

:mod:`ctypes` --- A foreign function library for Python

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 4)

Unknown directive type "module".

```
.. module:: ctypes
   :synopsis: A foreign function library for Python.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 7)

Unknown directive type "moduleauthor".

```
.. moduleauthor:: Thomas Heller <theller@python.net>
```

:mod:`ctypes` is a foreign function library for Python. It provides C compatible data types, and allows calling functions in DLLs or shared libraries. It can be used to wrap these libraries in pure Python.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 11); [backlink](#)

Unknown interpreted text role "mod".

ctypes tutorial

Note: The code samples in this tutorial use :mod:`doctest` to make sure that they actually work. Since some code samples behave differently under Linux, Windows, or macOS, they contain doctest directives in comments.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 21); [backlink](#)

Unknown interpreted text role "mod".

Note: Some code samples reference the ctypes :class:`c_int` type. On platforms where `sizeof(long) == sizeof(int)` it is an alias to :class:`c_long`. So, you should not be confused if :class:`c_long` is printed if you would expect :class:`c_int` --- they are actually the same type.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 25); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 25); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 25); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 25); [backlink](#)

Unknown interpreted text role "class".

Loading dynamic link libraries

`:mod:'ctypes'` exports the `cdll`, and on Windows `windll` and `oledll` objects, for loading dynamic link libraries.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 35); [backlink](#)

Unknown interpreted text role "mod".

You load libraries by accessing them as attributes of these objects. `cdll` loads libraries which export functions using the standard `cdecl` calling convention, while `windll` libraries call functions using the `stdcall` calling convention. `oledll` also uses the `stdcall` calling convention, and assumes the functions return a Windows `:type:'HRESULT'` error code. The error code is used to automatically raise an `:class:'OSError'` exception when the function call fails.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 38); [backlink](#)

Unknown interpreted text role "c.type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 38); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 46)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.3
   Windows errors used to raise :exc:`WindowsError`, which is now an alias
   of :exc:`OSError`.
```

Here are some examples for Windows. Note that `msvcrt` is the MS standard C library containing most standard C functions, and uses the `cdecl` calling convention:

```
>>> from ctypes import *
>>> print(windll.kernel32) # doctest: +WINDOWS
<WinDLL 'kernel32', handle ... at ...>
>>> print(cdll.msvcrt) # doctest: +WINDOWS
<CDLL 'msvcrt', handle ... at ...>
>>> libc = cdll.msvcrt # doctest: +WINDOWS
>>>
```

Windows appends the usual `.dll` file suffix automatically.

Note

Accessing the standard C library through `cdll.msvcrt` will use an outdated version of the library that may be incompatible with the one being used by Python. Where possible, use native Python functionality, or else import and use the `msvcrt` module.

On Linux, it is required to specify the filename *including* the extension to load a library, so attribute access can not be used to load libraries. Either the `:meth:'LoadLibrary'` method of the dll loaders should be used, or you should load the library by creating an instance of `CDLL` by calling the constructor:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 71); [backlink](#)

Unknown interpreted text role "meth".

```
>>> cdll.LoadLibrary("libc.so.6") # doctest: +LINUX
<CDLL 'libc.so.6', handle ... at ...>
>>> libc = CDLL("libc.so.6") # doctest: +LINUX
>>> libc # doctest: +LINUX
<CDLL 'libc.so.6', handle ... at ...>
>>>
```

Accessing functions from loaded dlls

Functions are accessed as attributes of dll objects:

```
>>> from ctypes import *
>>> libc.printf
<_FuncPtr object at 0x...>
>>> print(windll.kernel32.GetModuleHandleA) # doctest: +WINDOWS
<_FuncPtr object at 0x...>
>>> print(windll.kernel32.MyOwnFunction)      # doctest: +WINDOWS
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "ctypes.py", line 239, in __getattr__
    func = _StdcallFuncPtr(name, self)
AttributeError: function 'MyOwnFunction' not found
>>>
```

Note that win32 system dlls like `kernel32` and `user32` often export ANSI as well as UNICODE versions of a function. The UNICODE version is exported with an `W` appended to the name, while the ANSI version is exported with an `A` appended to the name. The win32 `GetModuleHandle` function, which returns a *module handle* for a given module name, has the following C prototype, and a macro is used to expose one of them as `GetModuleHandle` depending on whether UNICODE is defined or not:

```
/* ANSI version */
HMODULE GetModuleHandleA(LPCSTR lpModuleName);
/* UNICODE version */
HMODULE GetModuleHandleW(LPCWSTR lpModuleName);
```

`windll` does not try to select one of them by magic, you must access the version you need by specifying `GetModuleHandleA` or `GetModuleHandleW` explicitly, and then call it with bytes or string objects respectively.

Sometimes, dlls export functions with names which aren't valid Python identifiers, like `"???@YAPAXI@Z"`. In this case you have to use `:func:'getattr'` to retrieve the function:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 123); [backlink](#)

Unknown interpreted text role "func".

```
>>> getattr(cdll.msvcrt, "???@YAPAXI@Z") # doctest: +WINDOWS
<_FuncPtr object at 0x...>
>>>
```

On Windows, some dlls export functions not by name but by ordinal. These functions can be accessed by indexing the dll object with the ordinal number:

```
>>> cdll.kernel32[1] # doctest: +WINDOWS
<_FuncPtr object at 0x...>
>>> cdll.kernel32[0] # doctest: +WINDOWS
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "ctypes.py", line 310, in __getitem__
    func = _StdcallFuncPtr(name, self)
AttributeError: function ordinal 0 not found
>>>
```

Calling functions

You can call these functions like any other Python callable. This example uses the `time()` function, which returns system time in seconds since the Unix epoch, and the `GetModuleHandleA()` function, which returns a win32 module handle.

This example calls both functions with a `NULL` pointer (`None` should be used as the `NULL` pointer):

```
>>> print(libc.time(None)) # doctest: +SKIP
1150640792
>>> print(hex(windll.kernel32.GetModuleHandleA(None))) # doctest: +WINDOWS
0x1d000000
>>>
```

`:exc:'ValueError'` is raised when you call an `stdcall` function with the `cdecl` calling convention, or vice versa:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 164); [backlink](#)

Unknown interpreted text role "exc".

```
>>> cdll.kernel32.GetModuleHandleA(None) # doctest: +WINDOWS
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
```

```

ValueError: Procedure probably called with not enough arguments (4 bytes missing)
>>>

>>> windll.msvcrt.printf(b"spam") # doctest: +WINDOWS
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: Procedure probably called with too many arguments (4 bytes in excess)
>>>

```

To find out the correct calling convention you have to look into the C header file or the documentation for the function you want to call.

On Windows, `mod:ctypes` uses win32 structured exception handling to prevent crashes from general protection faults when functions are called with invalid argument values:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 182); [backlink](#)

Unknown interpreted text role "mod".

```

>>> windll.kernel32.GetModuleHandleA(32) # doctest: +WINDOWS
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
OSError: exception: access violation reading 0x00000020
>>>

```

There are, however, enough ways to crash Python with `mod:ctypes`, so you should be careful anyway. The `mod:faulthandler` module can be helpful in debugging crashes (e.g. from segmentation faults produced by erroneous C library calls).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 192); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 192); [backlink](#)

Unknown interpreted text role "mod".

