Common Object Structures

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

Unknown directive type "highlight".

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

There are a large number of structures which are used in the definition of object types for Python. This section describes these structures and how they are used.

Base object types and macros

All Python objects ultimately share a small number of fields at the beginning of the object's representation in memory. These are represented by the :c:type:`PyObject` and :c:type:`PyVarObject` types, which are defined, in turn, by the expansions of some macros also used, whether directly or indirectly, in the definition of all other Python objects.

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

Unknown interpreted text role "c:type".

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

Unknown interpreted text role "c:type".

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

Unknown directive type "c:type".

```
.. c:type:: PyObject

   All object types are extensions of this type.  This is a type which
   contains the information Python needs to treat a pointer to an object as an
   object.  In a normal "release" build, it contains only the object's
   reference count and a pointer to the corresponding type object.
   Nothing is actually declared to be a :c:type:`PyObject`, but every pointer
   to a Python object can be cast to a :c:type:`PyObject*`.  Access to the
   members must be done by using the macros :c:macro:`Py_REFCNT` and
   :c:macro:`Py_TYPE`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 35)

Unknown directive type "c:type".

```
.. c:type:: PyVarObject

   This is an extension of :c:type:`PyObject` that adds the :attr:`ob_size`
   field.  This is only used for objects that have some notion of *length*.
   This type does not often appear in the Python/C API.
   Access to the members must be done by using the macros
   :c:macro:`Py_REFCNT`, :c:macro:`Py_TYPE`, and :c:macro:`Py_SIZE`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 44)

Unknown directive type "c:macro".

```
.. c:macro:: PyObject_HEAD

   This is a macro used when declaring new types which represent objects
```

```
   without a varying length.  The PyObject_HEAD macro expands to::

      PyObject ob_base;

   See documentation of :c:type:`PyObject` above.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 54)

Unknown directive type "c:macro".

```
.. c:macro:: PyObject_VAR_HEAD

   This is a macro used when declaring new types which represent objects
   with a length that varies from instance to instance.
   The PyObject_VAR_HEAD macro expands to::

      PyVarObject ob_base;

   See documentation of :c:type:`PyVarObject` above.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 65)

Unknown directive type "c:function".

```
.. c:function:: int Py_Is(const PyObject *x, const PyObject *y)

   Test if the *x* object is the *y* object, the same as ``x is y`` in Python.

   .. versionadded:: 3.10
```

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

Unknown directive type "c:function".

```
.. c:function:: int Py_IsNone(const PyObject *x)

   Test if an object is the ``None`` singleton,
   the same as ``x is None`` in Python.

   .. versionadded:: 3.10
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 80)

Unknown directive type "c:function".

```
.. c:function:: int Py_IsTrue(const PyObject *x)

   Test if an object is the ``True`` singleton,
   the same as ``x is True`` in Python.

   .. versionadded:: 3.10
```

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

Unknown directive type "c:function".

```
.. c:function:: int Py_IsFalse(const PyObject *x)

   Test if an object is the ``False`` singleton,
   the same as ``x is False`` in Python.
```

```
   .. versionadded:: 3.10
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 96)

Unknown directive type "c:function".

```
.. c:function:: PyTypeObject* Py_TYPE(const PyObject *o)

   Get the type of the Python object *o*.

   Return a :term:`borrowed reference`.

   Use the :c:func:`Py_SET_TYPE` function to set an object type.

   .. versionchanged:: 3.11
      :c:func:`Py_TYPE()` is changed to an inline static function.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 108)

Unknown directive type "c:function".

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

   Return non-zero if the object *o* type is *type*. Return zero otherwise.
   Equivalent to: ``Py_TYPE(o) == type``.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 116)

Unknown directive type "c:function".

```
.. c:function:: void Py_SET_TYPE(PyObject *o, PyTypeObject *type)

   Set the object *o* type to *type*.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 123)

Unknown directive type "c:function".

```
.. c:function:: Py_ssize_t Py_REFCNT(const PyObject *o)

   Get the reference count of the Python object *o*.

   Use the :c:func:`Py_SET_REFCNT()` function to set an object reference count.

   .. versionchanged:: 3.10
      :c:func:`Py_REFCNT()` is changed to the inline static function.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 133)

Unknown directive type "c:function".

