methods with special names. This is Python's approach to :dfn:`operator overloading`, allowing classes to define their own behavior with respect to language operators. For instance, if a class defines a method named :meth:`~object.__getitem__`, and `x` is an instance of this class, then `x[i]` is roughly equivalent to `type(x).__getitem__(x, i)`. Except where mentioned, attempts to execute an operation raise an exception when no appropriate method is defined (typically :exc:`AttributeError` or :exc:`TypeError`).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1230); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1230); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1230); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1230); *backlink*

Unknown interpreted text role "exc".

Setting a special method to `None` indicates that the corresponding operation is not available. For example, if a class sets :meth:`~object.__iter__` to `None`, the class is not iterable, so calling :func:`iter` on its instances will raise a :exc:`TypeError` (without falling back to :meth:`~object.__getitem__`). [2]

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1241); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1241); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1241); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1241); *backlink*

Unknown interpreted text role "meth".

When implementing a class that emulates any built-in type, it is important that the emulation only be implemented to the degree that it makes sense for the object being modelled. For example, some sequences may work well with retrieval of individual elements, but extracting a slice may not make sense. (One example of this is the :class:`~xml.dom.NodeList` interface in the W3C's Document Object Model.)

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

Unknown interpreted text role "class".

Basic customization

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1260)

Unknown directive type "method".

```
.. method:: object.__new__(cls[, ...])
```

```
   .. index:: pair: subclassing; immutable types

   Called to create a new instance of class *cls*.  :meth:`__new__` is a static
   method (special-cased so you need not declare it as such) that takes the class
   of which an instance was requested as its first argument.  The remaining
   arguments are those passed to the object constructor expression (the call to the
   class).  The return value of :meth:`__new__` should be the new object instance
   (usually an instance of *cls*).

   Typical implementations create a new instance of the class by invoking the
   superclass's :meth:`__new__` method using ``super().__new__(cls[, ...])``
   with appropriate arguments and then modifying the newly-created instance
   as necessary before returning it.

   If :meth:`__new__` is invoked during object construction and it returns an
   instance of *cls*, then the new instanceâ€™s :meth:`__init__` method
   will be invoked like ``__init__(self[, ...])``, where *self* is the new instance
   and the remaining arguments are the same as were passed to the object constructor.

   If :meth:`__new__` does not return an instance of *cls*, then the new instance's
   :meth:`__init__` method will not be invoked.

   :meth:`__new__` is intended mainly to allow subclasses of immutable types (like
   int, str, or tuple) to customize instance creation.  It is also commonly
   overridden in custom metaclasses in order to customize class creation.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1289)

Unknown directive type "method".

```
.. method:: object.__init__(self[, ...])

   .. index:: pair: class; constructor

   Called after the instance has been created (by :meth:`__new__`), but before
   it is returned to the caller.  The arguments are those passed to the
   class constructor expression.  If a base class has an :meth:`__init__`
   method, the derived class's :meth:`__init__` method, if any, must explicitly
   call it to ensure proper initialization of the base class part of the
   instance; for example: ``super().__init__([args...])``.

   Because :meth:`__new__` and :meth:`__init__` work together in constructing
   objects (:meth:`__new__` to create it, and :meth:`__init__` to customize it),
   no non-``None`` value may be returned by :meth:`__init__`; doing so will
   cause a :exc:`TypeError` to be raised at runtime.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1306)

Unknown directive type "method".

```
.. method:: object.__del__(self)

   .. index::
      single: destructor
      single: finalizer
      statement: del

   Called when the instance is about to be destroyed.  This is also called a
   finalizer or (improperly) a destructor.  If a base class has a
   :meth:`__del__` method, the derived class's :meth:`__del__` method,
   if any, must explicitly call it to ensure proper deletion of the base
   class part of the instance.

   It is possible (though not recommended!) for the :meth:`__del__` method
   to postpone destruction of the instance by creating a new reference to
   it.  This is called object *resurrection*.  It is implementation-dependent
   whether :meth:`__del__` is called a second time when a resurrected object
   is about to be destroyed; the current :term:`CPython` implementation
   only calls it once.

   It is not guaranteed that :meth:`__del__` methods are called for objects
   that still exist when the interpreter exits.

   .. note::

      ``del x`` doesn't directly call ``x.__del__()`` --- the former decrements
      the reference count for ``x`` by one, and the latter is only called when
      ``x``'s reference count reaches zero.

   .. impl-detail::
      It is possible for a reference cycle to prevent the reference count
      of an object from going to zero.  In this case, the cycle will be
      later detected and deleted by the :term:`cyclic garbage collector
      <garbage collection>`.  A common cause of reference cycles is when
      an exception has been caught in a local variable.  The frame's
      locals then reference the exception, which references its own
      traceback, which references the locals of all frames caught in the
      traceback.

      .. seealso::
         Documentation for the :mod:`gc` module.

   .. warning::

      Due to the precarious circumstances under which :meth:`__del__` methods are
      invoked, exceptions that occur during their execution are ignored, and a warning
      is printed to ``sys.stderr`` instead.  In particular:

      * :meth:`__del__` can be invoked when arbitrary code is being executed,
        including from any arbitrary thread.  If :meth:`__del__` needs to take
        a lock or invoke any other blocking resource, it may deadlock as
        the resource may already be taken by the code that gets interrupted
        to execute :meth:`__del__`.

      * :meth:`__del__` can be executed during interpreter shutdown.  As a
        consequence, the global variables it needs to access (including other
        modules) may already have been deleted or set to ``None``. Python
        guarantees that globals whose name begins with a single underscore
        are deleted from their module before other globals are deleted; if
        no other references to such globals exist, this may help in assuring
        that imported modules are still available at the time when the
        :meth:`__del__` method is called.


   .. index::
      single: repr() (built-in function); __repr__() (object method)
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1373)

Unknown directive type "method".

```
.. method:: object.__repr__(self)

   Called by the :func:`repr` built-in function to compute the "official" string
   representation of an object.  If at all possible, this should look like a
   valid Python expression that could be used to recreate an object with the
   same value (given an appropriate environment).  If this is not possible, a
   string of the form ``<...some useful description...>`` should be returned.
   The return value must be a string object. If a class defines :meth:`__repr__`
   but not :meth:`__str__`, then :meth:`__repr__` is also used when an
   "informal" string representation of instances of that class is required.

   This is typically used for debugging, so it is important that the representation
   is information-rich and unambiguous.

   .. index::
      single: string; __str__() (object method)
      single: format() (built-in function); __str__() (object method)
      single: print() (built-in function); __str__() (object method)
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1393)

Unknown directive type "method".

```
.. method:: object.__str__(self)

   Called by :func:`str(object) <str>` and the built-in functions
   :func:`format` and :func:`print` to compute the "informal" or nicely
   printable string representation of an object.  The return value must be a
   :ref:`string <textseq>` object.

   This method differs from :meth:`object.__repr__` in that there is no
   expectation that :meth:`__str__` return a valid Python expression: a more
   convenient or concise representation can be used.

   The default implementation defined by the built-in type :class:`object`
   calls :meth:`object.__repr__`.

   .. XXX what about subclasses of string?
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1410)

Unknown directive type "method".

```
.. method:: object.__bytes__(self)

   .. index:: builtin: bytes

   Called by :ref:`bytes <func-bytes>` to compute a byte-string representation
   of an object. This should return a :class:`bytes` object.

   .. index::
      single: string; __format__() (object method)
      pair: string; conversion
      builtin: print
```

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

Unknown directive type "method".

```
.. method:: object.__format__(self, format_spec)

   Called by the :func:`format` built-in function,
   and by extension, evaluation of :ref:`formatted string literals
   <f-strings>` and the :meth:`str.format` method, to produce a "formatted"
   string representation of an object. The *format_spec* argument is
   a string that contains a description of the formatting options desired.
   The interpretation of the *format_spec* argument is up to the type
   implementing :meth:`__format__`, however most classes will either
   delegate formatting to one of the built-in types, or use a similar
   formatting option syntax.

   See :ref:`formatspec` for a description of the standard formatting syntax.

   The return value must be a string object.

   .. versionchanged:: 3.4
      The __format__ method of ``object`` itself raises a :exc:`TypeError`
      if passed any non-empty string.