`None`, integers, bytes objects and (unicode) strings are the only native Python objects that can directly be used as parameters in these function calls. `None` is passed as a C `NULL` pointer, bytes objects and strings are passed as pointer to the memory block that contains their data (`c:type:char *` or `c:type:wchar_t *`). Python integers are passed as the platform's default C `c:type:int` type, their value is masked to fit into the C type.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 197); [backlink](#)

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 197); [backlink](#)

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 197); [backlink](#)

Unknown interpreted text role "c:type".

Before we move on calling functions with other parameter types, we have to learn more about `mod:ctypes` data types.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 204); [backlink](#)

Unknown interpreted text role "mod".

Fundamental data types

`mod:ctypes` defines a number of primitive C compatible data types:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 213); [backlink](#)

Unknown interpreted text role "mod".

ctypes type	C type	Python type
<p><code>:class:`c_bool`</code></p> <div><p>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 219); backlink</p><p>Unknown interpreted text role "class".</p></div>	<p><code>:ctype:`_Bool`</code></p> <div><p>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 219); backlink</p><p>Unknown interpreted text role "ctype".</p></div>	<p>bool(1)</p>
<p><code>:class:`c_char`</code></p> <div><p>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 221); backlink</p><p>Unknown interpreted text role "class".</p></div>	<p><code>:ctype:`char`</code></p> <div><p>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 221); backlink</p><p>Unknown interpreted text role "ctype".</p></div>	<p>1-character bytes object</p>
<p><code>:class:`c_wchar`</code></p> <div><p>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 223); backlink</p><p>Unknown interpreted text role "class".</p></div>	<p><code>:ctype:`wchar_t`</code></p> <div><p>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 223); backlink</p><p>Unknown interpreted text role "ctype".</p></div>	<p>1-character string</p>

ctypes type	C type	Python type
:~class:~c_byte~ <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 225); backlink Unknown interpreted text role "class". </div>	:~type:~char~ <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 225); backlink Unknown interpreted text role "c.type". </div>	int
:~class:~c_ubyte~ <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 227); backlink Unknown interpreted text role "class". </div>	:~type:~unsigned char~ <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 227); backlink Unknown interpreted text role "c.type". </div>	int
:~class:~c_short~ <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 229); backlink Unknown interpreted text role "class". </div>	:~type:~short~ <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 229); backlink Unknown interpreted text role "c.type". </div>	int
:~class:~c_ushort~ <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 231); backlink Unknown interpreted text role "class". </div>	:~type:~unsigned short~ <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 231); backlink Unknown interpreted text role "c.type". </div>	int

ctypes type	C type	Python type
:code:`class: 'c_int'` <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 233); backlink Unknown interpreted text role "class". </div>	:code:`:c.type: 'int'` <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 233); backlink Unknown interpreted text role "c.type". </div>	int
:code:`class: 'c_uint'` <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 235); backlink Unknown interpreted text role "class". </div>	:code:`:c.type: 'unsigned int'` <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 235); backlink Unknown interpreted text role "c.type". </div>	int
:code:`class: 'c_long'` <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 237); backlink Unknown interpreted text role "class". </div>	:code:`:c.type: 'long'` <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 237); backlink Unknown interpreted text role "c.type". </div>	int
:code:`class: 'c_ulong'` <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 239); backlink Unknown interpreted text role "class". </div>	:code:`:c.type: 'unsigned long'` <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 239); backlink Unknown interpreted text role "c.type". </div>	int

ctypes type	C type	Python type
<p><code>:class:`c_longlong`</code></p> <div data-bbox="220 320 521 741"> <p>System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 241); backlink</p> <p>Unknown interpreted text role "class".</p> </div>	<p><code>:ctype:`__int64`</code> or <code>:ctype:`long long`</code></p> <div data-bbox="639 168 1062 524"> <p>System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 241); backlink</p> <p>Unknown interpreted text role "ctype".</p> </div> <div data-bbox="639 551 1062 907"> <p>System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 241); backlink</p> <p>Unknown interpreted text role "ctype".</p> </div>	<p>int</p>
<p><code>:class:`c_ulonglong`</code></p> <div data-bbox="220 1178 521 1599"> <p>System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 243); backlink</p> <p>Unknown interpreted text role "class".</p> </div>	<p><code>:ctype:`unsigned __int64`</code> or <code>:ctype:`unsigned long long`</code></p> <div data-bbox="639 1037 1062 1393"> <p>System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 243); backlink</p> <p>Unknown interpreted text role "ctype".</p> </div> <div data-bbox="639 1420 1062 1776"> <p>System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 243); backlink</p> <p>Unknown interpreted text role "ctype".</p> </div>	<p>int</p>

ctypes type	C type	Python type
: class: 'c_size_t' <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 246); backlink Unknown interpreted text role "class". </div>	: c.type: 'size_t' <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 246); backlink Unknown interpreted text role "c.type". </div>	int
: class: 'c_ssize_t' <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 248); backlink Unknown interpreted text role "class". </div>	: c.type: 'ssize_t' or : c.type: 'Py_ssize_t' <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 248); backlink Unknown interpreted text role "c.type". </div> <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 248); backlink Unknown interpreted text role "c.type". </div>	int
: class: 'c_float' <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 251); backlink Unknown interpreted text role "class". </div>	: c.type: 'float' <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 251); backlink Unknown interpreted text role "c.type". </div>	float

ctypes type	C type	Python type
: class: 'c_double' <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 253); backlink Unknown interpreted text role "class". </div>	: c.type: 'double' <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 253); backlink Unknown interpreted text role "c.type". </div>	float
: class: 'c_longdouble' <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 255); backlink Unknown interpreted text role "class". </div>	: c.type: 'long double' <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 255); backlink Unknown interpreted text role "c.type". </div>	float
: class: 'c_char_p' <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 257); backlink Unknown interpreted text role "class". </div>	: c.type: 'char *' (NUL terminated) <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 257); backlink Unknown interpreted text role "c.type". </div>	bytes object or None
: class: 'c_wchar_p' <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 259); backlink Unknown interpreted text role "class". </div>	: c.type: 'wchar_t *' (NUL terminated) <div> System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 259); backlink Unknown interpreted text role "c.type". </div>	string or None

ctypes type	C type	Python type
<pre>:class:'c_void_p'</pre> <div> <p>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 261); backlink</p> <p>Unknown interpreted text role "class".</p> </div>	<pre>:c.type:'void *'</pre> <div> <p>System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 261); backlink</p> <p>Unknown interpreted text role "c.type".</p> </div>	<pre>int or None</pre>

1. The constructor accepts any object with a truth value.

All these types can be created by calling them with an optional initializer of the correct type and value:

```
>>> c_int()
c_long(0)
>>> c_wchar_p("Hello, World")
c_wchar_p(140018365411392)
>>> c_ushort(-3)
c_ushort(65533)
>>>
```

Since these types are mutable, their value can also be changed afterwards:

```
>>> i = c_int(42)
>>> print(i)
c_long(42)
>>> print(i.value)
42
>>> i.value = -99
>>> print(i.value)
-99
>>>
```

Assigning a new value to instances of the pointer types `:class:'c_char_p'`, `:class:'c_wchar_p'`, and `:class:'c_void_p'` changes the *memory location* they point to, *not the contents* of the memory block (of course not, because Python bytes objects are immutable):

System Message: **ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 289); [backlink](#)

Unknown interpreted text role "class".

System Message: **ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 289); [backlink](#)

Unknown interpreted text role "class".