```
.. c:function:: void Py_SET_REFCNT(PyObject *o, Py_ssize_t refcnt)

   Set the object *o* reference counter to *refcnt*.

   .. versionadded:: 3.9
```

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

Unknown directive type "c:function".

```
.. c:function:: Py_ssize_t Py_SIZE(const PyVarObject *o)

   Get the size of the Python object *o*.

   Use the :c:func:`Py_SET_SIZE` function to set an object size.

   .. versionchanged:: 3.11
      :c:func:`Py_SIZE()` is changed to an inline static function.
```

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

Unknown directive type "c:function".

```
.. c:function:: void Py_SET_SIZE(PyVarObject *o, Py_ssize_t size)

   Set the object *o* size to *size*.

   .. versionadded:: 3.9
```

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

Unknown directive type "c:macro".

```
.. c:macro:: PyObject_HEAD_INIT(type)

   This is a macro which expands to initialization values for a new
   :c:type:`PyObject` type.  This macro expands to::

      _PyObject_EXTRA_INIT
      1, type,
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 166)

Unknown directive type "c:macro".

```
.. c:macro:: PyVarObject_HEAD_INIT(type, size)

   This is a macro which expands to initialization values for a new
   :c:type:`PyVarObject` type, including the :attr:`ob_size` field.
   This macro expands to::

      _PyObject_EXTRA_INIT
      1, type, size,
```

Implementing functions and methods

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 179)

Unknown directive type "c:type".

```
.. c:type:: PyCFunction

   Type of the functions used to implement most Python callables in C.
   Functions of this type take two :c:type:`PyObject*` parameters and return
   one such value.  If the return value is ``NULL``, an exception shall have
   been set.  If not ``NULL``, the return value is interpreted as the return
   value of the function as exposed in Python.  The function must return a new
```

```
   reference.

   The function signature is::

      PyObject *PyCFunction(PyObject *self,
                            PyObject *args);
```

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

Unknown directive type "c:type".

```
.. c:type:: PyCFunctionWithKeywords

   Type of the functions used to implement Python callables in C
   with signature :const:`METH_VARARGS | METH_KEYWORDS`.
   The function signature is::

      PyObject *PyCFunctionWithKeywords(PyObject *self,
                                        PyObject *args,
                                        PyObject *kwargs);
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 204)

Unknown directive type "c:type".

```
.. c:type:: _PyCFunctionFast

   Type of the functions used to implement Python callables in C
   with signature :const:`METH_FASTCALL`.
   The function signature is::

      PyObject *_PyCFunctionFast(PyObject *self,
                                 PyObject *const *args,
                                 Py_ssize_t nargs);
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 214)

Unknown directive type "c:type".

```
.. c:type:: _PyCFunctionFastWithKeywords

   Type of the functions used to implement Python callables in C
   with signature :const:`METH_FASTCALL | METH_KEYWORDS`.
   The function signature is::

      PyObject *_PyCFunctionFastWithKeywords(PyObject *self,
                                             PyObject *const *args,
                                             Py_ssize_t nargs,
                                             PyObject *kwnames);
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 225)

Unknown directive type "c:type".

```
.. c:type:: PyCMethod

   Type of the functions used to implement Python callables in C
   with signature :const:`METH_METHOD | METH_FASTCALL | METH_KEYWORDS`.
   The function signature is::

      PyObject *PyCMethod(PyObject *self,
                          PyTypeObject *defining_class,
                          PyObject *const *args,
                          Py_ssize_t nargs,
                          PyObject *kwnames)

   .. versionadded:: 3.9
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst`, line 240)

