Type Objects

Creating Heap-Allocated Types

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.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]type.rst, line 8)

Unknown directive type "index".

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

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 11)

Unknown directive type "c:type".

```
.. c:type:: PyTypeObject

   The C structure of the objects used to describe built-in types.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 16)

Unknown directive type "c:var".

```
.. c:var:: PyTypeObject PyType_Type

   This is the type object for type objects; it is the same object as
   :class:`type` in the Python layer.
```

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

Unknown directive type "c:function".

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

   Return non-zero if the object *o* is a type object, including instances of
   types derived from the standard type object.  Return 0 in all other cases.
   This function always succeeds.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 29)

Unknown directive type "c:function".

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

   Return non-zero if the object *o* is a type object, but not a subtype of
   the standard type object.  Return 0 in all other cases.  This function
   always succeeds.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 36)

Unknown directive type "c:function".

```
.. c:function:: unsigned int PyType_ClearCache()
```

```
   Clear the internal lookup cache. Return the current version tag.
```

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

Unknown directive type "c:function".

```
.. c:function:: unsigned long PyType_GetFlags(PyTypeObject* type)

   Return the :c:member:`~PyTypeObject.tp_flags` member of *type*. This function is primarily
   meant for use with `Py_LIMITED_API`; the individual flag bits are
   guaranteed to be stable across Python releases, but access to
   :c:member:`~PyTypeObject.tp_flags` itself is not part of the limited API.

   .. versionadded:: 3.2

   .. versionchanged:: 3.4
      The return type is now ``unsigned long`` rather than ``long``.
```

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

Unknown directive type "c:function".

```
.. c:function:: void PyType_Modified(PyTypeObject *type)

   Invalidate the internal lookup cache for the type and all of its
   subtypes.  This function must be called after any manual
   modification of the attributes or base classes of the type.
```

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

Unknown directive type "c:function".

```
.. c:function:: int PyType_HasFeature(PyTypeObject *o, int feature)

   Return non-zero if the type object *o* sets the feature *feature*.
   Type features are denoted by single bit flags.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 66)

Unknown directive type "c:function".

```
.. c:function:: int PyType_IS_GC(PyTypeObject *o)

   Return true if the type object includes support for the cycle detector; this
   tests the type flag :const:`Py_TPFLAGS_HAVE_GC`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 72)

Unknown directive type "c:function".

```
.. c:function:: int PyType_IsSubtype(PyTypeObject *a, PyTypeObject *b)

   Return true if *a* is a subtype of *b*.

   This function only checks for actual subtypes, which means that
   :meth:`~class.__subclasscheck__` is not called on *b*.  Call
   :c:func:`PyObject_IsSubclass` to do the same check that :func:`issubclass`
   would do.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 82)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyType_GenericAlloc(PyTypeObject *type, Py_ssize_t nitems)
```

```
   Generic handler for the :c:member:`~PyTypeObject.tp_alloc` slot of a type object.  Use
   Python's default memory allocation mechanism to allocate a new instance and
   initialize all its contents to ``NULL``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 88)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyType_GenericNew(PyTypeObject *type, PyObject *args, PyObject *kwds)

   Generic handler for the :c:member:`~PyTypeObject.tp_new` slot of a type object.  Create a
   new instance using the type's :c:member:`~PyTypeObject.tp_alloc` slot.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 93)

Unknown directive type "c:function".

```
.. c:function:: int PyType_Ready(PyTypeObject *type)

   Finalize a type object.  This should be called on all type objects to finish
   their initialization.  This function is responsible for adding inherited slots
   from a type's base class.  Return ``0`` on success, or return ``-1`` and sets an
   exception on error.

   .. note::
      If some of the base classes implements the GC protocol and the provided
      type does not include the :const:`Py_TPFLAGS_HAVE_GC` in its flags, then
      the GC protocol will be automatically implemented from its parents. On
      the contrary, if the type being created does include
      :const:`Py_TPFLAGS_HAVE_GC` in its flags then it **must** implement the
      GC protocol itself by at least implementing the
      :c:member:`~PyTypeObject.tp_traverse` handle.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 109)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyType_GetName(PyTypeObject *type)

   Return the type's name. Equivalent to getting the type's ``__name__`` attribute.

   .. versionadded:: 3.11
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 115)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyType_GetQualName(PyTypeObject *type)

   Return the type's qualified name. Equivalent to getting the
   type's ``__qualname__`` attribute.

   .. versionadded:: 3.11
```