System Message: **ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 289); [backlink](#)

Unknown interpreted text role "class".

```
>>> s = "Hello, World"
>>> c_s = c_wchar_p(s)
>>> print(c_s)
c_wchar_p(139966785747344)
>>> print(c_s.value)
Hello World
>>> c_s.value = "Hi, there"
>>> print(c_s) # the memory location has changed
c_wchar_p(139966783348904)
>>> print(c_s.value)
Hi, there
>>> print(s) # first object is unchanged
Hello, World
>>>
```

You should be careful, however, not to pass them to functions expecting pointers to mutable memory. If you need mutable memory blocks, ctypes has a `:func:'create_string_buffer'` function which creates these in various ways. The current memory block contents can be accessed (or changed) with the `raw` property; if you want to access it as NUL terminated string, use the `value` property:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 309); [backlink](#)

Unknown interpreted text role "func".

```
>>> from ctypes import *
>>> p = create_string_buffer(3)           # create a 3 byte buffer, initialized to NUL bytes
>>> print(sizeof(p), repr(p.raw))
3 b'\x00\x00\x00'
>>> p = create_string_buffer(b"Hello")   # create a buffer containing a NUL terminated string
>>> print(sizeof(p), repr(p.raw))
6 b'Hello\x00'
>>> print(repr(p.value))
b'Hello'
>>> p = create_string_buffer(b"Hello", 10) # create a 10 byte buffer
>>> print(sizeof(p), repr(p.raw))
10 b'Hello\x00\x00\x00\x00\x00'
>>> p.value = b"Hi"
>>> print(sizeof(p), repr(p.raw))
10 b'Hi\x00lo\x00\x00\x00\x00'
>>>
```

The `:func:'create_string_buffer'` function replaces the old `:func:'c_buffer'` function (which is still available as an alias). To create a mutable memory block containing unicode characters of the C type `:ctype:'wchar_t'`, use the `:func:'create_unicode_buffer'` function.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 333); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 333); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 333); [backlink](#)

Unknown interpreted text role "ctype".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 333); [backlink](#)

Unknown interpreted text role "func".

Calling functions, continued

Note that `printf` prints to the real standard output channel, *not* to `:data:'sys.stdout'`, so these examples will only work at the console prompt, not from within *IDLE* or *PythonWin*:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 344); [backlink](#)

Unknown interpreted text role "data".

```
>>> printf = libc.printf
>>> printf(b"Hello, %s\n", b"World!")
Hello, World!
14
>>> printf(b"Hello, %S\n", "World!")
Hello, World!
14
>>> printf(b"%d bottles of beer\n", 42)
42 bottles of beer
19
>>> printf(b"%f bottles of beer\n", 42.5)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
```

```
ArgumentError: argument 2: exceptions.TypeError: Don't know how to convert parameter 2
>>>
```

As has been mentioned before, all Python types except integers, strings, and bytes objects have to be wrapped in their corresponding `:mod:`ctypes`` type, so that they can be converted to the required C data type:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 364); [backlink](#)

Unknown interpreted text role "mod".

```
>>> printf(b"An int %d, a double %f\n", 1234, c_double(3.14))
An int 1234, a double 3.140000
31
>>>
```

Calling functions with your own custom data types

You can also customize `:mod:`ctypes`` argument conversion to allow instances of your own classes be used as function arguments. `:mod:`ctypes`` looks for an `:attr:`_as_parameter_`` attribute and uses this as the function argument. Of course, it must be one of integer, string, or bytes:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 379); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 379); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 379); [backlink](#)

Unknown interpreted text role "attr".

```
>>> class Bottles:
...     def __init__(self, number):
...         self._as_parameter_ = number
...
>>> bottles = Bottles(42)
>>> printf(b"%d bottles of beer\n", bottles)
42 bottles of beer
19
>>>
```

If you don't want to store the instance's data in the `:attr:`_as_parameter_`` instance variable, you could define a `:class:`property`` which makes the attribute available on request.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 394); [backlink](#)

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 394); [backlink](#)

Unknown interpreted text role "class".

Specifying the required argument types (function prototypes)

It is possible to specify the required argument types of functions exported from DLLs by setting the `:attr:`argtypes`` attribute.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 404); [backlink](#)

Unknown interpreted text role "attr".

`:attr:`argtypes`` must be a sequence of C data types (the `printf` function is probably not a good example here, because it takes a

variable number and different types of parameters depending on the format string, on the other hand this is quite handy to experiment with this feature):

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 407); [backlink](#)

Unknown interpreted text role "attr".

```
>>> printf.argtypes = [c_char_p, c_char_p, c_int, c_double]
>>> printf(b"String '%s', Int %d, Double %f\n", b"Hi", 10, 2.2)
String 'Hi', Int 10, Double 2.200000
37
>>>
```

Specifying a format protects against incompatible argument types (just as a prototype for a C function), and tries to convert the arguments to valid types:

```
>>> printf(b"%d %d %d", 1, 2, 3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ArgumentError: argument 2: exceptions.TypeError: wrong type
>>> printf(b"%s %d %f\n", b"X", 2, 3)
X 2 3.000000
13
>>>
```

If you have defined your own classes which you pass to function calls, you have to implement a `.meth:from_param` class method for them to be able to use them in the `.attr:argtypes` sequence. The `.meth:from_param` class method receives the Python object passed to the function call, it should do a typecheck or whatever is needed to make sure this object is acceptable, and then return the object itself, its `.attr:_as_parameter_` attribute, or whatever you want to pass as the C function argument in this case. Again, the result should be an integer, string, bytes, a `.mod:ctypes` instance, or an object with an `.attr:_as_parameter_` attribute.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 430); [backlink](#)

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 430); [backlink](#)

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 430); [backlink](#)

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 430); [backlink](#)

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 430); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 430); [backlink](#)

Unknown interpreted text role "attr".

Return types

By default functions are assumed to return the C `.ctype:int` type. Other return types can be specified by setting the `.attr:restype` attribute of the function object.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 446); [backlink](#)

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 446); [backlink](#)

Unknown interpreted text role "attr".

Here is a more advanced example, it uses the `strchr` function, which expects a string pointer and a char, and returns a pointer to a string:

```
>>> strchr = libc.strchr
>>> strchr(b"abcdef", ord("d")) # doctest: +SKIP
8059983
>>> strchr.restype = c_char_p # c_char_p is a pointer to a string
>>> strchr(b"abcdef", ord("d"))
b'def'
>>> print(strchr(b"abcdef", ord("x")))
None
>>>
```

If you want to avoid the `ord("x")` calls above, you can set the `:attr:'argtypes'` attribute, and the second argument will be converted from a single character Python bytes object into a C char:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 463); [backlink](#)

Unknown interpreted text role "attr".

```
>>> strchr.restype = c_char_p
>>> strchr.argtypes = [c_char_p, c_char]
>>> strchr(b"abcdef", b"d")
'def'
>>> strchr(b"abcdef", b"def")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ArgumentError: argument 2: exceptions.TypeError: one character string expected
>>> print(strchr(b"abcdef", b"x"))
None
>>> strchr(b"abcdef", b"d")
'def'
>>>
```

You can also use a callable Python object (a function or a class for example) as the `:attr:'restype'` attribute, if the foreign function returns an integer. The callable will be called with the *integer* the C function returns, and the result of this call will be used as the result of your function call. This is useful to check for error return values and automatically raise an exception:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 481); [backlink](#)

Unknown interpreted text role "attr".

```
>>> GetModuleHandle = windll.kernel32.GetModuleHandleA # doctest: +WINDOWS
>>> def ValidHandle(value):
...     if value == 0:
...         raise WinError()
...     return value
...
>>>
>>> GetModuleHandle.restype = ValidHandle # doctest: +WINDOWS
>>> GetModuleHandle(None) # doctest: +WINDOWS
486539264
>>> GetModuleHandle("something silly") # doctest: +WINDOWS
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 3, in ValidHandle
OSError: [Errno 126] The specified module could not be found.
>>>
```

`WinError` is a function which will call Windows `FormatMessage()` api to get the string representation of an error code, and *returns* an exception. `WinError` takes an optional error code parameter, if no one is used, it calls `:func:'GetLastError'` to retrieve it.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 504); [backlink](#)

Unknown interpreted text role "func".