Unknown directive type "c:type".

```
.. c:type:: PyMethodDef

   Structure used to describe a method of an extension type.  This structure has
   four fields:

   +------------------+---------------+-------------------------------+
   | Field            | C Type        | Meaning                       |
   +==================+===============+===============================+
   | :attr:`ml_name`  | const char \* | name of the method            |
   +------------------+---------------+-------------------------------+
   | :attr:`ml_meth`  | PyCFunction   | pointer to the C              |
   |                  |               | implementation                |
   +------------------+---------------+-------------------------------+
   | :attr:`ml_flags` | int           | flag bits indicating how the  |
   |                  |               | call should be constructed    |
   +------------------+---------------+-------------------------------+
   | :attr:`ml_doc`   | const char \* | points to the contents of the |
   |                  |               | docstring                     |
   +------------------+---------------+-------------------------------+
```

The :attr:`ml_meth` is a C function pointer. The functions may be of different types, but they always return :c:type:`PyObject*`. If the function is not of the :c:type:`PyCFunction`, the compiler will require a cast in the method table. Even though :c:type:`PyCFunction` defines the first parameter as :c:type:`PyObject*`, it is common that the method implementation uses the specific C type of the *self* object.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst`, line 260); ***backlink***

Unknown interpreted text role "attr".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst`, line 260); ***backlink***

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst`, line 260); ***backlink***

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst`, line 260); ***backlink***

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst`, line 260); ***backlink***

Unknown interpreted text role "c:type".

The :attr:`ml_flags` field is a bitfield which can include the following flags. The individual flags indicate either a calling convention or a binding convention.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst`, line 267); ***backlink***

Unknown interpreted text role "attr".

There are these calling conventions:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst`, line 273)

Unknown directive type "data".

```
.. data:: METH_VARARGS

   This is the typical calling convention, where the methods have the type
   :c:type:`PyCFunction`. The function expects two :c:type:`PyObject*` values.
   The first one is the *self* object for methods; for module functions, it is
   the module object.  The second parameter (often called *args*) is a tuple
   object representing all arguments. This parameter is typically processed
   using :c:func:`PyArg_ParseTuple` or :c:func:`PyArg_UnpackTuple`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 283)

Unknown directive type "data".

```
.. data:: METH_VARARGS | METH_KEYWORDS

   Methods with these flags must be of type :c:type:`PyCFunctionWithKeywords`.
   The function expects three parameters: *self*, *args*, *kwargs* where
   *kwargs* is a dictionary of all the keyword arguments or possibly ``NULL``
   if there are no keyword arguments.  The parameters are typically processed
   using :c:func:`PyArg_ParseTupleAndKeywords`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 292)

Unknown directive type "data".

```
.. data:: METH_FASTCALL

   Fast calling convention supporting only positional arguments.
   The methods have the type :c:type:`_PyCFunctionFast`.
   The first parameter is *self*, the second parameter is a C array
   of :c:type:`PyObject*` values indicating the arguments and the third
   parameter is the number of arguments (the length of the array).

   .. versionadded:: 3.7

   .. versionchanged:: 3.10

      ``METH_FASTCALL`` is now part of the stable ABI.
```

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

Unknown directive type "data".

```
.. data:: METH_FASTCALL | METH_KEYWORDS

   Extension of :const:`METH_FASTCALL` supporting also keyword arguments,
   with methods of type :c:type:`_PyCFunctionFastWithKeywords`.
   Keyword arguments are passed the same way as in the
   :ref:`vectorcall protocol <vectorcall>`:
   there is an additional fourth :c:type:`PyObject*` parameter
   which is a tuple representing the names of the keyword arguments
   (which are guaranteed to be strings)
   or possibly ``NULL`` if there are no keywords.  The values of the keyword
   arguments are stored in the *args* array, after the positional arguments.

   This is not part of the :ref:`limited API <stable>`.

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 324)

Unknown directive type "data".

```
.. data:: METH_METHOD | METH_FASTCALL | METH_KEYWORDS

   Extension of :const:`METH_FASTCALL | METH_KEYWORDS` supporting the *defining
   class*, that is, the class that contains the method in question.
   The defining class might be a superclass of ``Py_TYPE(self)``.

