Defining Extension Types: Assorted Topics

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 1)

Unknown directive type "highlight".

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

This section aims to give a quick fly-by on the various type methods you can implement and what they do.

Here is the definition of :c:type:`PyTypeObject`, with some fields only used in :ref:`debug builds <debug-build>` omitted:

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

Unknown interpreted text role "c:type".

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

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 17)

Unknown directive type "literalinclude".

```
.. literalinclude:: ../includes/typestruct.h
```

Now that's a *lot* of methods. Don't worry too much though -- if you have a type you want to define, the chances are very good that you will only implement a handful of these.

As you probably expect by now, we're going to go over this and give more information about the various handlers. We won't go in the order they are defined in the structure, because there is a lot of historical baggage that impacts the ordering of the fields. It's often easiest to find an example that includes the fields you need and then change the values to suit your new type.

```
const char *tp_name; /* For printing */
```

The name of the type -- as mentioned in the previous chapter, this will appear in various places, almost entirely for diagnostic purposes. Try to choose something that will be helpful in such a situation!

```
Py_ssize_t tp_basicsize, tp_itemsize; /* For allocation */
```

These fields tell the runtime how much memory to allocate when new objects of this type are created. Python has some built-in support for variable length structures (think: strings, tuples) which is where the :c:member:`~PyTypeObject.tp_itemsize` field comes in. This will be dealt with later.

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

Unknown interpreted text role "c:member".

```
const char *tp_doc;
```

Here you can put a string (or its address) that you want returned when the Python script references `obj.__doc__` to retrieve the doc string.

Now we come to the basic type methods -- the ones most extension types will implement.

Finalization and De-allocation

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 56)

Unknown directive type "index".

```
.. index::
   single: object; deallocation
   single: deallocation, object
   single: object; finalization
   single: finalization, of objects
```

```
destructor tp_dealloc;
```

This function is called when the reference count of the instance of your type is reduced to zero and the Python interpreter wants to reclaim it. If your type has memory to free or other clean-up to perform, you can put it here. The object itself needs to be freed here as well. Here is an example of this function:

```
static void
newdatatype_dealloc(newdatatypeobject *obj)
{
    free(obj->obj_UnderlyingDatatypePtr);
    Py_TYPE(obj)->tp_free((PyObject *)obj);
}
```

If your type supports garbage collection, the destructor should call :c:func:`PyObject_GC_UnTrack` before clearing any member fields:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 79); *backlink*

Unknown interpreted text role "c:func".

```
static void
newdatatype_dealloc(newdatatypeobject *obj)
{
    PyObject_GC_UnTrack(obj);
    Py_CLEAR(obj->other_obj);
    ...
    Py_TYPE(obj)->tp_free((PyObject *)obj);
}
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 91)

Unknown directive type "index".

```
.. index::
   single: PyErr_Fetch()
   single: PyErr_Restore()
```

One important requirement of the deallocator function is that it leaves any pending exceptions alone. This is important since deallocators are frequently called as the interpreter unwinds the Python stack; when the stack is unwound due to an exception (rather than normal returns), nothing is done to protect the deallocators from seeing that an exception has already been set. Any actions which a deallocator performs which may cause additional Python code to be executed may detect that an exception has been set. This can lead to misleading errors from the interpreter. The proper way to protect against this is to save a pending exception before performing the unsafe action, and restoring it when done. This can be done using the :c:func:`PyErr_Fetch` and :c:func:`PyErr_Restore` functions:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 95); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 95); *backlink*

Unknown interpreted text role "c:func".

```
static void
my_dealloc(PyObject *obj)
{
    MyObject *self = (MyObject *) obj;
    PyObject *cbresult;

    if (self->my_callback != NULL) {
        PyObject *err_type, *err_value, *err_traceback;
```

```
        /* This saves the current exception state */
        PyErr_Fetch(&err_type, &err_value, &err_traceback);

        cbresult = PyObject_CallNoArgs(self->my_callback);
        if (cbresult == NULL)
            PyErr_WriteUnraisable(self->my_callback);
        else
            Py_DECREF(cbresult);

        /* This restores the saved exception state */
        PyErr_Restore(err_type, err_value, err_traceback);