Please note that a much more powerful error checking mechanism is available through the `:attr:'errcheck'` attribute; see the reference manual for details.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 509); [backlink](#)

Unknown interpreted text role "attr".

Passing pointers (or: passing parameters by reference)

Sometimes a C api function expects a *pointer* to a data type as parameter, probably to write into the corresponding location, or if the data is too large to be passed by value. This is also known as *passing parameters by reference*.

`:mod:'ctypes'` exports the `:func:'byref'` function which is used to pass parameters by reference. The same effect can be achieved with the `:func:'pointer'` function, although `:func:'pointer'` does a lot more work since it constructs a real pointer object, so it is faster to use `:func:'byref'` if you don't need the pointer object in Python itself:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 522); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 522); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 522); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 522); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 522); [backlink](#)

Unknown interpreted text role "func".

```
>>> i = c_int()
>>> f = c_float()
>>> s = create_string_buffer(b'\000' * 32)
>>> print(i.value, f.value, repr(s.value))
0 0.0 b''
>>> libc.sscanf(b"1 3.14 Hello", b"%d %f %s",
...             byref(i), byref(f), s)
3
>>> print(i.value, f.value, repr(s.value))
1 3.1400001049 b'Hello'
>>>
```

Structures and unions

Structures and unions must derive from the `:class:'Structure'` and `:class:'Union'` base classes which are defined in the `:mod:'ctypes'` module. Each subclass must define a `:attr:'_fields_'` attribute. `:attr:'_fields_'` must be a list of 2-tuples, containing a *field name* and a *field type*.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 546); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 546); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 546); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 546); [backlink](#)

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 546); [backlink](#)

Unknown interpreted text role "attr".

The field type must be a `mod:`ctypes`` type like `class:`c_int``, or any other derived `mod:`ctypes`` type: structure, union, array, pointer.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 551); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 551); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 551); [backlink](#)

Unknown interpreted text role "mod".

Here is a simple example of a POINT structure, which contains two integers named *x* and *y*, and also shows how to initialize a structure in the constructor:

```
>>> from ctypes import *
>>> class POINT(Structure):
...     _fields_ = [("x", c_int),
...                 ("y", c_int)]
...
>>> point = POINT(10, 20)
>>> print(point.x, point.y)
10 20
>>> point = POINT(y=5)
>>> print(point.x, point.y)
0 5
>>> POINT(1, 2, 3)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: too many initializers
>>>
```

You can, however, build much more complicated structures. A structure can itself contain other structures by using a structure as a field type.

Here is a RECT structure which contains two POINTs named *upperleft* and *lowerright*:

```
>>> class RECT(Structure):
...     _fields_ = [("upperleft", POINT),
...                 ("lowerright", POINT)]
...
>>> rc = RECT(point)
>>> print(rc.upperleft.x, rc.upperleft.y)
0 5
>>> print(rc.lowerright.x, rc.lowerright.y)
0 0
>>>
```

Nested structures can also be initialized in the constructor in several ways:

```
>>> r = RECT(POINT(1, 2), POINT(3, 4))
>>> r = RECT((1, 2), (3, 4))
```

Field `term` descriptor's can be retrieved from the `class`, they are useful for debugging because they can provide useful information:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 596); [backlink](#)

Unknown interpreted text role "term".

```
>>> print(POINT.x)
<Field type=c_long, ofs=0, size=4>
>>> print(POINT.y)
<Field type=c_long, ofs=4, size=4>
>>>
```

Warning

`mod:ctypes` does not support passing unions or structures with bit-fields to functions by value. While this may work on 32-bit x86, it's not guaranteed by the library to work in the general case. Unions and structures with bit-fields should always be passed to functions by pointer.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 610); [backlink](#)

Unknown interpreted text role "mod".

Structure/union alignment and byte order

By default, Structure and Union fields are aligned in the same way the C compiler does it. It is possible to override this behavior by specifying a `attr: _pack_` class attribute in the subclass definition. This must be set to a positive integer and specifies the maximum alignment for the fields. This is what `#pragma pack(n)` also does in MSVC.

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

Unknown interpreted text role "attr".

`mod:ctypes` uses the native byte order for Structures and Unions. To build structures with non-native byte order, you can use one of the `:class: BigEndianStructure`, `:class: LittleEndianStructure`, `:class: BigEndianUnion`, and `:class: LittleEndianUnion` base classes. These classes cannot contain pointer fields.

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

Unknown interpreted text role "mod".

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "class".

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

Unknown interpreted text role "class".

Bit fields in structures and unions

It is possible to create structures and unions containing bit fields. Bit fields are only possible for integer fields, the bit width is specified as the third item in the `:attr: `_fields_`` tuples:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 636); [backlink](#)

Unknown interpreted text role "attr".

```
>>> class Int(Structure):
...     _fields_ = [("first_16", c_int, 16),
...                 ("second_16", c_int, 16)]
...
>>> print(Int.first_16)
<Field type=c_long, ofs=0:0, bits=16>
>>> print(Int.second_16)
<Field type=c_long, ofs=0:16, bits=16>
>>>
```

Arrays

Arrays are sequences, containing a fixed number of instances of the same type.

The recommended way to create array types is by multiplying a data type with a positive integer:

```
TenPointsArrayType = POINT * 10
```

Here is an example of a somewhat artificial data type, a structure containing 4 POINTs among other stuff:

```
>>> from ctypes import *
>>> class POINT(Structure):
...     _fields_ = ("x", c_int), ("y", c_int)
...
>>> class MyStruct(Structure):
...     _fields_ = [("a", c_int),
...                 ("b", c_float),
...                 ("point_array", POINT * 4)]
...
>>> print(len(MyStruct().point_array))
4
>>>
```

Instances are created in the usual way, by calling the class:

```
arr = TenPointsArrayType()
for pt in arr:
    print(pt.x, pt.y)
```

The above code print a series of 0 0 lines, because the array contents is initialized to zeros.