   .. versionchanged:: 3.7
      ``object.__format__(x, '')`` is now equivalent to ``str(x)`` rather
      than ``format(str(x), '')``.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1449)

Unknown directive type "method".

```
.. method:: object.__lt__(self, other)
            object.__le__(self, other)
            object.__eq__(self, other)
            object.__ne__(self, other)
            object.__gt__(self, other)
            object.__ge__(self, other)

   .. index::
      single: comparisons

   These are the so-called "rich comparison" methods. The correspondence between
   operator symbols and method names is as follows: ``x<y`` calls ``x.__lt__(y)``,
   ``x<=y`` calls ``x.__le__(y)``, ``x==y`` calls ``x.__eq__(y)``, ``x!=y`` calls
   ``x.__ne__(y)``, ``x>y`` calls ``x.__gt__(y)``, and ``x>=y`` calls
   ``x.__ge__(y)``.

   A rich comparison method may return the singleton ``NotImplemented`` if it does
   not implement the operation for a given pair of arguments. By convention,
   ``False`` and ``True`` are returned for a successful comparison. However, these
   methods can return any value, so if the comparison operator is used in a Boolean
   context (e.g., in the condition of an ``if`` statement), Python will call
   :func:`bool` on the value to determine if the result is true or false.

   By default, ``object`` implements :meth:`__eq__` by using ``is``, returning
   ``NotImplemented`` in the case of a false comparison:
   ``True if x is y else NotImplemented``. For :meth:`__ne__`, by default it
   delegates to :meth:`__eq__` and inverts the result unless it is
   ``NotImplemented``.  There are no other implied relationships among the
   comparison operators or default implementations; for example, the truth of
   ``(x<y or x==y)`` does not imply ``x<=y``. To automatically generate ordering
   operations from a single root operation, see :func:`functools.total_ordering`.
```

```
    See the paragraph on :meth:`__hash__` for
    some important notes on creating :term:`hashable` objects which support
    custom comparison operations and are usable as dictionary keys.

    There are no swapped-argument versions of these methods (to be used when the
    left argument does not support the operation but the right argument does);
    rather, :meth:`__lt__` and :meth:`__gt__` are each other's reflection,
    :meth:`__le__` and :meth:`__ge__` are each other's reflection, and
    :meth:`__eq__` and :meth:`__ne__` are their own reflection.
    If the operands are of different types, and right operand's type is
    a direct or indirect subclass of the left operand's type,
    the reflected method of the right operand has priority, otherwise
    the left operand's method has priority.  Virtual subclassing is
    not considered.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1496)

Unknown directive type "method".

```
.. method:: object.__hash__(self)

   .. index::
      object: dictionary
      builtin: hash

   Called by built-in function :func:`hash` and for operations on members of
   hashed collections including :class:`set`, :class:`frozenset`, and
   :class:`dict`.  The ``__hash__()`` method should return an integer. The only required
   property is that objects which compare equal have the same hash value; it is
   advised to mix together the hash values of the components of the object that
   also play a part in comparison of objects by packing them into a tuple and
   hashing the tuple. Example::

       def __hash__(self):
           return hash((self.name, self.nick, self.color))

   .. note::

     :func:`hash` truncates the value returned from an object's custom
     :meth:`__hash__` method to the size of a :c:type:`Py_ssize_t`.  This is
     typically 8 bytes on 64-bit builds and 4 bytes on 32-bit builds.  If an
     object's   :meth:`__hash__` must interoperate on builds of different bit
     sizes, be sure to check the width on all supported builds.  An easy way
     to do this is with
     ``python -c "import sys; print(sys.hash_info.width)"``.

   If a class does not define an :meth:`__eq__` method it should not define a
   :meth:`__hash__` operation either; if it defines :meth:`__eq__` but not
   :meth:`__hash__`, its instances will not be usable as items in hashable
   collections.  If a class defines mutable objects and implements an
   :meth:`__eq__` method, it should not implement :meth:`__hash__`, since the
   implementation of hashable collections requires that a key's hash value is
   immutable (if the object's hash value changes, it will be in the wrong hash
   bucket).

   User-defined classes have :meth:`__eq__` and :meth:`__hash__` methods
   by default; with them, all objects compare unequal (except with themselves)
   and ``x.__hash__()`` returns an appropriate value such that ``x == y``
   implies both that ``x is y`` and ``hash(x) == hash(y)``.

   A class that overrides :meth:`__eq__` and does not define :meth:`__hash__`
   will have its :meth:`__hash__` implicitly set to ``None``.  When the
   :meth:`__hash__` method of a class is ``None``, instances of the class will
   raise an appropriate :exc:`TypeError` when a program attempts to retrieve
   their hash value, and will also be correctly identified as unhashable when
   checking ``isinstance(obj, collections.abc.Hashable)``.

   If a class that overrides :meth:`__eq__` needs to retain the implementation
   of :meth:`__hash__` from a parent class, the interpreter must be told this
   explicitly by setting ``__hash__ = <ParentClass>.__hash__``.

   If a class that does not override :meth:`__eq__` wishes to suppress hash
   support, it should include ``__hash__ = None`` in the class definition.
   A class which defines its own :meth:`__hash__` that explicitly raises
   a :exc:`TypeError` would be incorrectly identified as hashable by
   an ``isinstance(obj, collections.abc.Hashable)`` call.


   .. note::

      By default, the :meth:`__hash__` values of str and bytes objects are
      "salted" with an unpredictable random value.  Although they
      remain constant within an individual Python process, they are not
      predictable between repeated invocations of Python.
      This is intended to provide protection against a denial-of-service caused
      by carefully-chosen inputs that exploit the worst case performance of a
      dict insertion, O(n\ :sup:`2`) complexity.  See
      http://www.ocert.org/advisories/ocert-2011-003.html for details.

      Changing hash values affects the iteration order of sets.
      Python has never made guarantees about this ordering
      (and it typically varies between 32-bit and 64-bit builds).

      See also :envvar:`PYTHONHASHSEED`.

   .. versionchanged:: 3.3
      Hash randomization is enabled by default.
```

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

Unknown directive type "method".

```
.. method:: object.__bool__(self)

   .. index:: single: __len__() (mapping object method)

   Called to implement truth value testing and the built-in operation
   ``bool()``; should return ``False`` or ``True``.  When this method is not
   defined, :meth:`__len__` is called, if it is defined, and the object is
   considered true if its result is nonzero.  If a class defines neither
   :meth:`__len__` nor :meth:`__bool__`, all its instances are considered
   true.
```

Customizing attribute access

The following methods can be defined to customize the meaning of attribute access (use of, assignment to, or deletion of `x.name`) for class instances.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1600)

Unknown directive type "method".

```
.. method:: object.__getattr__(self, name)

   Called when the default attribute access fails with an :exc:`AttributeError`
   (either :meth:`__getattribute__` raises an :exc:`AttributeError` because
   *name* is not an instance attribute or an attribute in the class tree
   for ``self``; or :meth:`__get__` of a *name* property raises
   :exc:`AttributeError`).  This method should either return the (computed)
   attribute value or raise an :exc:`AttributeError` exception.

   Note that if the attribute is found through the normal mechanism,
   :meth:`__getattr__` is not called.  (This is an intentional asymmetry between
   :meth:`__getattr__` and :meth:`__setattr__`.) This is done both for efficiency
   reasons and because otherwise :meth:`__getattr__` would have no way to access
   other attributes of the instance.  Note that at least for instance variables,
   you can fake total control by not inserting any values in the instance attribute
   dictionary (but instead inserting them in another object).  See the
   :meth:`__getattribute__` method below for a way to actually get total control
   over attribute access.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1620)

Unknown directive type "method".

```
.. method:: object.__getattribute__(self, name)

   Called unconditionally to implement attribute accesses for instances of the
   class. If the class also defines :meth:`__getattr__`, the latter will not be
   called unless :meth:`__getattribute__` either calls it explicitly or raises an
   :exc:`AttributeError`. This method should return the (computed) attribute value
   or raise an :exc:`AttributeError` exception. In order to avoid infinite
   recursion in this method, its implementation should always call the base class
   method with the same name to access any attributes it needs, for example,
   ``object.__getattribute__(self, name)``.

   .. note::

      This method may still be bypassed when looking up special methods as the
      result of implicit invocation via language syntax or built-in functions.
      See :ref:`special-lookup`.

   .. audit-event:: object.__getattr__ obj,name object.__getattribute__

      For certain sensitive attribute accesses, raises an
      :ref:`auditing event <auditing>` ``object.__getattr__`` with arguments
      ``obj`` and ``name``.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1644)

Unknown directive type "method".

```
.. method:: object.__setattr__(self, name, value)

   Called when an attribute assignment is attempted.  This is called instead of
   the normal mechanism (i.e. store the value in the instance dictionary).
   *name* is the attribute name, *value* is the value to be assigned to it.

   If :meth:`__setattr__` wants to assign to an instance attribute, it should
   call the base class method with the same name, for example,
   ``object.__setattr__(self, name, value)``.

   .. audit-event:: object.__setattr__ obj,name,value object.__setattr__

      For certain sensitive attribute assignments, raises an
      :ref:`auditing event <auditing>` ``object.__setattr__`` with arguments
      ``obj``, ``name``, ``value``.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1661)

Unknown directive type "method".

```
.. method:: object.__delattr__(self, name)

   Like :meth:`__setattr__` but for attribute deletion instead of assignment.  This
   should only be implemented if ``del obj.name`` is meaningful for the object.

   .. audit-event:: object.__delattr__ obj,name object.__delattr__

      For certain sensitive attribute deletions, raises an
      :ref:`auditing event <auditing>` ``object.__delattr__`` with arguments
      ``obj`` and ``name``.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1673)

Unknown directive type "method".