        Py_DECREF(self->my_callback);
    }
    Py_TYPE(obj)->tp_free((PyObject*)self);
}
```

Note

There are limitations to what you can safely do in a deallocator function. First, if your type supports garbage collection (using :c:member:`~PyTypeObject.tp_traverse` and/or :c:member:`~PyTypeObject.tp_clear`), some of the object's members can have been cleared or finalized by the time :c:member:`~PyTypeObject.tp_dealloc` is called. Second, in :c:member:`~PyTypeObject.tp_dealloc`, your object is in an unstable state: its reference count is equal to zero. Any call to a non-trivial object or API (as in the example above) might end up calling :c:member:`~PyTypeObject.tp_dealloc` again, causing a double free and a crash.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 134); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 134); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 134); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 134); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 134); *backlink*

Unknown interpreted text role "c:member".

Starting with Python 3.4, it is recommended not to put any complex finalization code in :c:member:`~PyTypeObject.tp_dealloc`, and instead use the new :c:member:`~PyTypeObject.tp_finalize` type method.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 143); *backlink*

Unknown interpreted text role "c:member".

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

resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 143); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 147)

Unknown directive type "seealso".

```
.. seealso::
   :pep:`442` explains the new finalization scheme.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 150)

Unknown directive type "index".

```
.. index::
   single: string; object representation
   builtin: repr
```

Object Presentation

In Python, there are two ways to generate a textual representation of an object: the :func:`repr` function, and the :func:`str` function. (The :func:`print` function just calls :func:`str`.) These handlers are both optional.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 157); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 157); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 157); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 157); *backlink*

Unknown interpreted text role "func".

```
reprfunc tp_repr;
reprfunc tp_str;
```

The :c:member:`~PyTypeObject.tp_repr` handler should return a string object containing a representation of the instance for which it is called. Here is a simple example:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 166); *backlink*

Unknown interpreted text role "c:member".

```
static PyObject *
newdatatype_repr(newdatatypeobject * obj)
{
    return PyUnicode_FromFormat("Repr-ified_newdatatype{{size:%d}}",
                                obj->obj_UnderlyingDatatypePtr->size);
}
```

If no :c:member:`~PyTypeObject.tp_repr` handler is specified, the interpreter will supply a representation that uses the type's :c:member:`~PyTypeObject.tp_name` and a uniquely-identifying value for the object.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 177); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 177); *backlink*

Unknown interpreted text role "c:member".

The :c:member:`~PyTypeObject.tp_str` handler is to :func:`str` what the :c:member:`~PyTypeObject.tp_repr` handler described above is to :func:`repr`; that is, it is called when Python code calls :func:`str` on an instance of your object. Its implementation is very similar to the :c:member:`~PyTypeObject.tp_repr` function, but the resulting string is intended for human consumption. If :c:member:`~PyTypeObject.tp_str` is not specified, the :c:member:`~PyTypeObject.tp_repr` handler is used instead.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 181); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 181); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 181); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 181); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 181); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 181); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 181); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 181); *backlink*

Unknown interpreted text role "c:member".

Here is a simple example:

```
static PyObject *
newdatatype_str(newdatatypeobject * obj)
{
    return PyUnicode_FromFormat("Stringified_newdatatype{{size:%d}}",
                                obj->obj_UnderlyingDatatypePtr->size);
}
```

Attribute Management

For every object which can support attributes, the corresponding type must provide the functions that control how the attributes are resolved. There needs to be a function which can retrieve attributes (if any are defined), and another to set attributes (if setting attributes is allowed). Removing an attribute is a special case, for which the new value passed to the handler is `NULL`.

Python supports two pairs of attribute handlers; a type that supports attributes only needs to implement the functions for one pair. The difference is that one pair takes the name of the attribute as a :c:type:`char\*`, while the other accepts a :c:type:`PyObject\*`. Each type can use whichever pair makes more sense for the implementation's convenience.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 208); *backlink***
>
> Unknown interpreted text role "c:type".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 208); *backlink***
>
> Unknown interpreted text role "c:type".

```
getattrfunc  tp_getattr;        /* char * version */
setattrfunc  tp_setattr;
/* ... */
getattrofunc tp_getattro;       /* PyObject * version */
setattrofunc tp_setattro;
```

If accessing attributes of an object is always a simple operation (this will be explained shortly), there are generic implementations which can be used to provide the :c:type:`PyObject\*` version of the attribute management functions. The actual need for type-specific attribute handlers almost completely disappeared starting with Python 2.2, though there are many examples which have not been updated to use some of the new generic mechanism that is available.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 220); *backlink***
>
> Unknown interpreted text role "c:type".