Initializers of the correct type can also be specified:

```
>>> from ctypes import *
>>> TenIntegers = c_int * 10
>>> ii = TenIntegers(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
>>> print(ii)
<c_long_Array_10 object at 0x...>
>>> for i in ii: print(i, end=" ")
...
1 2 3 4 5 6 7 8 9 10
>>>
```

Pointers

Pointer instances are created by calling the `:func: `pointer`` function on a `:mod: `ctypes`` type:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 706); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 706); [backlink](#)

Unknown interpreted text role "mod".

```
>>> from ctypes import *
```

```
>>> i = c_int(42)
>>> pi = pointer(i)
>>>
```

Pointer instances have a `attr:~_Pointer.contents` attribute which returns the object to which the pointer points, the `i` object above:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 714); [backlink](#)

Unknown interpreted text role "attr".

```
>>> pi.contents
c_long(42)
>>>
```

Note that `mod:'ctypes'` does not have OOR (original object return), it constructs a new, equivalent object each time you retrieve an attribute:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 721); [backlink](#)

Unknown interpreted text role "mod".

```
>>> pi.contents is i
False
>>> pi.contents is pi.contents
False
>>>
```

Assigning another `class:'c_int'` instance to the pointer's contents attribute would cause the pointer to point to the memory location where this is stored:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 730); [backlink](#)

Unknown interpreted text role "class".

```
>>> i = c_int(99)
>>> pi.contents = i
>>> pi.contents
c_long(99)
>>>
```

Pointer instances can also be indexed with integers:

```
>>> pi[0]
99
>>>
```

Assigning to an integer index changes the pointed to value:

```
>>> print(i)
c_long(99)
>>> pi[0] = 22
>>> print(i)
c_long(22)
>>>
```

It is also possible to use indexes different from 0, but you must know what you're doing, just as in C: You can access or change arbitrary memory locations. Generally you only use this feature if you receive a pointer from a C function, and you *know* that the pointer actually points to an array instead of a single item.

Behind the scenes, the `func:'pointer'` function does more than simply create pointer instances, it has to create pointer *types* first. This is done with the `func:'POINTER'` function, which accepts any `mod:'ctypes'` type, and returns a new type:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 763); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 763); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 763); [backlink](#)

Unknown interpreted text role "mod".

```
>>> PI = POINTER(c_int)
>>> PI
<class 'ctypes.LP_c_long'>
>>> PI(42)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: expected c_long instead of int
>>> PI(c_int(42))
<ctypes.LP_c_long object at 0x...>
>>>
```

Calling the pointer type without an argument creates a `NULL` pointer. `NULL` pointers have a `False` boolean value:

```
>>> null_ptr = POINTER(c_int)()
>>> print(bool(null_ptr))
False
>>>
```

`mod:'ctypes'` checks for `NULL` when dereferencing pointers (but dereferencing invalid non-`NULL` pointers would crash Python):

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 787); [backlink](#)

Unknown interpreted text role "mod".

```
>>> null_ptr[0]
Traceback (most recent call last):
  ....
ValueError: NULL pointer access
>>>

>>> null_ptr[0] = 1234
Traceback (most recent call last):
  ....
ValueError: NULL pointer access
>>>
```

Type conversions

Usually, `ctypes` does strict type checking. This means, if you have `POINTER(c_int)` in the `attr:'argtypes'` list of a function or as the type of a member field in a structure definition, only instances of exactly the same type are accepted. There are some exceptions to this rule, where `ctypes` accepts other objects. For example, you can pass compatible array instances instead of pointer types. So, for `POINTER(c_int)`, `ctypes` accepts an array of `c_int`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 808); [backlink](#)

Unknown interpreted text role "attr".

```
>>> class Bar(Structure):
...     _fields_ = [("count", c_int), ("values", POINTER(c_int))]
...
>>> bar = Bar()
>>> bar.values = (c_int * 3)(1, 2, 3)
>>> bar.count = 3
>>> for i in range(bar.count):
...     print(bar.values[i])
...
1
2
3
>>>
```

In addition, if a function argument is explicitly declared to be a pointer type (such as `POINTER(c_int)`) in `attr:'argtypes'`, an object of the pointed type (`c_int` in this case) can be passed to the function. `ctypes` will apply the required `func:'byref'` conversion in this case automatically.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 829); [backlink](#)

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 829); [backlink](#)

Unknown interpreted text role "func".

To set a POINTER type field to NULL, you can assign None:

```
>>> bar.values = None
>>>
```

Sometimes you have instances of incompatible types. In C, you can cast one type into another type. `mod:'ctypes'` provides a `:func:'cast'` function which can be used in the same way. The `Bar` structure defined above accepts `POINTER(c_int)` pointers or `:class:'c_int'` arrays for its `values` field, but not instances of other types:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 841); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 841); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 841); [backlink](#)

Unknown interpreted text role "class".

```
>>> bar.values = (c_byte * 4)()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: incompatible types, c_byte_Array_4 instance instead of LP_c_long instance
>>>
```

For these cases, the `:func:'cast'` function is handy.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 853); [backlink](#)

Unknown interpreted text role "func".

The `:func:'cast'` function can be used to cast a ctypes instance into a pointer to a different ctypes data type. `:func:'cast'` takes two parameters, a ctypes object that is or can be converted to a pointer of some kind, and a ctypes pointer type. It returns an instance of the second argument, which references the same memory block as the first argument:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 855); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 855); [backlink](#)

Unknown interpreted text role "func".

```
>>> a = (c_byte * 4)()
>>> cast(a, POINTER(c_int))
<ctypes.LP_c_long object at ...>
>>>
```

So, `:func:'cast'` can be used to assign to the `values` field of `Bar` the structure:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 866); [backlink](#)

Unknown interpreted text role "func".

```
>>> bar = Bar()
>>> bar.values = cast((c_byte * 4)(), POINTER(c_int))
>>> print(bar.values[0])
0
>>>
```

Incomplete Types

Incomplete Types are structures, unions or arrays whose members are not yet specified. In C, they are specified by forward declarations, which are defined later:

```
struct cell; /* forward declaration */

struct cell {
    char *name;
    struct cell *next;
};
```

The straightforward translation into ctypes code would be this, but it does not work:

```
>>> class cell(Structure):
...     _fields_ = [("name", c_char_p),
...                 ("next", POINTER(cell))]
...
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in cell
NameError: name 'cell' is not defined
>>>
```

because the new class `cell` is not available in the class statement itself. In `mod:ctypes`, we can define the `cell` class and set the `attr: _fields_` attribute later, after the class statement:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 905); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 905); [backlink](#)

Unknown interpreted text role "attr".

```
>>> from ctypes import *
>>> class cell(Structure):
...     pass
...
>>> cell._fields_ = [("name", c_char_p),
...                  ("next", POINTER(cell))]
>>>
```

Let's try it. We create two instances of `cell`, and let them point to each other, and finally follow the pointer chain a few times:

```
>>> c1 = cell()
>>> c1.name = b"foo"
>>> c2 = cell()
>>> c2.name = b"bar"
>>> c1.next = pointer(c2)
>>> c2.next = pointer(c1)
>>> p = c1
>>> for i in range(8):
...     print(p.name, end=" ")
...     p = p.next[0]
...
foo bar foo bar foo bar foo bar
>>>
```

Callback functions

`mod:ctypes` allows creating C callable function pointers from Python callables. These are sometimes called *callback functions*.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 940); [backlink](#)

Unknown interpreted text role "mod".

First, you must create a class for the callback function. The class knows the calling convention, the return type, and the number and

types of arguments this function will receive.

The `:func:'CFUNCTYPE'` factory function creates types for callback functions using the `cdecl` calling convention. On Windows, the `:func:'WINFUNCTYPE'` factory function creates types for callback functions using the `stdcall` calling convention.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 947); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 947); [backlink](#)

Unknown interpreted text role "func".

Both of these factory functions are called with the result type as first argument, and the callback functions expected argument types as the remaining arguments.