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

Unknown directive type "c:function".

```
.. c:function:: void* PyType_GetSlot(PyTypeObject *type, int slot)

   Return the function pointer stored in the given slot. If the
   result is ``NULL``, this indicates that either the slot is ``NULL``,
   or that the function was called with invalid parameters.
   Callers will typically cast the result pointer into the appropriate
   function type.

   See :c:member:`PyType_Slot.slot` for possible values of the *slot* argument.

   .. versionadded:: 3.4

   .. versionchanged:: 3.10
```

```
        :c:func:`PyType_GetSlot` can now accept all types.
        Previously, it was limited to :ref:`heap types <heap-types>`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 138)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyType_GetModule(PyTypeObject *type)

   Return the module object associated with the given type when the type was
   created using :c:func:`PyType_FromModuleAndSpec`.

   If no module is associated with the given type, sets :py:class:`TypeError`
   and returns ``NULL``.

   This function is usually used to get the module in which a method is defined.
   Note that in such a method, ``PyType_GetModule(Py_TYPE(self))``
   may not return the intended result.
   ``Py_TYPE(self)`` may be a *subclass* of the intended class, and subclasses
   are not necessarily defined in the same module as their superclass.
   See :c:type:`PyCMethod` to get the class that defines the method.
   See :c:func:`PyType_GetModuleByDef` for cases when ``PyCMethod`` cannot
   be used.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 157)

Unknown directive type "c:function".

```
.. c:function:: void* PyType_GetModuleState(PyTypeObject *type)

   Return the state of the module object associated with the given type.
   This is a shortcut for calling :c:func:`PyModule_GetState()` on the result
   of :c:func:`PyType_GetModule`.

   If no module is associated with the given type, sets :py:class:`TypeError`
   and returns ``NULL``.

   If the *type* has an associated module but its state is ``NULL``,
   returns ``NULL`` without setting an exception.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 171)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyType_GetModuleByDef(PyTypeObject *type, struct PyModuleDef *def)

   Find the first superclass whose module was created from
   the given :c:type:`PyModuleDef` *def*, and return that module.

   If no module is found, raises a :py:class:`TypeError` and returns ``NULL``.

   This function is intended to be used together with
   :c:func:`PyModule_GetState()` to get module state from slot methods (such as
   :c:member:`~PyTypeObject.tp_init` or :c:member:`~PyNumberMethods.nb_add`)
   and other places where a method's defining class cannot be passed using the
   :c:type:`PyCMethod` calling convention.

   .. versionadded:: 3.11
```

The following functions and structs are used to create :ref:`heap types <heap-types>`.

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

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 193)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyType_FromModuleAndSpec(PyObject *module, PyType_Spec *spec, PyObject *bas

   Creates and returns a :ref:`heap type <heap-types>` from the *spec*
   (:const:`Py_TPFLAGS_HEAPTYPE`).

   The *bases* argument can be used to specify base classes; it can either
   be only one class or a tuple of classes.
   If *bases* is ``NULL``, the *Py_tp_bases* slot is used instead.
   If that also is ``NULL``, the *Py_tp_base* slot is used instead.
   If that also is ``NULL``, the new type derives from :class:`object`.

   The *module* argument can be used to record the module in which the new
   class is defined. It must be a module object or ``NULL``.
   If not ``NULL``, the module is associated with the new type and can later be
   retrieved with :c:func:`PyType_GetModule`.
   The associated module is not inherited by subclasses; it must be specified
   for each class individually.

   This function calls :c:func:`PyType_Ready` on the new type.

   .. versionadded:: 3.9

   .. versionchanged:: 3.10

      The function now accepts a single class as the *bases* argument and
      ``NULL`` as the ``tp_doc`` slot.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 220)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyType_FromSpecWithBases(PyType_Spec *spec, PyObject *bases)

   Equivalent to ``PyType_FromModuleAndSpec(NULL, spec, bases)``.

   .. versionadded:: 3.3
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 226)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyType_FromSpec(PyType_Spec *spec)

   Equivalent to ``PyType_FromSpecWithBases(spec, NULL)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 230)

Unknown directive type "c:type".