Generic Attribute Management

Most extension types only use *simple* attributes. So, what makes the attributes simple? There are only a couple of conditions that must be met:

1. The name of the attributes must be known when :c:func:`PyType_Ready` is called.

   > **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 236); *backlink***
   >
   > Unknown interpreted text role "c:func".

2. No special processing is needed to record that an attribute was looked up or set, nor do actions need to be taken based on the value.

Note that this list does not place any restrictions on the values of the attributes, when the values are computed, or how relevant data is stored.

When :c:func:`PyType_Ready` is called, it uses three tables referenced by the type object to create :term:`descriptor`s which are placed in the dictionary of the type object. Each descriptor controls access to one attribute of the instance object. Each of the tables is optional; if all three are `NULL`, instances of the type will only have attributes that are inherited from their base type, and should leave the :c:member:`~PyTypeObject.tp_getattro` and :c:member:`~PyTypeObject.tp_setattro` fields `NULL` as well, allowing the base type to handle attributes.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 245); *backlink***
>
> Unknown interpreted text role "c:func".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 245); *backlink***

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 245); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 245); *backlink*

Unknown interpreted text role "c:member".

The tables are declared as three fields of the type object:

```
struct PyMethodDef *tp_methods;
struct PyMemberDef *tp_members;
struct PyGetSetDef *tp_getset;
```

If :c:member:`~PyTypeObject.tp_methods` is not `NULL`, it must refer to an array of :c:type:`PyMethodDef` structures. Each entry in the table is an instance of this structure:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 259); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 259); *backlink*

Unknown interpreted text role "c:type".

```
typedef struct PyMethodDef {
    const char  *ml_name;       /* method name */
    PyCFunction  ml_meth;       /* implementation function */
    int          ml_flags;      /* flags */
    const char  *ml_doc;        /* docstring */
} PyMethodDef;
```

One entry should be defined for each method provided by the type; no entries are needed for methods inherited from a base type. One additional entry is needed at the end; it is a sentinel that marks the end of the array. The :attr:`ml_name` field of the sentinel must be `NULL`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 270); *backlink*

Unknown interpreted text role "attr".

The second table is used to define attributes which map directly to data stored in the instance. A variety of primitive C types are supported, and access may be read-only or read-write. The structures in the table are defined as:

```
typedef struct PyMemberDef {
    const char *name;
    int         type;
    int         offset;
    int         flags;
    const char *doc;
} PyMemberDef;
```

For each entry in the table, a :term:`descriptor` will be constructed and added to the type which will be able to extract a value from the instance structure. The :attr:`type` field should contain one of the type codes defined in the :file:`structmember.h` header; the value will be used to determine how to convert Python values to and from C values. The :attr:`flags` field is used to store flags which control how the attribute can be accessed.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 287); *backlink*

Unknown interpreted text role "term".

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

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 287); backlink

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 287); backlink

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 287); backlink

Unknown interpreted text role "attr".

The following flag constants are defined in :file:`structmember.h`; they may be combined using bitwise-OR.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 294); backlink

Unknown interpreted text role "file".

Constant	Meaning
:const:`READONLY` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 301); backlink Unknown interpreted text role "const".	Never writable.
:const:`PY_AUDIT_READ` System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 303); backlink Unknown interpreted text role "const".	Emit an `object.__getattr__` :ref:`audit events <audit-events>` before reading. System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 303); backlink Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 307)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.10
   :const:`RESTRICTED`, :const:`READ_RESTRICTED` and :const:`WRITE_RESTRICTED`
   are deprecated. However, :const:`READ_RESTRICTED` is an alias for
   :const:`PY_AUDIT_READ`, so fields that specify either :const:`RESTRICTED`
   or :const:`READ_RESTRICTED` will also raise an audit event.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 313)

Unknown directive type "index".

```
.. index::
   single: READONLY
   single: READ_RESTRICTED
   single: WRITE_RESTRICTED
   single: RESTRICTED
   single: PY_AUDIT_READ
```

An interesting advantage of using the :c:member:`~PyTypeObject.tp_members` table to build descriptors that are used at runtime is that any attribute defined this way can have an associated doc string simply by providing the text in the table. An application can use the introspection API to retrieve the descriptor from the class object, and get the doc string using its :attr:`__doc__` attribute.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 320); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 320); *backlink*

Unknown interpreted text role "attr".