   The method needs to be of type :c:type:`PyCMethod`, the same as for
   ``METH_FASTCALL | METH_KEYWORDS`` with ``defining_class`` argument added after
   ``self``.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 337)

Unknown directive type "data".

```
.. data:: METH_NOARGS

   Methods without parameters don't need to check whether arguments are given if
   they are listed with the :const:`METH_NOARGS` flag.  They need to be of type
   :c:type:`PyCFunction`.  The first parameter is typically named *self* and will
   hold a reference to the module or object instance.  In all cases the second
   parameter will be ``NULL``.

   The function must have 2 parameters. Since the second parameter is unused,
   :c:macro:`Py_UNUSED` can be used to prevent a compiler warning.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 349)

Unknown directive type "data".

```
.. data:: METH_O

   Methods with a single object argument can be listed with the :const:`METH_O`
   flag, instead of invoking :c:func:`PyArg_ParseTuple` with a ``"O"`` argument.
   They have the type :c:type:`PyCFunction`, with the *self* parameter, and a
   :c:type:`PyObject*` parameter representing the single argument.
```

These two constants are not used to indicate the calling convention but the binding when use with methods of classes. These may not be used for functions defined for modules. At most one of these flags may be set for any given method.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 363)

Unknown directive type "data".

```
.. data:: METH_CLASS

   .. index:: builtin: classmethod

   The method will be passed the type object as the first parameter rather
   than an instance of the type.  This is used to create *class methods*,
   similar to what is created when using the :func:`classmethod` built-in
   function.
```

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

Unknown directive type "data".

```
.. data:: METH_STATIC

   .. index:: builtin: staticmethod

   The method will be passed ``NULL`` as the first parameter rather than an
   instance of the type.  This is used to create *static methods*, similar to
   what is created when using the :func:`staticmethod` built-in function.
```

One other constant controls whether a method is loaded in place of another definition with the same method name.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 385)

Unknown directive type "data".

```
.. data:: METH_COEXIST

   The method will be loaded in place of existing definitions.  Without
   *METH_COEXIST*, the default is to skip repeated definitions.  Since slot
   wrappers are loaded before the method table, the existence of a
   *sq_contains* slot, for example, would generate a wrapped method named
   :meth:`__contains__` and preclude the loading of a corresponding
   PyCFunction with the same name.  With the flag defined, the PyCFunction
   will be loaded in place of the wrapper object and will co-exist with the
   slot.  This is helpful because calls to PyCFunctions are optimized more
   than wrapper object calls.
```

Accessing attributes of extension types

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 401)

Unknown directive type "c:type".

```
.. c:type:: PyMemberDef

   Structure which describes an attribute of a type which corresponds to a C
   struct member.  Its fields are:

   +------------------+---------------+------------------------------+
   | Field            | C Type        | Meaning                      |
   +==================+===============+==============================+
   | :attr:`name`     | const char \* | name of the member           |
   +------------------+---------------+------------------------------+
   | :attr:`!type`    | int           | the type of the member in the |
   |                  |               | C struct                     |
   +------------------+---------------+------------------------------+
   | :attr:`offset`   | Py_ssize_t    | the offset in bytes that the  |
   |                  |               | member is located on the     |
   |                  |               | type's object struct         |
   +------------------+---------------+------------------------------+
   | :attr:`flags`    | int           | flag bits indicating if the   |
   |                  |               | field should be read-only or  |
   |                  |               | writable                     |
   +------------------+---------------+------------------------------+
   | :attr:`doc`      | const char \* | points to the contents of the |
   |                  |               | docstring                    |
   +------------------+---------------+------------------------------+

   :attr:`!type` can be one of many ``T_`` macros corresponding to various C
   types.  When the member is accessed in Python, it will be converted to the
   equivalent Python type.

   =============== ==================
   Macro name      C type
   =============== ==================
   T_SHORT         short
   T_INT           int
   T_LONG          long
   T_FLOAT         float
   T_DOUBLE        double
   T_STRING        const char \*
   T_OBJECT        PyObject \*
   T_OBJECT_EX     PyObject \*
   T_CHAR          char
   T_BYTE          char
   T_UBYTE         unsigned char
   T_UINT          unsigned int
   T_USHORT        unsigned short
   T_ULONG         unsigned long
   T_BOOL          char
   T_LONGLONG      long long
```

```
   T_ULONGLONG     unsigned long long
   T_PYSSIZET      Py_ssize_t
   =============== ==================

   :c:macro:`T_OBJECT` and :c:macro:`T_OBJECT_EX` differ in that
   :c:macro:`T_OBJECT` returns ``None`` if the member is ``NULL`` and
   :c:macro:`T_OBJECT_EX` raises an :exc:`AttributeError`.  Try to use
   :c:macro:`T_OBJECT_EX` over :c:macro:`T_OBJECT` because :c:macro:`T_OBJECT_EX`
   handles use of the :keyword:`del` statement on that attribute more correctly
   than :c:macro:`T_OBJECT`.