I will present an example here which uses the standard C library's `:c:func:'qsort'` function, that is used to sort items with the help of a callback function. `:c:func:'qsort'` will be used to sort an array of integers:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 956); [backlink](#)

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 956); [backlink](#)

Unknown interpreted text role "c:func".

```
>>> IntArray5 = c_int * 5
>>> ia = IntArray5(5, 1, 7, 33, 99)
>>> qsort = libc.qsort
>>> qsort.restype = None
>>>
```

`:func:'qsort'` must be called with a pointer to the data to sort, the number of items in the data array, the size of one item, and a pointer to the comparison function, the callback. The callback will then be called with two pointers to items, and it must return a negative integer if the first item is smaller than the second, a zero if they are equal, and a positive integer otherwise.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 966); [backlink](#)

Unknown interpreted text role "func".

So our callback function receives pointers to integers, and must return an integer. First we create the `type` for the callback function:

```
>>> CMPFUNC = CFUNCTYPE(c_int, POINTER(c_int), POINTER(c_int))
>>>
```

To get started, here is a simple callback that shows the values it gets passed:

```
>>> def py_cmp_func(a, b):
...     print("py_cmp_func", a[0], b[0])
...     return 0
...
>>> cmp_func = CMPFUNC(py_cmp_func)
>>>
```

The result:

```
>>> qsort(ia, len(ia), sizeof(c_int), cmp_func) # doctest: +LINUX
py_cmp_func 5 1
py_cmp_func 33 99
py_cmp_func 7 33
py_cmp_func 5 7
py_cmp_func 1 7
>>>
```

Now we can actually compare the two items and return a useful result:

```
>>> def py_cmp_func(a, b):
...     print("py_cmp_func", a[0], b[0])
```



```

...     return a[0] - b[0]
...
>>>
>>> qsort(ia, len(ia), sizeof(c_int), CMPFUNC(py_cmp_func)) # doctest: +LINUX
py_cmp_func 5 1
py_cmp_func 33 99
py_cmp_func 7 33
py_cmp_func 1 7
py_cmp_func 5 7
>>>

```

As we can easily check, our array is sorted now:

```

>>> for i in ia: print(i, end=" ")
...
1 5 7 33 99
>>>

```

The function factories can be used as decorator factories, so we may as well write:

```

>>> @CFUNCTYPE(c_int, POINTER(c_int), POINTER(c_int))
... def py_cmp_func(a, b):
...     print("py_cmp_func", a[0], b[0])
...     return a[0] - b[0]
...
>>> qsort(ia, len(ia), sizeof(c_int), py_cmp_func)
py_cmp_func 5 1
py_cmp_func 33 99
py_cmp_func 7 33
py_cmp_func 1 7
py_cmp_func 5 7
>>>

```

Note

Make sure you keep references to `:func:'CFUNCTYPE'` objects as long as they are used from C code. `:mod:'ctypes'` doesn't, and if you don't, they may be garbage collected, crashing your program when a callback is made.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1038); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1038); [backlink](#)

Unknown interpreted text role "mod".

Also, note that if the callback function is called in a thread created outside of Python's control (e.g. by the foreign code that calls the callback), ctypes creates a new dummy Python thread on every invocation. This behavior is correct for most purposes, but it means that values stored with `:class:'threading.local'` will *not* survive across different callbacks, even when those calls are made from the same C thread.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1042); [backlink](#)

Unknown interpreted text role "class".

Accessing values exported from dlls

Some shared libraries not only export functions, they also export variables. An example in the Python library itself is the `:c:data:'Py_OptimizeFlag'`, an integer set to 0, 1, or 2, depending on the `:option:'-O'` or `:option:'-OO'` flag given on startup.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1054); [backlink](#)

Unknown interpreted text role "c:data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1054); [backlink](#)

Unknown interpreted text role "option".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1054); [backlink](#)

Unknown interpreted text role "option".

`:mod:'ctypes'` can access values like this with the `:meth:'in_dll'` class methods of the type. `pythonapi` is a predefined symbol giving access to the Python C api:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1059); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1059); [backlink](#)

Unknown interpreted text role "meth".

```
>>> opt_flag = c_int.in_dll(pythonapi, "Py_OptimizeFlag")
>>> print(opt_flag)
c_long(0)
>>>
```

If the interpreter would have been started with `:option:'-O'`, the sample would have printed `c_long(1)`, or `c_long(2)` if `:option:'-OO'` would have been specified.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1068); [backlink](#)

Unknown interpreted text role "option".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1068); [backlink](#)

Unknown interpreted text role "option".

An extended example which also demonstrates the use of pointers accesses the `:cdata:'PyImport_FrozenModules'` pointer exported by Python.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1072); [backlink](#)

Unknown interpreted text role "c:data".

Quoting the docs for that value:

This pointer is initialized to point to an array of `:ctype:'struct _frozen'` records, terminated by one whose members are all NULL or zero. When a frozen module is imported, it is searched in this table. Third-party code could play tricks with this to provide a dynamically created collection of frozen modules.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1077); [backlink](#)

Unknown interpreted text role "c:type".

So manipulating this pointer could even prove useful. To restrict the example size, we show only how this table can be read with `:mod:'ctypes'`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1082); [backlink](#)

Unknown interpreted text role "mod".

```
>>> from ctypes import *
>>>
>>> class struct_frozen(Structure):
...     _fields_ = [("name", c_char_p),
...                  ("code", POINTER(c_ubyte)),
...                  ("size", c_int),
...                  ("get_code", POINTER(c_ubyte)), # Function pointer
...                  ]
...
>>>
```

We have defined the `ctype:struct_frozen` data type, so we can get the pointer to the table:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1096); [backlink](#)
 Unknown interpreted text role "ctype".

```
>>> FrozenTable = POINTER(struct_frozen)
>>> table = FrozenTable.in_dll(pythonapi, "_PyImport_FrozenBootstrap")
>>>
```

Since `table` is a pointer to the array of `struct_frozen` records, we can iterate over it, but we just have to make sure that our loop terminates, because pointers have no size. Sooner or later it would probably crash with an access violation or whatever, so it's better to break out of the loop when we hit the `NULL` entry:

```
>>> for item in table:
...     if item.name is None:
...         break
...     print(item.name.decode("ascii"), item.size)
...
_frozen_importlib 31764
_frozen_importlib_external 41499
zipimport 12345
>>>
```

The fact that standard Python has a frozen module and a frozen package (indicated by the negative `size` member) is not well known, it is only used for testing. Try it out with `import __hello__` for example.

Surprises

There are some edges in `mod:ctypes` where you might expect something other than what actually happens.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1129); [backlink](#)
 Unknown interpreted text role "mod".

Consider the following example:

```
>>> from ctypes import *
>>> class POINT(Structure):
...     _fields_ = ("x", c_int), ("y", c_int)
...
>>> class RECT(Structure):
...     _fields_ = ("a", POINT), ("b", POINT)
...
>>> p1 = POINT(1, 2)
>>> p2 = POINT(3, 4)
>>> rc = RECT(p1, p2)
>>> print(rc.a.x, rc.a.y, rc.b.x, rc.b.y)
1 2 3 4
>>> # now swap the two points
>>> rc.a, rc.b = rc.b, rc.a
>>> print(rc.a.x, rc.a.y, rc.b.x, rc.b.y)
3 4 3 4
>>>
```

Hm. We certainly expected the last statement to print `3 4 1 2`. What happened? Here are the steps of the `rc.a, rc.b = rc.b, rc.a` line above:

```
>>> temp0, temp1 = rc.b, rc.a
>>> rc.a = temp0
>>> rc.b = temp1
>>>
```

Note that `temp0` and `temp1` are objects still using the internal buffer of the `rc` object above. So executing `rc.a = temp0` copies the

buffer contents of `temp0` into `rc`'s buffer. This, in turn, changes the contents of `temp1`. So, the last assignment `rc.b = temp1`, doesn't have the expected effect.