As with the :c:member:`~PyTypeObject.tp_methods` table, a sentinel entry with a :attr:`name` value of `NULL` is required.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 326); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 326); *backlink*

Unknown interpreted text role "attr".

Type-specific Attribute Management

For simplicity, only the :c:type:`char\*` version will be demonstrated here; the type of the name parameter is the only difference between the :c:type:`char\*` and :c:type:`PyObject\*` flavors of the interface. This example effectively does the same thing as the generic example above, but does not use the generic support added in Python 2.2. It explains how the handler functions are called, so that if you do need to extend their functionality, you'll understand what needs to be done.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 342); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 342); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 342); *backlink*

Unknown interpreted text role "c:type".

The :c:member:`~PyTypeObject.tp_getattr` handler is called when the object requires an attribute look-up. It is called in the same situations where the :meth:`__getattr__` method of a class would be called.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 350); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 350); *backlink*

Unknown interpreted text role "meth".

Here is an example:

```
static PyObject *
newdatatype_getattr(newdatatypeobject *obj, char *name)
{
    if (strcmp(name, "data") == 0)
    {
        return PyLong_FromLong(obj->data);
    }

    PyErr_Format(PyExc_AttributeError,
                 "'%.50s' object has no attribute '%.400s'",
                 tp->tp_name, name);
    return NULL;
}
```

The :c:member:`~PyTypeObject.tp_setattr` handler is called when the :meth:`__setattr__` or :meth:`__delattr__` method of a class instance would be called. When an attribute should be deleted, the third parameter will be `NULL`. Here is an example that simply raises an exception; if this were really all you wanted, the :c:member:`~PyTypeObject.tp_setattr` handler should be set to `NULL`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 370); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 370); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 370); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 370); *backlink*

Unknown interpreted text role "c:member".

```
static int
newdatatype_setattr(newdatatypeobject *obj, char *name, PyObject *v)
{
    PyErr_Format(PyExc_RuntimeError, "Read-only attribute: %s", name);
    return -1;
}
```

Object Comparison

```
richcmpfunc tp_richcompare;
```

The :c:member:`~PyTypeObject.tp_richcompare` handler is called when comparisons are needed. It is analogous to the :ref:`rich comparison methods <richcmpfuncs>`, like :meth:`__lt__`, and also called by :c:func:`PyObject_RichCompare` and :c:func:`PyObject_RichCompareBool`.

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

Unknown interpreted text role "c:member".

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

Unknown interpreted text role "ref".

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

Unknown interpreted text role "meth".

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

Unknown interpreted text role "c:func".

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

Unknown interpreted text role "c:func".

This function is called with two Python objects and the operator as arguments, where the operator is one of `Py_EQ`, `Py_NE`, `Py_LE`, `Py_GE`, `Py_LT` or `Py_GT`. It should compare the two objects with respect to the specified operator and return `Py_True` or `Py_False` if the comparison is successful, `Py_NotImplemented` to indicate that comparison is not implemented and the other object's comparison method should be tried, or `NULL` if an exception was set.

Here is a sample implementation, for a datatype that is considered equal if the size of an internal pointer is equal:

```
static PyObject *
newdatatype_richcmp(PyObject *obj1, PyObject *obj2, int op)
{
    PyObject *result;
    int c, size1, size2;

    /* code to make sure that both arguments are of type
       newdatatype omitted */

    size1 = obj1->obj_UnderlyingDatatypePtr->size;
    size2 = obj2->obj_UnderlyingDatatypePtr->size;

    switch (op) {
    case Py_LT: c = size1 <  size2; break;
    case Py_LE: c = size1 <= size2; break;
    case Py_EQ: c = size1 == size2; break;
    case Py_NE: c = size1 != size2; break;
    case Py_GT: c = size1 >  size2; break;
    case Py_GE: c = size1 >= size2; break;
    }
    result = c ? Py_True : Py_False;
    Py_INCREF(result);
    return result;
}
```

Abstract Protocol Support

Python supports a variety of *abstract* 'protocols;' the specific interfaces provided to use these interfaces are documented in :ref:`abstract`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 435); *backlink*

Unknown interpreted text role "ref".

