Object Protocol

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 1)

Unknown directive type "highlight".

```
.. highlight:: c
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 9)

Unknown directive type "c:var".

```
.. c:var:: PyObject* Py_NotImplemented

   The ``NotImplemented`` singleton, used to signal that an operation is
   not implemented for the given type combination.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 15)

Unknown directive type "c:macro".

```
.. c:macro:: Py_RETURN_NOTIMPLEMENTED

   Properly handle returning :c:data:`Py_NotImplemented` from within a C
   function (that is, increment the reference count of NotImplemented and
   return it).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 22)

Unknown directive type "c:function".

```
.. c:function:: int PyObject_Print(PyObject *o, FILE *fp, int flags)

   Print an object *o*, on file *fp*.  Returns ``-1`` on error.  The flags argument
   is used to enable certain printing options.  The only option currently supported
   is :const:`Py_PRINT_RAW`; if given, the :func:`str` of the object is written
   instead of the :func:`repr`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 30)

Unknown directive type "c:function".

```
.. c:function:: int PyObject_HasAttr(PyObject *o, PyObject *attr_name)

   Returns ``1`` if *o* has the attribute *attr_name*, and ``0`` otherwise.  This
   is equivalent to the Python expression ``hasattr(o, attr_name)``.  This function
   always succeeds.

   Note that exceptions which occur while calling :meth:`__getattr__` and
   :meth:`__getattribute__` methods will get suppressed.
   To get error reporting use :c:func:`PyObject_GetAttr()` instead.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 41)

Unknown directive type "c:function".

```
.. c:function:: int PyObject_HasAttrString(PyObject *o, const char *attr_name)
```

```
Returns ``1`` if *o* has the attribute *attr_name*, and ``0`` otherwise.  This
is equivalent to the Python expression ``hasattr(o, attr_name)``.  This function
always succeeds.

Note that exceptions which occur while calling :meth:`__getattr__` and
:meth:`__getattribute__` methods and creating a temporary string object
will get suppressed.
To get error reporting use :c:func:`PyObject_GetAttrString()` instead.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 53)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_GetAttr(PyObject *o, PyObject *attr_name)

   Retrieve an attribute named *attr_name* from object *o*. Returns the attribute
   value on success, or ``NULL`` on failure.  This is the equivalent of the Python
   expression ``o.attr_name``.
```

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

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_GetAttrString(PyObject *o, const char *attr_name)

   Retrieve an attribute named *attr_name* from object *o*. Returns the attribute
   value on success, or ``NULL`` on failure. This is the equivalent of the Python
   expression ``o.attr_name``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 67)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_GenericGetAttr(PyObject *o, PyObject *name)

   Generic attribute getter function that is meant to be put into a type
   object's ``tp_getattro`` slot.  It looks for a descriptor in the dictionary
   of classes in the object's MRO as well as an attribute in the object's
   :attr:`~object.__dict__` (if present).  As outlined in :ref:`descriptors`,
   data descriptors take preference over instance attributes, while non-data
   descriptors don't.  Otherwise, an :exc:`AttributeError` is raised.
```

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

Unknown directive type "c:function".

```
.. c:function:: int PyObject_SetAttr(PyObject *o, PyObject *attr_name, PyObject *v)

   Set the value of the attribute named *attr_name*, for object *o*, to the value
   *v*. Raise an exception and return ``-1`` on failure;
   return ``0`` on success.  This is the equivalent of the Python statement
   ``o.attr_name = v``.

   If *v* is ``NULL``, the attribute is deleted. This behaviour is deprecated
   in favour of using :c:func:`PyObject_DelAttr`, but there are currently no
   plans to remove it.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 89)

Unknown directive type "c:function".

```
.. c:function:: int PyObject_SetAttrString(PyObject *o, const char *attr_name, PyObject *v)

   Set the value of the attribute named *attr_name*, for object *o*, to the value
   *v*. Raise an exception and return ``-1`` on failure;
   return ``0`` on success.  This is the equivalent of the Python statement
   ``o.attr_name = v``.