Keep in mind that retrieving sub-objects from Structure, Unions, and Arrays doesn't *copy* the sub-object, instead it retrieves a wrapper object accessing the root-object's underlying buffer.

Another example that may behave differently from what one would expect is this:

```
>>> s = c_char_p()
>>> s.value = b"abc def ghi"
>>> s.value
b'abc def ghi'
>>> s.value is s.value
False
>>>
```

Note

Objects instantiated from `class: 'c_char_p'` can only have their value set to bytes or integers.

System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 1182); [backlink](#)

Unknown interpreted text role "class".

Why is it printing `False`? `ctypes` instances are objects containing a memory block plus some `term: 'descriptor'`'s accessing the contents of the memory. Storing a Python object in the memory block does not store the object itself, instead the contents of the object is stored. Accessing the contents again constructs a new Python object each time!

System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 1185); [backlink](#)

Unknown interpreted text role "term".

Variable-sized data types

`mod: 'ctypes'` provides some support for variable-sized arrays and structures.

System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 1197); [backlink](#)

Unknown interpreted text role "mod".

The `func: 'resize'` function can be used to resize the memory buffer of an existing `ctypes` object. The function takes the object as first argument, and the requested size in bytes as the second argument. The memory block cannot be made smaller than the natural memory block specified by the objects type, a `exc: 'ValueError'` is raised if this is tried:

System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 1199); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D: \onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 1199); [backlink](#)

Unknown interpreted text role "exc".

```
>>> short_array = (c_short * 4)()
>>> print(sizeof(short_array))
8
>>> resize(short_array, 4)
Traceback (most recent call last):
...
ValueError: minimum size is 8
>>> resize(short_array, 32)
>>> sizeof(short_array)
32
>>> sizeof(type(short_array))
8
>>>
```

This is nice and fine, but how would one access the additional elements contained in this array? Since the type still only knows about 4 elements, we get errors accessing other elements:

```
>>> short_array[:]
[0, 0, 0, 0]
>>> short_array[7]
Traceback (most recent call last):
...
IndexError: invalid index
>>>
```

Another way to use variable-sized data types with `mod:ctypes` is to use the dynamic nature of Python, and (re-)define the data type after the required size is already known, on a case by case basis.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 1231); [backlink](#)
Unknown interpreted text role "mod".

ctypes reference

Finding shared libraries

When programming in a compiled language, shared libraries are accessed when compiling/linking a program, and when the program is run.

The purpose of the `func:find_library` function is to locate a library in a way similar to what the compiler or runtime loader does (on platforms with several versions of a shared library the most recent should be loaded), while the ctypes library loaders act like when a program is run, and call the runtime loader directly.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 1250); [backlink](#)
Unknown interpreted text role "func".

The `mod:ctypes.util` module provides a function which can help to determine the library to load.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 1256); [backlink](#)
Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 1260)
Unknown directive type "data".

```
.. data:: find_library(name)
:module: ctypes.util
:noindex:
```

Try to find a library and return a pathname. *name* is the library name without any prefix like *lib*, suffix like ``.so``, ``.dylib`` or version number (this is the form used for the posix linker option `:option:~!-l~`). If no library can be found, returns ```None```.

The exact functionality is system dependent.

On Linux, `func:find_library` tries to run external programs (`/sbin/ldconfig`, `gcc`, `objdump` and `ld`) to find the library file. It returns the filename of the library file.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 1271); [backlink](#)
Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 1275)
Unknown directive type "versionchanged".

```
.. versionchanged:: 3.6
   On Linux, the value of the environment variable ``LD_LIBRARY_PATH`` is used
   when searching for libraries, if a library cannot be found by any other means.
```

Here are some examples:

```
>>> from ctypes.util import find_library
>>> find_library("m")
'libm.so.6'
>>> find_library("c")
'libc.so.6'
>>> find_library("bz2")
'libbz2.so.1.0'
>>>
```

On macOS, `func:find_library` tries several predefined naming schemes and paths to locate the library, and returns a full pathname if successful:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 1290); [backlink](#)

Unknown interpreted text role "func".

```
>>> from ctypes.util import find_library
>>> find_library("c")
'/usr/lib/libc.dylib'
>>> find_library("m")
'/usr/lib/libm.dylib'
>>> find_library("bz2")
'/usr/lib/libbz2.dylib'
>>> find_library("AGL")
'/System/Library/Frameworks/AGL.framework/AGL'
>>>
```

On Windows, `func:find_library` searches along the system search path, and returns the full pathname, but since there is no predefined naming scheme a call like `find_library("c")` will fail and return `None`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 1304); [backlink](#)

Unknown interpreted text role "func".

If wrapping a shared library with `mod:ctypes`, it *may* be better to determine the shared library name at development time, and hardcode that into the wrapper module instead of using `func:find_library` to locate the library at runtime.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 1308); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 1308); [backlink](#)

Unknown interpreted text role "func".

Loading shared libraries

There are several ways to load shared libraries into the Python process. One way is to instantiate one of the following classes:

Instances of this class represent loaded shared libraries. Functions in these libraries use the standard C calling convention, and are assumed to return `c:type:int`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 1324); [backlink](#)

Unknown interpreted text role "c:type".

On Windows creating a `class:CDLL` instance may fail even if the DLL name exists. When a dependent DLL of the loaded DLL is not found, a `exc:OSErr` error is raised with the message "[WinError 126] The specified module could not be found". This error message does not contain the name of the missing DLL because the Windows API does not return this information making this error hard to diagnose. To resolve this error and determine which DLL is not found, you need to find the list of dependent DLLs and

determine which one is not found using Windows debugging and tracing tools.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1328); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1328); [backlink](#)

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1338)

Unknown directive type "seealso".

```
.. seealso::
```

```
    \Microsoft DUMPBIN tool <https://docs.microsoft.com/cpp/build/reference/dependents>`_
    -- A tool to find DLL dependents.
```