```
.. method:: object.__dir__(self)

   Called when :func:`dir` is called on the object. A sequence must be
   returned. :func:`dir` converts the returned sequence to a list and sorts it.
```

Customizing module attribute access

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1682)

Unknown directive type "index".

```
.. index::
   single: __getattr__ (module attribute)
   single: __dir__ (module attribute)
   single: __class__ (module attribute)
```

Special names `__getattr__` and `__dir__` can be also used to customize access to module attributes. The `__getattr__` function at the module level should accept one argument which is the name of an attribute and return the computed value or raise an :exc:`AttributeError`. If an attribute is not found on a module object through the normal lookup, i.e. :meth:`object.__getattribute__`, then `__getattr__` is searched in the module `__dict__` before raising an :exc:`AttributeError`. If found, it is called with the attribute name and the result is returned.

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

Unknown interpreted text role "exc".

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

Unknown interpreted text role "meth".

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

Unknown interpreted text role "exc".

The `__dir__` function should accept no arguments, and return a sequence of strings that represents the names accessible on module. If present, this function overrides the standard :func:`dir` search on a module.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1696); *backlink*

Unknown interpreted text role "func".

For a more fine grained customization of the module behavior (setting attributes, properties, etc.), one can set the `__class__` attribute of a module object to a subclass of :class:`types.ModuleType`. For example:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1700); *backlink*

Unknown interpreted text role "class".

```
import sys
from types import ModuleType

class VerboseModule(ModuleType):
    def __repr__(self):
        return f'Verbose {self.__name__}'

    def __setattr__(self, attr, value):
        print(f'Setting {attr}...')
        super().__setattr__(attr, value)

sys.modules[__name__].__class__ = VerboseModule
```

Note

Defining module `__getattr__` and setting module `__class__` only affect lookups made using the attribute access syntax -- directly accessing the module globals (whether by code within the module, or via a reference to the module's globals dictionary) is unaffected.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1723)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.5
   ``__class__`` module attribute is now writable.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1726)

Unknown directive type "versionadded".

```
.. versionadded:: 3.7
   ``__getattr__`` and ``__dir__`` module attributes.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1729)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`562` - Module __getattr__ and __dir__
      Describes the ``__getattr__`` and ``__dir__`` functions on modules.
```

Implementing Descriptors

The following methods only apply when an instance of the class containing the method (a so-called *descriptor* class) appears in an *owner* class (the descriptor must be in either the owner's class dictionary or in the class dictionary for one of its parents). In the examples below, "the attribute" refers to the attribute whose name is the key of the property in the owner class' :attr:`~object.__dict__`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1740); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1748)

Unknown directive type "method".

```
.. method:: object.__get__(self, instance, owner=None)

   Called to get the attribute of the owner class (class attribute access) or
   of an instance of that class (instance attribute access). The optional
   *owner* argument is the owner class, while *instance* is the instance that
   the attribute was accessed through, or ``None`` when the attribute is
   accessed through the *owner*.

   This method should return the computed attribute value or raise an
   :exc:`AttributeError` exception.

   :PEP:`252` specifies that :meth:`__get__` is callable with one or two
   arguments.  Python's own built-in descriptors support this specification;
   however, it is likely that some third-party tools have descriptors
   that require both arguments.  Python's own :meth:`__getattribute__`
   implementation always passes in both arguments whether they are required
   or not.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1766)

Unknown directive type "method".

```
.. method:: object.__set__(self, instance, value)

   Called to set the attribute on an instance *instance* of the owner class to a
   new value, *value*.

   Note, adding :meth:`__set__` or :meth:`__delete__` changes the kind of
   descriptor to a "data descriptor".  See :ref:`descriptor-invocation` for
   more details.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1775)

Unknown directive type "method".

```
.. method:: object.__delete__(self, instance)

   Called to delete the attribute on an instance *instance* of the owner class.
```

The attribute :attr:`__objclass__` is interpreted by the :mod:`inspect` module as specifying the class where this object was defined (setting this appropriately can assist in runtime introspection of dynamic class attributes). For callables, it may indicate that an instance of the given type (or a subclass) is expected or required as the first positional argument (for example, CPython sets this attribute for unbound methods that are implemented in C).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1780); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1780); *backlink*

Unknown interpreted text role "mod".

Invoking Descriptors

In general, a descriptor is an object attribute with "binding behavior", one whose attribute access has been overridden by methods in the descriptor protocol: :meth:`~object.__get__`, :meth:`~object.__set__`, and :meth:`~object.__delete__`. If any of those methods are defined for an object, it is said to be a descriptor.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1793); *backlink*
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1793); *backlink*
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1793); *backlink*
>
> Unknown interpreted text role "meth".

The default behavior for attribute access is to get, set, or delete the attribute from an object's dictionary. For instance, `a.x` has a lookup chain starting with `a.__dict__['x']`, then `type(a).__dict__['x']`, and continuing through the base classes of `type(a)` excluding metaclasses.

However, if the looked-up value is an object defining one of the descriptor methods, then Python may override the default behavior and invoke the descriptor method instead. Where this occurs in the precedence chain depends on which descriptor methods were defined and how they were called.

The starting point for descriptor invocation is a binding, `a.x`. How the arguments are assembled depends on `a`:

Direct Call
: The simplest and least common call is when user code directly invokes a descriptor method: `x.__get__(a)`.

Instance Binding
: If binding to an object instance, `a.x` is transformed into the call: `type(a).__dict__['x'].__get__(a, type(a))`.

Class Binding
: If binding to a class, `A.x` is transformed into the call: `A.__dict__['x'].__get__(None, A)`.

Super Binding
: A dotted lookup such as `super(A, a).x` searches `a.__class__.__mro__` for a base class `B` following `A` and then returns `B.__dict__['x'].__get__(a, A)`. If not a descriptor, `x` is returned unchanged.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1830)
>
> Unknown directive type "testcode".
>
> ```
> .. testcode::
>     :hide:
>
>     class Desc:
>         def __get__(*args):
>             return args
>
>     class B:
>
>         x = Desc()
>
>     class A(B):
>
>         x = 999
>
>         def m(self):
>             'Demonstrate these two descriptor invocations are equivalent'
>             result1 = super(A, self).x
>             result2 = B.__dict__['x'].__get__(self, A)
>             return result1 == result2
> ```

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1851)
>
> Unknown directive type "doctest".
>
> ```
> .. doctest::
>     :hide:
>
>     >>> a = A()
>     >>> a.__class__.__mro__.index(B) > a.__class__.__mro__.index(A)
>     True
>     >>> super(A, a).x == B.__dict__['x'].__get__(a, A)
> ```

```
        True
        >>> a.m()
        True
```

For instance bindings, the precedence of descriptor invocation depends on which descriptor methods are defined. A descriptor can define any combination of :meth:`~object.__get__`, :meth:`~object.__set__` and :meth:`~object.__delete__`. If it does not define :meth:`__get__`, then accessing the attribute will return the descriptor object itself unless there is a value in the object's instance dictionary. If the descriptor defines :meth:`__set__` and/or :meth:`__delete__`, it is a data descriptor; if it defines neither, it is a non-data descriptor. Normally, data descriptors define both :meth:`__get__` and :meth:`__set__`, while non-data descriptors have just the :meth:`__get__` method. Data descriptors with :meth:`__get__` and :meth:`__set__` (and/or :meth:`__delete__`) defined always override a redefinition in an instance dictionary. In contrast, non-data descriptors can be overridden by instances.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1862**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1862**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1862**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1862**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1862**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1862**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1862**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1862**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1862**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1862**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1862**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1862**); ***backlink***

Unknown interpreted text role "meth".

Python methods (including those decorated with :func:`@staticmethod <staticmethod>` and :func:`@classmethod <classmethod>`) are implemented as non-data descriptors. Accordingly, instances can redefine and override methods. This allows individual instances to acquire behaviors that differ from other instances of the same class.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1876**); ***backlink***

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1876**); ***backlink***

Unknown interpreted text role "func".

The :func:`property` function is implemented as a data descriptor. Accordingly, instances cannot override the behavior of a property.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1882**); ***backlink***

Unknown interpreted text role "func".

__slots__

__*slots*__ allow us to explicitly declare data members (like properties) and deny the creation of :attr:`~object.__dict__` and __*weakref*__ (unless explicitly declared in __*slots*__ or available in a parent.)

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1891**); ***backlink***

Unknown interpreted text role "attr".

The space saved over using :attr:`~object.__dict__` can be significant. Attribute lookup speed can be significantly improved as well.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1895**); ***backlink***

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1898**)

Unknown directive type "data".