A number of these abstract interfaces were defined early in the development of the Python implementation. In particular, the number, mapping, and sequence protocols have been part of Python since the beginning. Other protocols have been added over time. For protocols which depend on several handler routines from the type implementation, the older protocols have been defined as optional blocks of handlers referenced by the type object. For newer protocols there are additional slots in the main type object, with a flag bit being set to indicate that the slots are present and should be checked by the interpreter. (The flag bit does not indicate that the slot values are non-`NULL`. The flag may be set to indicate the presence of a slot, but a slot may still be unfilled.)

```
PyNumberMethods   *tp_as_number;
PySequenceMethods *tp_as_sequence;
PyMappingMethods  *tp_as_mapping;
```

If you wish your object to be able to act like a number, a sequence, or a mapping object, then you place the address of a structure that implements the C type :c:type:`PyNumberMethods`, :c:type:`PySequenceMethods`, or :c:type:`PyMappingMethods`, respectively. It is up to you to fill in this structure with appropriate values. You can find examples of the use of each of these in the

:file:`Objects` directory of the Python source distribution.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 454); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 454); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 454); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 454); *backlink*

Unknown interpreted text role "file".

```
hashfunc tp_hash;
```

This function, if you choose to provide it, should return a hash number for an instance of your data type. Here is a simple example:

```
static Py_hash_t
newdatatype_hash(newdatatypeobject *obj)
{
    Py_hash_t result;
    result = obj->some_size + 32767 * obj->some_number;
    if (result == -1)
       result = -2;
    return result;
}
```

:c:type:`Py_hash_t` is a signed integer type with a platform-varying width. Returning `-1` from :c:member:`~PyTypeObject.tp_hash` indicates an error, which is why you should be careful to avoid returning it when hash computation is successful, as seen above.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 476); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 476); *backlink*

Unknown interpreted text role "c:member".

```
ternaryfunc tp_call;
```

This function is called when an instance of your data type is "called", for example, if `obj1` is an instance of your data type and the Python script contains `obj1('hello')`, the :c:member:`~PyTypeObject.tp_call` handler is invoked.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 485); *backlink*

Unknown interpreted text role "c:member".

This function takes three arguments:

1. *self* is the instance of the data type which is the subject of the call. If the call is `obj1('hello')`, then *self* is `obj1`.
2. *args* is a tuple containing the arguments to the call. You can use :c:func:`PyArg_ParseTuple` to extract the arguments.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 494); *backlink*

Unknown interpreted text role "c:func".

3. *kwds* is a dictionary of keyword arguments that were passed. If this is non-`NULL` and you support keyword arguments, use :c:func:`PyArg_ParseTupleAndKeywords` to extract the arguments. If you do not want to support keyword arguments and this is non-`NULL`, raise a :exc:`TypeError` with a message saying that keyword arguments are not supported.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 497); *backlink***
>
> Unknown interpreted text role "c:func".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 497); *backlink***
>
> Unknown interpreted text role "exc".

Here is a toy `tp_call` implementation:

```
static PyObject *
newdatatype_call(newdatatypeobject *self, PyObject *args, PyObject *kwds)
{
    PyObject *result;
    const char *arg1;
    const char *arg2;
    const char *arg3;

    if (!PyArg_ParseTuple(args, "sss:call", &arg1, &arg2, &arg3)) {
        return NULL;
    }
    result = PyUnicode_FromFormat(
        "Returning -- value: [%d] arg1: [%s] arg2: [%s] arg3: [%s]\n",
        obj->obj_UnderlyingDatatypePtr->size,
        arg1, arg2, arg3);
    return result;
}

/* Iterators */
getiterfunc tp_iter;
iternextfunc tp_iternext;
```

These functions provide support for the iterator protocol. Both handlers take exactly one parameter, the instance for which they are being called, and return a new reference. In the case of an error, they should set an exception and return `NULL`. :c:member:`~PyTypeObject.tp_iter` corresponds to the Python :meth:`__iter__` method, while :c:member:`~PyTypeObject.tp_iternext` corresponds to the Python :meth:`~iterator.__next__` method.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 529); *backlink***
>
> Unknown interpreted text role "c:member".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 529); *backlink***
>
> Unknown interpreted text role "meth".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 529); *backlink***
>
> Unknown interpreted text role "c:member".

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 529); *backlink***
>
> Unknown interpreted text role "meth".

Any :term:`iterable` object must implement the :c:member:`~PyTypeObject.tp_iter` handler, which must return an :term:`iterator` object. Here the same guidelines apply as for Python classes:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 536); *backlink***

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 536); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 536); *backlink*

Unknown interpreted text role "term".