   :attr:`flags` can be ``0`` for write and read access or :c:macro:`READONLY` for
   read-only access.  Using :c:macro:`T_STRING` for :attr:`type` implies
   :c:macro:`READONLY`.  :c:macro:`T_STRING` data is interpreted as UTF-8.
   Only :c:macro:`T_OBJECT` and :c:macro:`T_OBJECT_EX`
   members can be deleted.  (They are set to ``NULL``).

   .. _pymemberdef-offsets:

   Heap allocated types (created using :c:func:`PyType_FromSpec` or similar),
   ``PyMemberDef`` may contain definitions for the special members
   ``__dictoffset__``, ``__weaklistoffset__`` and ``__vectorcalloffset__``,
   corresponding to
   :c:member:`~PyTypeObject.tp_dictoffset`,
   :c:member:`~PyTypeObject.tp_weaklistoffset` and
   :c:member:`~PyTypeObject.tp_vectorcall_offset` in type objects.
   These must be defined with ``T_PYSSIZET`` and ``READONLY``, for example::

      static PyMemberDef spam_type_members[] = {
          {"__dictoffset__", T_PYSSIZET, offsetof(Spam_object, dict), READONLY},
          {NULL}  /* Sentinel */
      };
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 483)

Unknown directive type "c:function".

```
   .. c:function:: PyObject* PyMember_GetOne(const char *obj_addr, struct PyMemberDef *m)

      Get an attribute belonging to the object at address *obj_addr*.  The
      attribute is described by ``PyMemberDef`` *m*.  Returns ``NULL``
      on error.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 490)

Unknown directive type "c:function".

```
   .. c:function:: int PyMember_SetOne(char *obj_addr, struct PyMemberDef *m, PyObject *o)

      Set an attribute belonging to the object at address *obj_addr* to object *o*.
      The attribute to set is described by ``PyMemberDef`` *m*.  Returns ``0``
      if successful and a negative value on failure.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]structures.rst, line 497)

Unknown directive type "c:type".

```
   .. c:type:: PyGetSetDef

      Structure to define property-like access for a type. See also description of
      the :c:member:`PyTypeObject.tp_getset` slot.

      +-------------+------------------+-----------------------------------+
      | Field       | C Type           | Meaning                           |
      +=============+==================+===================================+
      | name        | const char \*    | attribute name                    |
      +-------------+------------------+-----------------------------------+
      | get         | getter           | C function to get the attribute   |
      +-------------+------------------+-----------------------------------+
```

```
| set         | setter           | optional C function to set or     |
|             |                  | delete the attribute, if omitted  |
|             |                  | the attribute is readonly         |
+-------------+------------------+-----------------------------------+
| doc         | const char \*    | optional docstring                |
+-------------+------------------+-----------------------------------+
| closure     | void \*          | optional function pointer,        |
|             |                  | providing additional data for     |
|             |                  | getter and setter                 |
+-------------+------------------+-----------------------------------+
```

The ``get`` function takes one :c:type:`PyObject*` parameter (the instance) and a function pointer (the associated ``closure``)::

```
   typedef PyObject *(*getter)(PyObject *, void *);
```

It should return a new reference on success or ``NULL`` with a set exception on failure.

``set`` functions take two :c:type:`PyObject*` parameters (the instance and the value to be set) and a function pointer (the associated ``closure``)::

```
   typedef int (*setter)(PyObject *, PyObject *, void *);
```

In case the attribute should be deleted the second parameter is ``NULL``. Should return ``0`` on success or ``-1`` with a set exception on failure.