```
.. data:: object.__slots__

   This class variable can be assigned a string, iterable, or sequence of
   strings with variable names used by instances.  *__slots__* reserves space
   for the declared variables and prevents the automatic creation of
   :attr:`~object.__dict__`
   and *__weakref__* for each instance.
```

Notes on using __*slots*__

- When inheriting from a class without __*slots*__, the :attr:`~object.__dict__` and __*weakref*__ attribute of the instances will always be accessible.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1910**); ***backlink***

  Unknown interpreted text role "attr".

- Without a :attr:`~object.__dict__` variable, instances cannot be assigned new variables not listed in the __*slots*__ definition. Attempts to assign to an unlisted variable name raises :exc:`AttributeError`. If dynamic assignment of new variables is desired, then add `'__dict__'` to the sequence of strings in the __*slots*__ declaration.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1914**); ***backlink***

  Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1914); *backlink*

Unknown interpreted text role "exc".

- Without a __*weakref*__ variable for each instance, classes defining __*slots*__ do not support :mod:`weak references <weakref>` to its instances. If weak reference support is needed, then add `'__weakref__'` to the sequence of strings in the __*slots*__ declaration.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1921); *backlink*

Unknown interpreted text role "mod".

- __*slots*__ are implemented at the class level by creating :ref:`descriptors <descriptors>` for each variable name. As a result, class attributes cannot be used to set default values for instance variables defined by __*slots*__; otherwise, the class attribute would overwrite the descriptor assignment.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1927); *backlink*

Unknown interpreted text role "ref".

- The action of a __*slots*__ declaration is not limited to the class where it is defined. __*slots*__ declared in parents are available in child classes. However, child subclasses will get a :attr:`~object.__dict__` and __*weakref*__ unless they also define __*slots*__ (which should only contain names of any *additional* slots).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1933); *backlink*

Unknown interpreted text role "attr".

- If a class defines a slot also defined in a base class, the instance variable defined by the base class slot is inaccessible (except by retrieving its descriptor directly from the base class). This renders the meaning of the program undefined. In the future, a check may be added to prevent this.
- Nonempty __*slots*__ does not work for classes derived from "variable-length" built-in types such as :class:`int`, :class:`bytes` and :class:`tuple`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1944); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1944); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1944); *backlink*

Unknown interpreted text role "class".

- Any non-string :term:`iterable` may be assigned to __*slots*__.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 1947); *backlink*

Unknown interpreted text role "term".

- If a :class:`dictionary <dict>` is used to assign __*slots*__, the dictionary keys will be used as the slot names. The values of the dictionary can be used to provide per-attribute docstrings that will be recognised by :func:`inspect.getdoc` and displayed in the output of :func:`help`.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1949**); ***backlink***
>
> Unknown interpreted text role "class".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1949**); ***backlink***
>
> Unknown interpreted text role "func".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1949**); ***backlink***
>
> Unknown interpreted text role "func".

- :attr:`~instance.__class__` assignment works only if both classes have the same __*slots*__.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1954**); ***backlink***
>
> Unknown interpreted text role "attr".

- :ref:`Multiple inheritance <tut-multiple>` with multiple slotted parent classes can be used, but only one parent is allowed to have attributes created by slots (the other bases must have empty slot layouts) - violations raise :exc:`TypeError`.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1957**); ***backlink***
>
> Unknown interpreted text role "ref".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1957**); ***backlink***
>
> Unknown interpreted text role "exc".

- If an :term:`iterator` is used for __*slots*__ then a :term:`descriptor` is created for each of the iterator's values. However, the __*slots*__ attribute will be an empty iterator.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1963**); ***backlink***
>
> Unknown interpreted text role "term".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1963**); ***backlink***
>
> Unknown interpreted text role "term".

Customizing class creation

Whenever a class inherits from another class, :meth:`~object.__init_subclass__` is called on the parent class. This way, it is possible to write classes which change the behavior of subclasses. This is closely related to class decorators, but where class decorators only affect the specific class they're applied to, `__init_subclass__` solely applies to future subclasses of the class defining the method.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1973**); ***backlink***
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 1980**)

Unknown directive type "classmethod".

```
.. classmethod:: object.__init_subclass__(cls)

   This method is called whenever the containing class is subclassed.
   *cls* is then the new subclass. If defined as a normal instance method,
   this method is implicitly converted to a class method.

   Keyword arguments which are given to a new class are passed to
   the parent's class ``__init_subclass__``. For compatibility with
   other classes using ``__init_subclass__``, one should take out the
   needed keyword arguments and pass the others over to the base
   class, as in::

       class Philosopher:
           def __init_subclass__(cls, /, default_name, **kwargs):
               super().__init_subclass__(**kwargs)
               cls.default_name = default_name

       class AustralianPhilosopher(Philosopher, default_name="Bruce"):
           pass

   The default implementation ``object.__init_subclass__`` does
   nothing, but raises an error if it is called with any arguments.

   .. note::

      The metaclass hint ``metaclass`` is consumed by the rest of the type
      machinery, and is never passed to ``__init_subclass__`` implementations.
      The actual metaclass (rather than the explicit hint) can be accessed as
      ``type(cls)``.

   .. versionadded:: 3.6
```

When a class is created, :meth:`type.__new__` scans the class variables and makes callbacks to those with a :meth:`~object.__set_name__` hook.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2013); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2013); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2016)

Unknown directive type "method".

```
.. method:: object.__set_name__(self, owner, name)

   Automatically called at the time the owning class *owner* is
   created. The object has been assigned to *name* in that class::

       class A:
           x = C()  # Automatically calls: x.__set_name__(A, 'x')

   If the class variable is assigned after the class is created,
   :meth:`__set_name__` will not be called automatically.
   If needed, :meth:`__set_name__` can be called directly::

       class A:
          pass

       c = C()
       A.x = c                  # The hook is not called
       c.__set_name__(A, 'x')   # Manually invoke the hook

   See :ref:`class-object-creation` for more details.

   .. versionadded:: 3.6
```

Metaclasses

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2045)

Unknown directive type "index".

```
.. index::
   single: metaclass
   builtin: type
   single: = (equals); class definition
```

By default, classes are constructed using :func:`type`. The class body is executed in a new namespace and the class name is bound locally to the result of `type(name, bases, namespace)`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 2050); *backlink*

Unknown interpreted text role "func".

The class creation process can be customized by passing the `metaclass` keyword argument in the class definition line, or by inheriting from an existing class that included such an argument. In the following example, both `MyClass` and `MySubclass` are instances of `Meta`:

```
class Meta(type):
    pass

class MyClass(metaclass=Meta):
    pass

class MySubclass(MyClass):
    pass
```

Any other keyword arguments that are specified in the class definition are passed through to all metaclass operations described below.

When a class definition is executed, the following steps occur:

- MRO entries are resolved;
- the appropriate metaclass is determined;
- the class namespace is prepared;
- the class body is executed;
- the class object is created.

Resolving MRO entries

If a base that appears in class definition is not an instance of :class:`type`, then an `__mro_entries__` method is searched on it. If found, it is called with the original bases tuple. This method must return a tuple of classes that will be used instead of this base. The tuple may be empty, in such case the original base is ignored.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 2083); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 2089)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`560` - Core support for typing module and generic types
```

Determining the appropriate metaclass

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 2096)

Unknown directive type "index".

```
.. index::
    single: metaclass hint
```

The appropriate metaclass for a class definition is determined as follows:

- if no bases and no explicit metaclass are given, then :func:`type` is used;

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

line 2101); *backlink*

Unknown interpreted text role "func".

- if an explicit metaclass is given and it is *not* an instance of :func:`type`, then it is used directly as the metaclass;

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 2102); *backlink*

Unknown interpreted text role "func".

- if an instance of :func:`type` is given as the explicit metaclass, or bases are defined, then the most derived metaclass is used.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 2104); *backlink*

Unknown interpreted text role "func".

The most derived metaclass is selected from the explicitly specified metaclass (if any) and the metaclasses (i.e. `type(cls)`) of all specified base classes. The most derived metaclass is one which is a subtype of *all* of these candidate metaclasses. If none of the candidate metaclasses meets that criterion, then the class definition will fail with `TypeError`.

Preparing the class namespace

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 2119)

Unknown directive type "index".

```
.. index::
   single: __prepare__ (metaclass method)
```

Once the appropriate metaclass has been identified, then the class namespace is prepared. If the metaclass has a `__prepare__` attribute, it is called as `namespace = metaclass.__prepare__(name, bases, **kwds)` (where the additional keyword arguments, if any, come from the class definition). The `__prepare__` method should be implemented as a :func:`classmethod <classmethod>`. The namespace returned by `__prepare__` is passed in to `__new__`, but when the final class object is created the namespace is copied into a new `dict`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 2122); *backlink*

Unknown interpreted text role "func".

If the metaclass has no `__prepare__` attribute, then the class namespace is initialised as an empty ordered mapping.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 2134)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`3115` - Metaclasses in Python 3000
      Introduced the ``__prepare__`` namespace hook
```

Executing the class body

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst, line 2143)

Unknown directive type "index".