   If *v* is ``NULL``, the attribute is deleted, but this feature is
   deprecated in favour of using :c:func:`PyObject_DelAttrString`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 100)

Unknown directive type "c:function".

```
.. c:function:: int PyObject_GenericSetAttr(PyObject *o, PyObject *name, PyObject *value)

   Generic attribute setter and deleter function that is meant
   to be put into a type object's :c:member:`~PyTypeObject.tp_setattro`
   slot.  It looks for a data descriptor in the
   dictionary of classes in the object's MRO, and if found it takes preference
   over setting or deleting the attribute in the instance dictionary. Otherwise, the
   attribute is set or deleted in the object's :attr:`~object.__dict__` (if present).
   On success, ``0`` is returned, otherwise an :exc:`AttributeError`
   is raised and ``-1`` is returned.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 112)

Unknown directive type "c:function".

```
.. c:function:: int PyObject_DelAttr(PyObject *o, PyObject *attr_name)

   Delete attribute named *attr_name*, for object *o*. Returns ``-1`` on failure.
   This is the equivalent of the Python statement ``del o.attr_name``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 118)

Unknown directive type "c:function".

```
.. c:function:: int PyObject_DelAttrString(PyObject *o, const char *attr_name)

   Delete attribute named *attr_name*, for object *o*. Returns ``-1`` on failure.
   This is the equivalent of the Python statement ``del o.attr_name``.
```

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

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_GenericGetDict(PyObject *o, void *context)

   A generic implementation for the getter of a ``__dict__`` descriptor. It
   creates the dictionary if necessary.

   .. versionadded:: 3.3
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 132)

Unknown directive type "c:function".

```
.. c:function:: int PyObject_GenericSetDict(PyObject *o, PyObject *value, void *context)

   A generic implementation for the setter of a ``__dict__`` descriptor. This
```

```
implementation does not allow the dictionary to be deleted.

.. versionadded:: 3.3
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 140)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_RichCompare(PyObject *o1, PyObject *o2, int opid)

   Compare the values of *o1* and *o2* using the operation specified by *opid*,
   which must be one of :const:`Py_LT`, :const:`Py_LE`, :const:`Py_EQ`,
   :const:`Py_NE`, :const:`Py_GT`, or :const:`Py_GE`, corresponding to ``<``,
   ``<=``, ``==``, ``!=``, ``>``, or ``>=`` respectively. This is the equivalent of
   the Python expression ``o1 op o2``, where ``op`` is the operator corresponding
   to *opid*. Returns the value of the comparison on success, or ``NULL`` on failure.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 150)

Unknown directive type "c:function".

```
.. c:function:: int PyObject_RichCompareBool(PyObject *o1, PyObject *o2, int opid)

   Compare the values of *o1* and *o2* using the operation specified by *opid*,
   which must be one of :const:`Py_LT`, :const:`Py_LE`, :const:`Py_EQ`,
   :const:`Py_NE`, :const:`Py_GT`, or :const:`Py_GE`, corresponding to ``<``,
   ``<=``, ``==``, ``!=``, ``>``, or ``>=`` respectively. Returns ``-1`` on error,
   ``0`` if the result is false, ``1`` otherwise. This is the equivalent of the
   Python expression ``o1 op o2``, where ``op`` is the operator corresponding to
   *opid*.
```

Note

If *o1* and *o2* are the same object, :c:func:`PyObject_RichCompareBool` will always return `1` for :const:`Py_EQ` and `0` for :const:`Py_NE`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 161); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 161); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 161); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 164)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_Repr(PyObject *o)

   .. index:: builtin: repr

   Compute a string representation of object *o*.  Returns the string
```

```
   representation on success, ``NULL`` on failure.  This is the equivalent of the
   Python expression ``repr(o)``.  Called by the :func:`repr` built-in function.

   .. versionchanged:: 3.4
      This function now includes a debug assertion to help ensure that it
      does not silently discard an active exception.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 176)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_ASCII(PyObject *o)

   .. index:: builtin: ascii

   As :c:func:`PyObject_Repr`, compute a string representation of object *o*, but
   escape the non-ASCII characters in the string returned by
   :c:func:`PyObject_Repr` with ``\x``, ``\u`` or ``\U`` escapes.  This generates
   a string similar to that returned by :c:func:`PyObject_Repr` in Python 2.
   Called by the :func:`ascii` built-in function.

   .. index:: string; PyObject_Str (C function)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 189)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_Str(PyObject *o)

   Compute a string representation of object *o*.  Returns the string
   representation on success, ``NULL`` on failure.  This is the equivalent of the
   Python expression ``str(o)``.  Called by the :func:`str` built-in function
   and, therefore, by the :func:`print` function.

   .. versionchanged:: 3.4
      This function now includes a debug assertion to help ensure that it
      does not silently discard an active exception.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 201)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_Bytes(PyObject *o)

   .. index:: builtin: bytes

   Compute a bytes representation of object *o*.  ``NULL`` is returned on
   failure and a bytes object on success.  This is equivalent to the Python
   expression ``bytes(o)``, when *o* is not an integer.  Unlike ``bytes(o)``,
   a TypeError is raised when *o* is an integer instead of a zero-initialized
   bytes object.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 212)

Unknown directive type "c:function".