- For collections (such as lists and tuples) which can support multiple independent iterators, a new iterator should be created and returned by each call to :c:member:`~PyTypeObject.tp_iter`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 540); *backlink*

Unknown interpreted text role "c:member".

- Objects which can only be iterated over once (usually due to side effects of iteration, such as file objects) can implement :c:member:`~PyTypeObject.tp_iter` by returning a new reference to themselves -- and should also therefore implement the :c:member:`~PyTypeObject.tp_iternext` handler.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 543); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 543); *backlink*

Unknown interpreted text role "c:member".

Any :term:`iterator` object should implement both :c:member:`~PyTypeObject.tp_iter` and :c:member:`~PyTypeObject.tp_iternext`. An iterator's :c:member:`~PyTypeObject.tp_iter` handler should return a new reference to the iterator. Its :c:member:`~PyTypeObject.tp_iternext` handler should return a new reference to the next object in the iteration, if there is one. If the iteration has reached the end, :c:member:`~PyTypeObject.tp_iternext` may return `NULL` without setting an exception, or it may set :exc:`StopIteration` *in addition* to returning `NULL`; avoiding the exception can yield slightly better performance. If an actual error occurs, :c:member:`~PyTypeObject.tp_iternext` should always set an exception and return `NULL`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 548); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 548); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 548); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 548); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 548); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 548); *backlink*

Unknown interpreted text role "c:member".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 548); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 548); *backlink*

Unknown interpreted text role "c:member".

Weak Reference Support

One of the goals of Python's weak reference implementation is to allow any type to participate in the weak reference mechanism without incurring the overhead on performance-critical objects (such as numbers).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 570)

Unknown directive type "seealso".

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

For an object to be weakly referencable, the extension type must do two things:

1. Include a :c:type:`PyObject\*` field in the C object structure dedicated to the weak reference mechanism. The object's constructor should leave it `NULL` (which is automatic when using the default :c:member:`~PyTypeObject.tp_alloc`).

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 575); *backlink*

   Unknown interpreted text role "c:type".

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 575); *backlink*

   Unknown interpreted text role "c:member".

2. Set the :c:member:`~PyTypeObject.tp_weaklistoffset` type member to the offset of the aforementioned field in the C object structure, so that the interpreter knows how to access and modify that field.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 580); *backlink*

   Unknown interpreted text role "c:member".

Concretely, here is how a trivial object structure would be augmented with the required field:

```
typedef struct {
    PyObject_HEAD
    PyObject *weakreflist;  /* List of weak references */
} TrivialObject;
```

And the corresponding member in the statically-declared type object:

```
static PyTypeObject TrivialType = {
    PyVarObject_HEAD_INIT(NULL, 0)
    /* ... other members omitted for brevity ... */
    .tp_weaklistoffset = offsetof(TrivialObject, weakreflist),
};
```

The only further addition is that `tp_dealloc` needs to clear any weak references (by calling :c:func:`PyObject_ClearWeakRefs`) if the field is non-`NULL`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 600); *backlink*

Unknown interpreted text role "c:func".

```
static void
Trivial_dealloc(TrivialObject *self)
{
    /* Clear weakrefs first before calling any destructors */
    if (self->weakreflist != NULL)
        PyObject_ClearWeakRefs((PyObject *) self);
    /* ... remainder of destruction code omitted for brevity ... */
    Py_TYPE(self)->tp_free((PyObject *) self);
}
```

More Suggestions

In order to learn how to implement any specific method for your new data type, get the :term:`CPython` source code. Go to the :file:`Objects` directory, then search the C source files for `tp_` plus the function you want (for example, `tp_richcompare`). You will find examples of the function you want to implement.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 618); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 618); *backlink*

Unknown interpreted text role "file".

When you need to verify that an object is a concrete instance of the type you are implementing, use the :c:func:`PyObject_TypeCheck` function. A sample of its use might be something like the following:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 624); *backlink*

Unknown interpreted text role "c:func".

```
if (!PyObject_TypeCheck(some_object, &MyType)) {
    PyErr_SetString(PyExc_TypeError, "arg #1 not a mything");
    return NULL;
}
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\extending\[cpython-main][Doc][extending]newtypes.rst, line 633)

Unknown directive type "seealso".

```
.. seealso::
   Download CPython source releases.
      https://www.python.org/downloads/source/

   The CPython project on GitHub, where the CPython source code is developed.
      https://github.com/python/cpython
```