```
.. index::
   single: class; body
```

The class body is executed (approximately) as `exec(body, globals(), namespace)`. The key difference from a normal call to :func:`exec` is that lexical scoping allows the class body (including any methods) to reference names from the current and outer scopes when the class definition occurs inside a function.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2146); *backlink*

Unknown interpreted text role "func".

However, even when the class definition occurs inside the function, methods defined inside the class still cannot see names defined at the class scope. Class variables must be accessed through the first parameter of instance or class methods, or through the implicit lexically scoped `__class__` reference described in the next section.

Creating the class object

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2163)

Unknown directive type "index".

```
.. index::
   single: __class__ (method cell)
   single: __classcell__ (class namespace entry)
```

Once the class namespace has been populated by executing the class body, the class object is created by calling `metaclass(name, bases, namespace, **kwds)` (the additional keywords passed here are the same as those passed to `__prepare__`).

This class object is the one that will be referenced by the zero-argument form of :func:`super`. `__class__` is an implicit closure reference created by the compiler if any methods in a class body refer to either `__class__` or `super`. This allows the zero argument form of :func:`super` to correctly identify the class being defined based on lexical scoping, while the class or instance that was used to make the current call is identified based on the first argument passed to the method.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2173); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2173); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2181)

Unknown directive type "impl-detail".

```
.. impl-detail::

   In CPython 3.6 and later, the ``__class__`` cell is passed to the metaclass
   as a ``__classcell__`` entry in the class namespace. If present, this must
   be propagated up to the ``type.__new__`` call in order for the class to be
   initialised correctly.
   Failing to do so will result in a :exc:`RuntimeError` in Python 3.8.
```

When using the default metaclass :class:`type`, or any metaclass that ultimately calls `type.__new__`, the following additional customization steps are invoked after creating the class object:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2189); *backlink*

Unknown interpreted text role "class".

1. The `type.__new__` method collects all of the attributes in the class namespace that define a :meth:`~object.__set_name__` method;

   System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2193); *backlink*

   Unknown interpreted text role "meth".

2. Those `__set_name__` methods are called with the class being defined and the assigned name of that particular attribute;
3. The :meth:`~object.__init_subclass__` hook is called on the immediate parent of the new class in its method resolution order.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2197); *backlink*

Unknown interpreted text role "meth".

After the class object is created, it is passed to the class decorators included in the class definition (if any) and the resulting object is bound in the local namespace as the defined class.

When a new class is created by `type.__new__`, the object provided as the namespace parameter is copied to a new ordered mapping and the original object is discarded. The new copy is wrapped in a read-only proxy, which becomes the :attr:`~object.__dict__` attribute of the class object.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2204); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2209)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`3135` - New super
      Describes the implicit ``__class__`` closure reference
```

Uses for metaclasses

The potential uses for metaclasses are boundless. Some ideas that have been explored include enum, logging, interface checking, automatic delegation, automatic property creation, proxies, frameworks, and automatic resource locking/synchronization.

Customizing instance and subclass checks

The following methods are used to override the default behavior of the :func:`isinstance` and :func:`issubclass` built-in functions.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2227); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2227); *backlink*

Unknown interpreted text role "func".

In particular, the metaclass :class:`abc.ABCMeta` implements these methods in order to allow the addition of Abstract Base Classes (ABCs) as "virtual base classes" to any class or type (including built-in types), including other ABCs.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2230); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2235)

Unknown directive type "method".

```
.. method:: class.__instancecheck__(self, instance)

   Return true if *instance* should be considered a (direct or indirect)
   instance of *class*. If defined, called to implement ``isinstance(instance,
   class)``.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2242)

Unknown directive type "method".

```
.. method:: class.__subclasscheck__(self, subclass)
```

```
      Return true if *subclass* should be considered a (direct or indirect)
      subclass of *class*.  If defined, called to implement ``issubclass(subclass,
      class)``.
```

Note that these methods are looked up on the type (metaclass) of a class. They cannot be defined as class methods in the actual class. This is consistent with the lookup of special methods that are called on instances, only in this case the instance is itself a class.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2254)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`3119` - Introducing Abstract Base Classes
      Includes the specification for customizing :func:`isinstance` and
      :func:`issubclass` behavior through :meth:`~class.__instancecheck__` and
      :meth:`~class.__subclasscheck__`, with motivation for this functionality
      in the context of adding Abstract Base Classes (see the :mod:`abc`
      module) to the language.
```

Emulating generic types

When using :term:`type annotations<annotation>`, it is often useful to *parameterize* a :term:`generic type` using Python's square-brackets notation. For example, the annotation `list[int]` might be used to signify a :class:`list` in which all the elements are of type :class:`int`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2267); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2267); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2267); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2267); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2272)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`484` - Type Hints
      Introducing Python's framework for type annotations

   :ref:`Generic Alias Types<types-genericalias>`
      Documentation for objects representing parameterized generic classes

   :ref:`Generics`, :ref:`user-defined generics<user-defined-generics>` and :class:`typing.Generic`
      Documentation on how to implement generic classes that can be
      parameterized at runtime and understood by static type-checkers.
```

A class can *generally* only be parameterized if it defines the special class method `__class_getitem__()`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2287)

Unknown directive type "classmethod".

```
.. classmethod:: object.__class_getitem__(cls, key)

   Return an object representing the specialization of a generic class
```

```
    by type arguments found in *key*.

    When defined on a class, ``__class_getitem__()`` is automatically a class
    method. As such, there is no need for it to be decorated with
    :func:`@classmethod<classmethod>` when it is defined.
```

The purpose of *__class_getitem__*

The purpose of :meth:`~object.__class_getitem__` is to allow runtime parameterization of standard-library generic classes in order to more easily apply :term:`type hints<type hint>` to these classes.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2300); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2300); ***backlink***

Unknown interpreted text role "term".

To implement custom generic classes that can be parameterized at runtime and understood by static type-checkers, users should either inherit from a standard library class that already implements :meth:`~object.__class_getitem__`, or inherit from :class:`typing.Generic`, which has its own implementation of `__class_getitem__()`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2304); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2304); ***backlink***

Unknown interpreted text role "class".

Custom implementations of :meth:`~object.__class_getitem__` on classes defined outside of the standard library may not be understood by third-party type-checkers such as mypy. Using `__class_getitem__()` on any class for purposes other than type hinting is discouraged.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2310); ***backlink***

Unknown interpreted text role "meth".

__class_getitem__ versus *__getitem__*

Usually, the :ref:`subscription<subscriptions>` of an object using square brackets will call the :meth:`~object.__getitem__` instance method defined on the object's class. However, if the object being subscribed is itself a class, the class method :meth:`~object.__class_getitem__` may be called instead. `__class_getitem__()` should return a :ref:`GenericAlias<types-genericalias>` object if it is properly defined.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2322); ***backlink***

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2322); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2322); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2322); ***backlink***

Unknown interpreted text role "ref".

Presented with the :term:`expression` `obj[x]`, the Python interpreter follows something like the following process to decide whether :meth:`~object.__getitem__` or :meth:`~object.__class_getitem__` should be called:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2329); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2329); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2329); *backlink*

Unknown interpreted text role "meth".

```
from inspect import isclass

def subscribe(obj, x):
    """Return the result of the expression 'obj[x]'"""

    class_of_obj = type(obj)

    # If the class of obj defines __getitem__,
    # call class_of_obj.__getitem__(obj, x)
    if hasattr(class_of_obj, '__getitem__'):
        return class_of_obj.__getitem__(obj, x)

    # Else, if obj is a class and defines __class_getitem__,
    # call obj.__class_getitem__(x)
    elif isclass(obj) and hasattr(obj, '__class_getitem__'):
        return obj.__class_getitem__(x)

    # Else, raise an exception
    else:
        raise TypeError(
            f"'{class_of_obj.__name__}' object is not subscriptable"
        )
```

In Python, all classes are themselves instances of other classes. The class of a class is known as that class's :term:`metaclass`, and most classes have the :class:`type` class as their metaclass. :class:`type` does not define :meth:`~object.__getitem__`, meaning that expressions such as `list[int]`, `dict[str, float]` and `tuple[str, bytes]` all result in :meth:`~object.__class_getitem__` being called:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2357); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2357); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2357); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2357); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2357); *backlink*

Unknown interpreted text role "meth".

```
>>> # list has class "type" as its metaclass, like most classes:
>>> type(list)
<class 'type'>
>>> type(dict) == type(list) == type(tuple) == type(str) == type(bytes)
True
>>> # "list[int]" calls "list.__class_getitem__(int)"
>>> list[int]
list[int]
>>> # list.__class_getitem__ returns a GenericAlias object:
>>> type(list[int])
<class 'types.GenericAlias'>
```

However, if a class has a custom metaclass that defines :meth:`~object.__getitem__`, subscribing the class may result in different behaviour. An example of this can be found in the :mod:`enum` module:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2376); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2376); *backlink*

Unknown interpreted text role "mod".