```
.. c:function:: int PyObject_IsSubclass(PyObject *derived, PyObject *cls)

   Return ``1`` if the class *derived* is identical to or derived from the class
   *cls*, otherwise return ``0``.  In case of an error, return ``-1``.

   If *cls* is a tuple, the check will be done against every entry in *cls*.
   The result will be ``1`` when at least one of the checks returns ``1``,
   otherwise it will be ``0``.

   If *cls* has a :meth:`~class.__subclasscheck__` method, it will be called to
```

```
   determine the subclass status as described in :pep:`3119`.  Otherwise,
   *derived* is a subclass of *cls* if it is a direct or indirect subclass,
   i.e. contained in ``cls.__mro__``.

   Normally only class objects, i.e. instances of :class:`type` or a derived
   class, are considered classes.  However, objects can override this by having
   a :attr:`__bases__` attribute (which must be a tuple of base classes).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 231)

Unknown directive type "c:function".

```
.. c:function:: int PyObject_IsInstance(PyObject *inst, PyObject *cls)

   Return ``1`` if *inst* is an instance of the class *cls* or a subclass of
   *cls*, or ``0`` if not.  On error, returns ``-1`` and sets an exception.

   If *cls* is a tuple, the check will be done against every entry in *cls*.
   The result will be ``1`` when at least one of the checks returns ``1``,
   otherwise it will be ``0``.

   If *cls* has a :meth:`~class.__instancecheck__` method, it will be called to
   determine the subclass status as described in :pep:`3119`.  Otherwise, *inst*
   is an instance of *cls* if its class is a subclass of *cls*.

   An instance *inst* can override what is considered its class by having a
   :attr:`__class__` attribute.

   An object *cls* can override if it is considered a class, and what its base
   classes are, by having a :attr:`__bases__` attribute (which must be a tuple
   of base classes).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 252)

Unknown directive type "c:function".

```
.. c:function:: Py_hash_t PyObject_Hash(PyObject *o)

   .. index:: builtin: hash

   Compute and return the hash value of an object *o*.  On failure, return ``-1``.
   This is the equivalent of the Python expression ``hash(o)``.

   .. versionchanged:: 3.2
      The return type is now Py_hash_t.  This is a signed integer the same size
      as Py_ssize_t.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 264)

Unknown directive type "c:function".

```
.. c:function:: Py_hash_t PyObject_HashNotImplemented(PyObject *o)

   Set a :exc:`TypeError` indicating that ``type(o)`` is not hashable and return ``-1``.
   This function receives special treatment when stored in a ``tp_hash`` slot,
   allowing a type to explicitly indicate to the interpreter that it is not
   hashable.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 272)

Unknown directive type "c:function".