```
.. c:type:: PyType_Spec

   Structure defining a type's behavior.

   .. c:member:: const char* PyType_Spec.name

      Name of the type, used to set :c:member:`PyTypeObject.tp_name`.

   .. c:member:: int PyType_Spec.basicsize
   .. c:member:: int PyType_Spec.itemsize

      Size of the instance in bytes, used to set
      :c:member:`PyTypeObject.tp_basicsize` and
      :c:member:`PyTypeObject.tp_itemsize`.

   .. c:member:: int PyType_Spec.flags

      Type flags, used to set :c:member:`PyTypeObject.tp_flags`.

      If the ``Py_TPFLAGS_HEAPTYPE`` flag is not set,
      :c:func:`PyType_FromSpecWithBases` sets it automatically.

   .. c:member:: PyType_Slot *PyType_Spec.slots

      Array of :c:type:`PyType_Slot` structures.
      Terminated by the special slot value ``{0, NULL}``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]type.rst, line 257)

Unknown directive type "c:type".

```
.. c:type:: PyType_Slot

   Structure defining optional functionality of a type, containing a slot ID
   and a value pointer.

   .. c:member:: int PyType_Slot.slot

      A slot ID.

      Slot IDs are named like the field names of the structures
      :c:type:`PyTypeObject`, :c:type:`PyNumberMethods`,
      :c:type:`PySequenceMethods`, :c:type:`PyMappingMethods` and
      :c:type:`PyAsyncMethods` with an added ``Py_`` prefix.
      For example, use:

      * ``Py_tp_dealloc`` to set :c:member:`PyTypeObject.tp_dealloc`
      * ``Py_nb_add`` to set :c:member:`PyNumberMethods.nb_add`
      * ``Py_sq_length`` to set :c:member:`PySequenceMethods.sq_length`

      The following fields cannot be set at all using :c:type:`PyType_Spec` and
      :c:type:`PyType_Slot`:

      * :c:member:`~PyTypeObject.tp_dict`
      * :c:member:`~PyTypeObject.tp_mro`
      * :c:member:`~PyTypeObject.tp_cache`
      * :c:member:`~PyTypeObject.tp_subclasses`
      * :c:member:`~PyTypeObject.tp_weaklist`
      * :c:member:`~PyTypeObject.tp_vectorcall`
      * :c:member:`~PyTypeObject.tp_weaklistoffset`
        (see :ref:`PyMemberDef <pymemberdef-offsets>`)
      * :c:member:`~PyTypeObject.tp_dictoffset`
        (see :ref:`PyMemberDef <pymemberdef-offsets>`)
      * :c:member:`~PyTypeObject.tp_vectorcall_offset`
        (see :ref:`PyMemberDef <pymemberdef-offsets>`)

      Setting :c:data:`Py_tp_bases` or :c:data:`Py_tp_base` may be
      problematic on some platforms.
      To avoid issues, use the *bases* argument of
      :py:func:`PyType_FromSpecWithBases` instead.

     .. versionchanged:: 3.9

        Slots in :c:type:`PyBufferProcs` may be set in the unlimited API.

     .. versionchanged:: 3.11
        :c:member:`~PyBufferProcs.bf_getbuffer` and
        :c:member:`~PyBufferProcs.bf_releasebuffer` are now available
        under the limited API.

   .. c:member:: void *PyType_Slot.pfunc

      The desired value of the slot. In most cases, this is a pointer
      to a function.

      Slots other than ``Py_tp_doc`` may not be ``NULL``.
```