```
>>> from enum import Enum
>>> class Menu(Enum):
...     """A breakfast menu"""
...     SPAM = 'spam'
...     BACON = 'bacon'
...
>>> # Enum classes have a custom metaclass:
>>> type(Menu)
<class 'enum.EnumMeta'>
>>> # EnumMeta defines __getitem__,
>>> # so __class_getitem__ is not called,
>>> # and the result is not a GenericAlias object:
>>> Menu['SPAM']
<Menu.SPAM: 'spam'>
>>> type(Menu['SPAM'])
<enum 'Menu'>
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2398)

Unknown directive type "seealso".

```
.. seealso::
   :pep:`560` - Core Support for typing module and generic types
      Introducing :meth:`~object.__class_getitem__`, and outlining when a
      :ref:`subscription<subscriptions>` results in ``__class_getitem__()``
      being called instead of :meth:`~object.__getitem__`
```

Emulating callable objects

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2411)

Unknown directive type "method".

```
.. method:: object.__call__(self[, args...])

   .. index:: pair: call; instance

   Called when the instance is "called" as a function; if this method is defined,
   ``x(arg1, arg2, ...)`` roughly translates to ``type(x).__call__(x, arg1, ...)``.
```

Emulating container types

The following methods can be defined to implement container objects. Containers usually are :term:`sequences <sequence>` (such as :class:`lists <list>` or :class:`tuples <tuple>`) or :term:`mappings <mapping>` (like :class:`dictionaries <dict>`), but can represent other containers as well. The first set of methods is used either to emulate a sequence or to emulate a mapping; the difference is that for a sequence, the allowable keys should be the integers *k* for which `0 <= k < N` where *N* is the length of the sequence, or :class:`slice` objects, which define a range of items. It is also recommended that mappings provide the methods :meth:`keys`, :meth:`values`, :meth:`items`, :meth:`get`, :meth:`clear`, :meth:`setdefault`, :meth:`pop`, :meth:`popitem`, :meth:`!copy`, and :meth:`update` behaving similar to those for Python's standard :class:`dictionary <dict>` objects. The :mod:`collections.abc` module provides a :class:`~collections.abc.MutableMapping` :term:`abstract base class` to help create those methods from a base set of :meth:`~object.__getitem__`, :meth:`~object.__setitem__`, :meth:`~object.__delitem__`, and :meth:`keys`. Mutable sequences should provide methods :meth:`append`, :meth:`count`, :meth:`index`, :meth:`extend`, :meth:`insert`, :meth:`pop`, :meth:`remove`, :meth:`reverse` and :meth:`sort`, like Python standard :class:`list` objects. Finally, sequence types should implement addition (meaning concatenation) and multiplication (meaning repetition) by defining the methods :meth:`~object.__add__`, :meth:`~object.__radd__`, :meth:`~object.__iadd__`, :meth:`~object.__mul__`, :meth:`~object.__rmul__` and :meth:`~object.__imul__` described below; they should not define other numerical operators. It is recommended that both mappings and sequences implement the :meth:`~object.__contains__` method to allow efficient use of the `in` operator; for mappings, `in` should search the mapping's keys; for sequences, it should search through the values. It is further recommended that both mappings and sequences implement the :meth:`~object.__iter__` method to allow efficient iteration through the container; for mappings, :meth:`__iter__` should iterate through the object's keys; for sequences, it should iterate through the values.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 2424**); ***backlink***

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 2424**); ***backlink***

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 2424**); ***backlink***

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 2424**); ***backlink***

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 2424**); ***backlink***

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 2424**); ***backlink***

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 2424**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 2424**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 2424**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 2424**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 2424**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 2424**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 2424**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2424); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2458)

Unknown directive type "method".

```
.. method:: object.__len__(self)

   .. index::
      builtin: len
      single: __bool__() (object method)

   Called to implement the built-in function :func:`len`.  Should return the length
   of the object, an integer ``>=`` 0.  Also, an object that doesn't define a
   :meth:`__bool__` method and whose :meth:`__len__` method returns zero is
   considered to be false in a Boolean context.

   .. impl-detail::

      In CPython, the length is required to be at most :attr:`sys.maxsize`.
      If the length is larger than :attr:`!sys.maxsize` some features (such as
      :func:`len`) may raise :exc:`OverflowError`.  To prevent raising
      :exc:`!OverflowError` by truth value testing, an object must define a
      :meth:`__bool__` method.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2478)

Unknown directive type "method".

```
.. method:: object.__length_hint__(self)

   Called to implement :func:`operator.length_hint`. Should return an estimated
   length for the object (which may be greater or less than the actual length).
   The length must be an integer ``>=`` 0. The return value may also be
   :const:`NotImplemented`, which is treated the same as if the
   ``__length_hint__`` method didn't exist at all. This method is purely an
   optimization and is never required for correctness.

   .. versionadded:: 3.4
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2490)

Unknown directive type "index".

```
.. index:: object: slice
```

Note

Slicing is done exclusively with the following three methods. A call like

```
a[1:2] = b
```

is translated to

```
a[slice(1, 2, None)] = b
```

and so forth. Missing slice items are always filled in with `None`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2505)

Unknown directive type "method".

```
.. method:: object.__getitem__(self, key)

   Called to implement evaluation of ``self[key]``. For :term:`sequence` types,
   the accepted keys should be integers and slice objects.  Note that the
   special interpretation of negative indexes (if the class wishes to emulate a
   :term:`sequence` type) is up to the :meth:`__getitem__` method. If *key* is
   of an inappropriate type, :exc:`TypeError` may be raised; if of a value
   outside the set of indexes for the sequence (after any special
   interpretation of negative values), :exc:`IndexError` should be raised. For
   :term:`mapping` types, if *key* is missing (not in the container),
   :exc:`KeyError` should be raised.

   .. note::

      :keyword:`for` loops expect that an :exc:`IndexError` will be raised for
      illegal indexes to allow proper detection of the end of the sequence.

   .. note::

      When :ref:`subscripting<subscriptions>` a *class*, the special
      class method :meth:`~object.__class_getitem__` may be called instead of
      ``__getitem__()``. See :ref:`classgetitem-versus-getitem` for more
      details.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2530)

Unknown directive type "method".

```
.. method:: object.__setitem__(self, key, value)

   Called to implement assignment to ``self[key]``.  Same note as for
   :meth:`__getitem__`.  This should only be implemented for mappings if the
   objects support changes to the values for keys, or if new keys can be added, or
   for sequences if elements can be replaced.  The same exceptions should be raised
   for improper *key* values as for the :meth:`__getitem__` method.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2539)

Unknown directive type "method".

```
.. method:: object.__delitem__(self, key)

   Called to implement deletion of ``self[key]``.  Same note as for
   :meth:`__getitem__`.  This should only be implemented for mappings if the
   objects support removal of keys, or for sequences if elements can be removed
   from the sequence.  The same exceptions should be raised for improper *key*
   values as for the :meth:`__getitem__` method.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2548)

Unknown directive type "method".

```
.. method:: object.__missing__(self, key)

   Called by :class:`dict`\ .\ :meth:`__getitem__` to implement ``self[key]`` for dict subclasses
   when key is not in the dictionary.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2554)

Unknown directive type "method".

```
.. method:: object.__iter__(self)

   This method is called when an :term:`iterator` is required for a container.
   This method should return a new iterator object that can iterate over all the
   objects in the container.  For mappings, it should iterate over the keys of
   the container.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2562)

Unknown directive type "method".

```
.. method:: object.__reversed__(self)

   Called (if present) by the :func:`reversed` built-in to implement
   reverse iteration.  It should return a new iterator object that iterates
   over all the objects in the container in reverse order.

   If the :meth:`__reversed__` method is not provided, the :func:`reversed`
   built-in will fall back to using the sequence protocol (:meth:`__len__` and
   :meth:`__getitem__`).  Objects that support the sequence protocol should
   only provide :meth:`__reversed__` if they can provide an implementation
   that is more efficient than the one provided by :func:`reversed`.
```

The membership test operators (:keyword:`in` and :keyword:`not in`) are normally implemented as an iteration through a container. However, container objects can supply the following special method with a more efficient implementation, which also does not require the object be iterable.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2575); ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2575); ***backlink***

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2580)

Unknown directive type "method".

```
.. method:: object.__contains__(self, item)

   Called to implement membership test operators.  Should return true if *item*
   is in *self*, false otherwise.  For mapping objects, this should consider the
   keys of the mapping rather than the values or the key-item pairs.

   For objects that don't define :meth:`__contains__`, the membership test first
   tries iteration via :meth:`__iter__`, then the old sequence iteration
   protocol via :meth:`__getitem__`, see :ref:`this section in the language
   reference <membership-test-details>`.
```

Emulating numeric types

The following methods can be defined to emulate numeric objects. Methods corresponding to operations that are not supported by the particular kind of number implemented (e.g., bitwise operations for non-integral numbers) should be left undefined.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2603)

Unknown directive type "method".