```
.. c:function:: int PyObject_IsTrue(PyObject *o)

   Returns ``1`` if the object *o* is considered to be true, and ``0`` otherwise.
```

```
   This is equivalent to the Python expression ``not not o``.  On failure, return
   ``-1``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 279)

Unknown directive type "c:function".

```
.. c:function:: int PyObject_Not(PyObject *o)

   Returns ``0`` if the object *o* is considered to be true, and ``1`` otherwise.
   This is equivalent to the Python expression ``not o``.  On failure, return
   ``-1``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 286)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_Type(PyObject *o)

   .. index:: builtin: type

   When *o* is non-``NULL``, returns a type object corresponding to the object type
   of object *o*. On failure, raises :exc:`SystemError` and returns ``NULL``.  This
   is equivalent to the Python expression ``type(o)``. This function increments the
   reference count of the return value. There's really no reason to use this
   function instead of the :c:func:`Py_TYPE()` function, which returns a
   pointer of type :c:type:`PyTypeObject*`, except when the incremented reference
   count is needed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 299)

Unknown directive type "c:function".

```
.. c:function:: int PyObject_TypeCheck(PyObject *o, PyTypeObject *type)

   Return non-zero if the object *o* is of type *type* or a subtype of *type*, and
   ``0`` otherwise.  Both parameters must be non-``NULL``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 305)

Unknown directive type "c:function".

```
.. c:function:: Py_ssize_t PyObject_Size(PyObject *o)
                Py_ssize_t PyObject_Length(PyObject *o)

   .. index:: builtin: len

   Return the length of object *o*.  If the object *o* provides either the sequence
   and mapping protocols, the sequence length is returned.  On error, ``-1`` is
   returned.  This is the equivalent to the Python expression ``len(o)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 315)

Unknown directive type "c:function".

```
.. c:function:: Py_ssize_t PyObject_LengthHint(PyObject *o, Py_ssize_t defaultvalue)

   Return an estimated length for the object *o*. First try to return its
   actual length, then an estimate using :meth:`~object.__length_hint__`, and
   finally return the default value. On error return ``-1``. This is the
   equivalent to the Python expression ``operator.length_hint(o, defaultvalue)``.
```

```
   .. versionadded:: 3.4
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 325)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_GetItem(PyObject *o, PyObject *key)

   Return element of *o* corresponding to the object *key* or ``NULL`` on failure.
   This is the equivalent of the Python expression ``o[key]``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 331)

Unknown directive type "c:function".

```
.. c:function:: int PyObject_SetItem(PyObject *o, PyObject *key, PyObject *v)

   Map the object *key* to the value *v*.  Raise an exception and
   return ``-1`` on failure; return ``0`` on success.  This is the
   equivalent of the Python statement ``o[key] = v``.  This function *does
   not* steal a reference to *v*.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 339)

Unknown directive type "c:function".

```
.. c:function:: int PyObject_DelItem(PyObject *o, PyObject *key)

   Remove the mapping for the object *key* from the object *o*.  Return ``-1``
   on failure.  This is equivalent to the Python statement ``del o[key]``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 345)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_Dir(PyObject *o)

   This is equivalent to the Python expression ``dir(o)``, returning a (possibly
   empty) list of strings appropriate for the object argument, or ``NULL`` if there
   was an error.  If the argument is ``NULL``, this is like the Python ``dir()``,
   returning the names of the current locals; in this case, if no execution frame
   is active then ``NULL`` is returned but :c:func:`PyErr_Occurred` will return false.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 354)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_GetIter(PyObject *o)

   This is equivalent to the Python expression ``iter(o)``. It returns a new
   iterator for the object argument, or the object  itself if the object is already
   an iterator.  Raises :exc:`TypeError` and returns ``NULL`` if the object cannot be
   iterated.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)object.rst, line 362)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyObject_GetAIter(PyObject *o)

   This is the equivalent to the Python expression ``aiter(o)``. Takes an
   :class:`AsyncIterable` object and returns an :class:`AsyncIterator` for it.
   This is typically a new iterator but if the argument is an
   :class:`AsyncIterator`, this returns itself. Raises :exc:`TypeError` and
   returns ``NULL`` if the object cannot be iterated.

   .. versionadded:: 3.10
```