```
.. method:: object.__add__(self, other)
            object.__sub__(self, other)
            object.__mul__(self, other)
            object.__matmul__(self, other)
            object.__truediv__(self, other)
            object.__floordiv__(self, other)
            object.__mod__(self, other)
            object.__divmod__(self, other)
            object.__pow__(self, other[, modulo])
            object.__lshift__(self, other)
            object.__rshift__(self, other)
            object.__and__(self, other)
            object.__xor__(self, other)
            object.__or__(self, other)

   .. index::
      builtin: divmod
      builtin: pow
      builtin: pow

   These methods are called to implement the binary arithmetic operations
   (``+``, ``-``, ``*``, ``@``, ``/``, ``//``, ``%``, :func:`divmod`,
   :func:`pow`, ``**``, ``<<``, ``>>``, ``&``, ``^``, ``|``).  For instance, to
   evaluate the expression ``x + y``, where *x* is an instance of a class that
   has an :meth:`__add__` method, ``type(x).__add__(x, y)`` is called.  The
   :meth:`__divmod__` method should be the equivalent to using
   :meth:`__floordiv__` and :meth:`__mod__`; it should not be related to
   :meth:`__truediv__`.  Note that :meth:`__pow__` should be defined to accept
   an optional third argument if the ternary version of the built-in :func:`pow`
   function is to be supported.

   If one of those methods does not support the operation with the supplied
   arguments, it should return ``NotImplemented``.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2638)

Unknown directive type "method".

```
.. method:: object.__radd__(self, other)
            object.__rsub__(self, other)
            object.__rmul__(self, other)
            object.__rmatmul__(self, other)
            object.__rtruediv__(self, other)
            object.__rfloordiv__(self, other)
            object.__rmod__(self, other)
            object.__rdivmod__(self, other)
            object.__rpow__(self, other[, modulo])
            object.__rlshift__(self, other)
            object.__rrshift__(self, other)
            object.__rand__(self, other)
            object.__rxor__(self, other)
            object.__ror__(self, other)

   .. index::
      builtin: divmod
      builtin: pow

   These methods are called to implement the binary arithmetic operations
   (``+``, ``-``, ``*``, ``@``, ``/``, ``//``, ``%``, :func:`divmod`,
   :func:`pow`, ``**``, ``<<``, ``>>``, ``&``, ``^``, ``|``) with reflected
   (swapped) operands.  These functions are only called if the left operand does
   not support the corresponding operation [#]_ and the operands are of different
   types. [#]_ For instance, to evaluate the expression ``x - y``, where *y* is
   an instance of a class that has an :meth:`__rsub__` method,
   ``type(y).__rsub__(y, x)`` is called if ``type(x).__sub__(x, y)`` returns
   *NotImplemented*.

   .. index:: builtin: pow

   Note that ternary :func:`pow` will not try calling :meth:`__rpow__` (the
   coercion rules would become too complicated).

   .. note::

      If the right operand's type is a subclass of the left operand's type and
      that subclass provides a different implementation of the reflected method
      for the operation, this method will be called before the left operand's
      non-reflected method. This behavior allows subclasses to override their
      ancestors' operations.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2681)

Unknown directive type "method".

```
.. method:: object.__iadd__(self, other)
            object.__isub__(self, other)
            object.__imul__(self, other)
            object.__imatmul__(self, other)
            object.__itruediv__(self, other)
            object.__ifloordiv__(self, other)
            object.__imod__(self, other)
            object.__ipow__(self, other[, modulo])
            object.__ilshift__(self, other)
            object.__irshift__(self, other)
            object.__iand__(self, other)
            object.__ixor__(self, other)
            object.__ior__(self, other)

   These methods are called to implement the augmented arithmetic assignments
   (``+=``, ``-=``, ``*=``, ``@=``, ``/=``, ``//=``, ``%=``, ``**=``, ``<<=``,
   ``>>=``, ``&=``, ``^=``, ``|=``).  These methods should attempt to do the
   operation in-place (modifying *self*) and return the result (which could be,
   but does not have to be, *self*).  If a specific method is not defined, the
   augmented assignment falls back to the normal methods.  For instance, if *x*
   is an instance of a class with an :meth:`__iadd__` method, ``x += y`` is
   equivalent to ``x = x.__iadd__(y)`` . Otherwise, ``x.__add__(y)`` and
   ``y.__radd__(x)`` are considered, as with the evaluation of ``x + y``. In
   certain situations, augmented assignment can result in unexpected errors (see
   :ref:`faq-augmented-assignment-tuple-error`), but this behavior is in fact
   part of the data model.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2709)

Unknown directive type "method".

```
.. method:: object.__neg__(self)
            object.__pos__(self)
            object.__abs__(self)
            object.__invert__(self)

   .. index:: builtin: abs

   Called to implement the unary arithmetic operations (``-``, ``+``, :func:`abs`
   and ``~``).
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2720)

Unknown directive type "method".

```
.. method:: object.__complex__(self)
            object.__int__(self)
            object.__float__(self)

   .. index::
      builtin: complex
      builtin: int
      builtin: float

   Called to implement the built-in functions :func:`complex`,
   :func:`int` and :func:`float`.  Should return a value
   of the appropriate type.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2734)

Unknown directive type "method".

```
.. method:: object.__index__(self)

   Called to implement :func:`operator.index`, and whenever Python needs to
   losslessly convert the numeric object to an integer object (such as in
   slicing, or in the built-in :func:`bin`, :func:`hex` and :func:`oct`
   functions). Presence of this method indicates that the numeric object is
   an integer type.  Must return an integer.

   If :meth:`__int__`, :meth:`__float__` and :meth:`__complex__` are not
   defined then corresponding built-in functions :func:`int`, :func:`float`
   and :func:`complex` fall back to :meth:`__index__`.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2747)

Unknown directive type "method".

```
.. method:: object.__round__(self, [,ndigits])
            object.__trunc__(self)
            object.__floor__(self)
            object.__ceil__(self)

   .. index:: builtin: round

   Called to implement the built-in function :func:`round` and :mod:`math`
   functions :func:`~math.trunc`, :func:`~math.floor` and :func:`~math.ceil`.
   Unless *ndigits* is passed to :meth:`!__round__` all these methods should
   return the value of the object truncated to an :class:`~numbers.Integral`
   (typically an :class:`int`).

   The built-in function :func:`int` falls back to :meth:`__trunc__` if neither
   :meth:`__int__` nor :meth:`__index__` is defined.

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

With Statement Context Managers

A :dfn:`context manager` is an object that defines the runtime context to be established when executing a :keyword:`with` statement. The context manager handles the entry into, and the exit from, the desired runtime context for the execution of the block of code. Context managers are normally invoked using the :keyword:`!with` statement (described in section :ref:`with`), but can also be used by directly invoking their methods.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2772); *backlink*

Unknown interpreted text role "dfn".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2772); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2772); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2772); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2779)

Unknown directive type "index".

```
.. index::
   statement: with
   single: context manager
```

Typical uses of context managers include saving and restoring various kinds of global state, locking and unlocking resources, closing opened files, etc.

For more information on context managers, see :ref:`typecontextmanager`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2786); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2789)

Unknown directive type "method".

```
.. method:: object.__enter__(self)

   Enter the runtime context related to this object. The :keyword:`with` statement
   will bind this method's return value to the target(s) specified in the
   :keyword:`!as` clause of the statement, if any.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2796)

Unknown directive type "method".

```
.. method:: object.__exit__(self, exc_type, exc_value, traceback)

   Exit the runtime context related to this object. The parameters describe the
   exception that caused the context to be exited. If the context was exited
   without an exception, all three arguments will be :const:`None`.

   If an exception is supplied, and the method wishes to suppress the exception
   (i.e., prevent it from being propagated), it should return a true value.
   Otherwise, the exception will be processed normally upon exit from this method.

   Note that :meth:`__exit__` methods should not reraise the passed-in exception;
```

```
    this is the caller's responsibility.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2810)

Unknown directive type "seealso".

```
.. seealso::

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

Customizing positional arguments in class pattern matching

When using a class name in a pattern, positional arguments in the pattern are not allowed by default, i.e. `case MyClass(x, y)` is typically invalid without special support in `MyClass`. To be able to use that kind of patterns, the class needs to define a *__match_args__* attribute.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2827)

Unknown directive type "data".

```
.. data:: object.__match_args__

   This class variable can be assigned a tuple of strings. When this class is
   used in a class pattern with positional arguments, each positional argument will
   be converted into a keyword argument, using the corresponding value in
   *__match_args__* as the keyword. The absence of this attribute is equivalent to
   setting it to ``()``.
```

For example, if `MyClass.__match_args__` is `("left", "center", "right")` that means that `case MyClass(x, y)` is equivalent to `case MyClass(left=x, center=y)`. Note that the number of arguments in the pattern must be smaller than or equal to the number of elements in *__match_args__*; if it is larger, the pattern match attempt will raise a :exc:`TypeError`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2835); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2841)

Unknown directive type "versionadded".

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

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2843)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`634` - Structural Pattern Matching
      The specification for the Python ``match`` statement.
```

Special method lookup

For custom classes, implicit invocations of special methods are only guaranteed to work correctly if defined on an object's type, not in the object's instance dictionary. That behaviour is the reason why the following code raises an exception:

```
>>> class C:
...     pass
...
>>> c = C()
>>> c.__len__ = lambda: 5
>>> len(c)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: object of type 'C' has no len()
```

The rationale behind this behaviour lies with a number of special methods such as :meth:`~object.__hash__` and :meth:`~object.__repr__` that are implemented by all objects, including type objects. If the implicit lookup of these methods used the conventional lookup process, they would fail when invoked on the type object itself:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 2869**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 2869**); ***backlink***

Unknown interpreted text role "meth".

```
>>> 1 .__hash__() == hash(1)
True
>>> int.__hash__() == hash(int)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: descriptor '__hash__' of 'int' object needs an argument
```

Incorrectly attempting to invoke an unbound method of a class in this way is sometimes referred to as 'metaclass confusion', and is avoided by bypassing the instance when looking up special methods:

```
>>> type(1).__hash__(1) == hash(1)
True
>>> type(int).__hash__(int) == hash(int)
True
```

In addition to bypassing any instance attributes in the interest of correctness, implicit special method lookup generally also bypasses the :meth:`~object.__getattribute__` method even of the object's metaclass:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 2892**); ***backlink***

Unknown interpreted text role "meth".

```
>>> class Meta(type):
...     def __getattribute__(*args):
...         print("Metaclass getattribute invoked")
...         return type.__getattribute__(*args)
...
>>> class C(object, metaclass=Meta):
...     def __len__(self):
...         return 10
...     def __getattribute__(*args):
...         print("Class getattribute invoked")
...         return object.__getattribute__(*args)
...
>>> c = C()
>>> c.__len__()                 # Explicit lookup via instance
Class getattribute invoked
10
>>> type(c).__len__(c)          # Explicit lookup via type
Metaclass getattribute invoked
10
>>> len(c)                      # Implicit lookup
10
```

Bypassing the :meth:`~object.__getattribute__` machinery in this fashion provides significant scope for speed optimisations within the interpreter, at the cost of some flexibility in the handling of special methods (the special method *must* be set on the class object itself in order to be consistently invoked by the interpreter).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 2918**); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 2925**)

Unknown directive type "index".

```
.. index::
   single: coroutine
```

Coroutines

Awaitable Objects

An :term:`awaitable` object generally implements an :meth:`~object.__await__` method. :term:`Coroutine objects <coroutine>` returned from :keyword:`async def` functions are awaitable.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2935); ***backlink***

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2935); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2935); ***backlink***

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2935); ***backlink***

Unknown interpreted text role "keyword".

Note

The :term:`generator iterator` objects returned from generators decorated with :func:`types.coroutine` are also awaitable, but they do not implement :meth:`~object.__await__`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2941); ***backlink***

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2941); ***backlink***

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2941); ***backlink***

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2945)

Unknown directive type "method".

```
.. method:: object.__await__(self)

   Must return an :term:`iterator`.  Should be used to implement
   :term:`awaitable` objects.  For instance, :class:`asyncio.Future` implements
   this method to be compatible with the :keyword:`await` expression.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2951)

Unknown directive type "versionadded".

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

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2953)

Unknown directive type "seealso".

```
.. seealso:: :pep:`492` for additional information about awaitable objects.
```

Coroutine Objects

:term:`Coroutine objects <coroutine>` are :term:`awaitable` objects. A coroutine's execution can be controlled by calling :meth:`~object.__await__` and iterating over the result. When the coroutine has finished executing and returns, the iterator raises :exc:`StopIteration`, and the exception's :attr:`~StopIteration.value` attribute holds the return value. If the coroutine raises an exception, it is propagated by the iterator. Coroutines should not directly raise unhandled :exc:`StopIteration` exceptions.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2961); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2961); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2961); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2961); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2961); *backlink*

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2961); *backlink*

Unknown interpreted text role "exc".

Coroutines also have the methods listed below, which are analogous to those of generators (see :ref:`generator-methods`). However, unlike generators, coroutines do not directly support iteration.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2969); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2973)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.5.2
   It is a :exc:`RuntimeError` to await on a coroutine more than once.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2977)

Unknown directive type "method".

```
.. method:: coroutine.send(value)

   Starts or resumes execution of the coroutine.  If *value* is ``None``,
   this is equivalent to advancing the iterator returned by
   :meth:`~object.__await__`.  If *value* is not ``None``, this method delegates
   to the :meth:`~generator.send` method of the iterator that caused
   the coroutine to suspend.  The result (return value,
   :exc:`StopIteration`, or other exception) is the same as when
   iterating over the :meth:`__await__` return value, described above.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2987)

Unknown directive type "method".

```
.. method:: coroutine.throw(value)
            coroutine.throw(type[, value[, traceback]])

   Raises the specified exception in the coroutine.  This method delegates
   to the :meth:`~generator.throw` method of the iterator that caused
   the coroutine to suspend, if it has such a method.  Otherwise,
   the exception is raised at the suspension point.  The result
   (return value, :exc:`StopIteration`, or other exception) is the same as
   when iterating over the :meth:`~object.__await__` return value, described
   above.  If the exception is not caught in the coroutine, it propagates
   back to the caller.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 2999)

Unknown directive type "method".

```
.. method:: coroutine.close()

   Causes the coroutine to clean itself up and exit.  If the coroutine
   is suspended, this method first delegates to the :meth:`~generator.close`
   method of the iterator that caused the coroutine to suspend, if it
   has such a method.  Then it raises :exc:`GeneratorExit` at the
   suspension point, causing the coroutine to immediately clean itself up.
   Finally, the coroutine is marked as having finished executing, even if
   it was never started.

   Coroutine objects are automatically closed using the above process when
   they are about to be destroyed.
```

Asynchronous Iterators

An *asynchronous iterator* can call asynchronous code in its `__anext__` method.

Asynchronous iterators can be used in an :keyword:`async for` statement.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 3020); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 3022)

Unknown directive type "method".

```
.. method:: object.__aiter__(self)

   Must return an *asynchronous iterator* object.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 3026)

Unknown directive type "method".

```
.. method:: object.__anext__(self)

   Must return an *awaitable* resulting in a next value of the iterator.  Should
   raise a :exc:`StopAsyncIteration` error when the iteration is over.
```

An example of an asynchronous iterable object:

```
class Reader:
    async def readline(self):
        ...

    def __aiter__(self):
        return self

    async def __anext__(self):
        val = await self.readline()
        if val == b'':
            raise StopAsyncIteration
        return val
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 3046)

Unknown directive type "versionadded".

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

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 3048)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
   Prior to Python 3.7, :meth:`~object.__aiter__` could return an *awaitable*
   that would resolve to an
   :term:`asynchronous iterator <asynchronous iterator>`.

   Starting with Python 3.7, :meth:`~object.__aiter__` must return an
   asynchronous iterator object.  Returning anything else
   will result in a :exc:`TypeError` error.
```

Asynchronous Context Managers

An *asynchronous context manager* is a *context manager* that is able to suspend execution in its `__aenter__` and `__aexit__` methods.

Asynchronous context managers can be used in an :keyword:`async with` statement.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 3066); *backlink*

Unknown interpreted text role "keyword".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 3068)

Unknown directive type "method".

```
.. method:: object.__aenter__(self)

   Semantically similar to :meth:`__enter__`, the only
   difference being that it must return an *awaitable*.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 3073)

Unknown directive type "method".

```
.. method:: object.__aexit__(self, exc_type, exc_value, traceback)

   Semantically similar to :meth:`__exit__`, the only
   difference being that it must return an *awaitable*.
```

An example of an asynchronous context manager class:

```
class AsyncContextManager:
    async def __aenter__(self):
        await log('entering context')

    async def __aexit__(self, exc_type, exc, tb):
        await log('exiting context')
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, line 3087)

Unknown directive type "versionadded".

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

Footnotes

[1] It *is* possible in some cases to change an object's type, under certain controlled conditions. It generally isn't a good idea though, since it can lead to some very strange behaviour if it is handled incorrectly.

[2] The :meth:`~object.__hash__`, :meth:`~object.__iter__`, :meth:`~object.__reversed__`, and :meth:`~object.__contains__`

methods have special handling for this; others will still raise a :exc:`TypeError`, but may do so by relying on the behavior that `None` is not callable.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 3096**); *backlink*
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 3096**); *backlink*
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 3096**); *backlink*
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 3096**); *backlink*
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 3096**); *backlink*
>
> Unknown interpreted text role "exc".

[3] "Does not support" here means that the class has no such method, or the method returns `NotImplemented`. Do not set the method to `None` if you want to force fallback to the right operand's reflected methodâ€"that will instead have the opposite effect of explicitly *blocking* such fallback.

[4] For operands of the same type, it is assumed that if the non-reflected method -- such as :meth:`~object.__add__` -- fails then the overall operation is not supported, which is why the reflected method is not called.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]datamodel.rst`, **line 3108**); *backlink*
>
> Unknown interpreted text role "meth".