Windows only: Instances of this class represent loaded shared libraries, functions in these libraries use the `stdcall` calling convention, and are assumed to return the windows specific `:class:`HRESULT`` code. `:class:`HRESULT`` values contain information specifying whether the function call failed or succeeded, together with additional error code. If the return value signals a failure, an `:class:`OSError`` is automatically raised.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1346); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1346); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1346); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1353)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.3
   :exc:`WindowsError` used to be raised.
```

Windows only: Instances of this class represent loaded shared libraries, functions in these libraries use the `stdcall` calling convention, and are assumed to return `:c:type:`int`` by default.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1359); [backlink](#)

Unknown interpreted text role "c:type".

On Windows CE only the standard calling convention is used, for convenience the `:class:`WinDLL`` and `:class:`OleDLL`` use the standard calling convention on this platform.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1363); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1363); [backlink](#)

Unknown interpreted text role "class".

The Python `term` 'global interpreter lock' is released before calling any function exported by these libraries, and reacquired afterwards.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1367); [backlink](#)

Unknown interpreted text role "term".

Instances of this class behave like `class:CDLL` instances, except that the Python GIL is *not* released during the function call, and after the function execution the Python error flag is checked. If the error flag is set, a Python exception is raised.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1373); [backlink](#)

Unknown interpreted text role "class".

Thus, this is only useful to call Python C api functions directly.

All these classes can be instantiated by calling them with at least one argument, the pathname of the shared library. If you have an existing handle to an already loaded shared library, it can be passed as the `handle` named parameter, otherwise the underlying platforms `dlopen` or `LoadLibrary` function is used to load the library into the process, and to get a handle to it.

The `mode` parameter can be used to specify how the library is loaded. For details, consult the `manpage:dlopen(3)` manpage. On Windows, `mode` is ignored. On posix systems, `RTLD_NOW` is always added, and is not configurable.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1387); [backlink](#)

Unknown interpreted text role "manpage".

The `use_errno` parameter, when set to true, enables a ctypes mechanism that allows accessing the system `data:errno` error number in a safe way. `mod:ctypes` maintains a thread-local copy of the systems `data:errno` variable; if you call foreign functions created with `use_errno=True` then the `data:errno` value before the function call is swapped with the ctypes private copy, the same happens immediately after the function call.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1392); [backlink](#)

Unknown interpreted text role "data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1392); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1392); [backlink](#)

Unknown interpreted text role "data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1392); [backlink](#)

Unknown interpreted text role "data".

The function `func:ctypes.get_errno` returns the value of the ctypes private copy, and the function `func:ctypes.set_errno` changes the ctypes private copy to a new value and returns the former value.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1399); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1399); [backlink](#)

Unknown interpreted text role "func".

The `use_last_error` parameter, when set to true, enables the same mechanism for the Windows error code which is managed by the `:func:`GetLastError`` and `:func:`SetLastError`` Windows API functions; `:func:`ctypes.get_last_error`` and `:func:`ctypes.set_last_error`` are used to request and change the ctypes private copy of the windows error code.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1403); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1403); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1403); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1403); [backlink](#)

Unknown interpreted text role "func".

The `winmode` parameter is used on Windows to specify how the library is loaded (since `mode` is ignored). It takes any value that is valid for the Win32 API `LoadLibraryEx` flags parameter. When omitted, the default is to use the flags that result in the most secure DLL load to avoiding issues such as DLL hijacking. Passing the full path to the DLL is the safest way to ensure the correct library and dependencies are loaded.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1416)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.8
   Added *winmode* parameter.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1420)

Unknown directive type "data".

```
.. data:: RTLD_GLOBAL
   :noindex:
```

Flag to use as `*mode*` parameter. On platforms where this flag is not available, it is defined as the integer zero.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1427)

Unknown directive type "data".

```
.. data:: RTLD_LOCAL
   :noindex:
```

Flag to use as `*mode*` parameter. On platforms where this is not available, it is the same as `*RTLD_GLOBAL*`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1434)

Unknown directive type "data".

```
.. data:: DEFAULT_MODE
: noindex:
```

The default mode which is used to load shared libraries. On OSX 10.3, this is *RTLD_GLOBAL*, otherwise it is the same as *RTLD_LOCAL*.

Instances of these classes have no public methods. Functions exported by the shared library can be accessed as attributes or by index. Please note that accessing the function through an attribute caches the result and therefore accessing it repeatedly returns the same object each time. On the other hand, accessing it through an index returns a new object each time:

```
>>> from ctypes import CDLL
>>> libc = CDLL("libc.so.6") # On Linux
>>> libc.time == libc.time
True
>>> libc['time'] == libc['time']
False
```

The following public attributes are available, their name starts with an underscore to not clash with exported function names:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1457)

Unknown directive type "attribute".

```
.. attribute:: PyDLL._handle
```

The system handle used to access the library.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1462)

Unknown directive type "attribute".

```
.. attribute:: PyDLL._name
```

The name of the library passed in the constructor.

Shared libraries can also be loaded by using one of the prefabricated objects, which are instances of the `:class:'LibraryLoader'` class, either by calling the `:meth:'LoadLibrary'` method, or by retrieving the library as attribute of the loader instance.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1466); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1466); [backlink](#)

Unknown interpreted text role "meth".

Class which loads shared libraries. *dlltype* should be one of the `:class:'CDLL'`, `:class:'PyDLL'`, `:class:'WinDLL'`, or `:class:'OleDLL'` types.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1474); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1474); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1474); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1474); [backlink](#)

Unknown interpreted text role "class".

`.meth'__getattr__'` has special behavior: It allows loading a shared library by accessing it as attribute of a library loader instance. The result is cached, so repeated attribute accesses return the same library each time.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1477); [backlink](#)

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1481)

Unknown directive type "method".

```
.. method:: LoadLibrary(name)
```

Load a shared library into the process and return it. This method always returns a new instance of the library.

These prefabricated library loaders are available:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1489)

Unknown directive type "data".

```
.. data:: cdll
: noindex:
```

Creates :class:`CDLL` instances.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1495)

Unknown directive type "data".

```
.. data:: windll
: noindex:
```

Windows only: Creates :class:`WinDLL` instances.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1501)

Unknown directive type "data".

```
.. data:: oledll
: noindex:
```

Windows only: Creates :class:`OleDLL` instances.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1507)

Unknown directive type "data".

```
.. data:: pydll
```

```
:noindex:

Creates :class:`PyDLL` instances.
```

For accessing the C Python api directly, a ready-to-use Python shared library object is available:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1516)

Unknown directive type "data".

```
.. data:: pythonapi
   :noindex:
```

An instance of :class:`PyDLL` that exposes Python C API functions as attributes. Note that all these functions are assumed to return C :c:type:`int`, which is of course not always the truth, so you have to assign the correct :attr:`restype` attribute to use these functions.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1524)

Unknown directive type "audit-event".

```
.. audit-event:: ctypes.dlopen name ctypes.LibraryLoader
```

Loading a library through any of these objects raises an :ref:`auditing event <auditing>` ``ctypes.dlopen`` with string argument ``name``, the name used to load the library.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1530)

Unknown directive type "audit-event".

```
.. audit-event:: ctypes.dlsym library,name ctypes.LibraryLoader
```

Accessing a function on a loaded library raises an auditing event ``ctypes.dlsym`` with arguments ``library`` (the library object) and ``name`` (the symbol's name as a string or integer).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1536)

Unknown directive type "audit-event".

```
.. audit-event:: ctypes.dlsym/handle handle,name ctypes.LibraryLoader
```

In cases when only the library handle is available rather than the object, accessing a function raises an auditing event ``ctypes.dlsym/handle`` with arguments ``handle`` (the raw library handle) and ``name``.

Foreign functions

As explained in the previous section, foreign functions can be accessed as attributes of loaded shared libraries. The function objects created in this way by default accept any number of arguments, accept any ctypes data instances as arguments, and return the default result type specified by the library loader. They are instances of a private class:

Base class for C callable foreign functions.

Instances of foreign functions are also C compatible data types; they represent C function pointers.

This behavior can be customized by assigning to special attributes of the foreign function object.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1564)

Unknown directive type "attribute".

```
.. attribute:: restype
```

Assign a ctypes type to specify the result type of the foreign function. Use ``None`` for :c:type:`void`, a function not returning anything.

It is possible to assign a callable Python object that is not a ctypes type, in this case the function is assumed to return a C :c:type:`int`, and the callable will be called with this integer, allowing further processing or error checking. Using this is deprecated, for more flexible post processing or error checking use a ctypes data type as :attr:`restype` and assign a callable to the :attr:`errcheck` attribute.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 1576)

Unknown directive type "attribute".

```
.. attribute:: argtypes
```

Assign a tuple of ctypes types to specify the argument types that the function accepts. Functions using the ``stdcall`` calling convention can only be called with the same number of arguments as the length of this tuple; functions using the C calling convention accept additional, unspecified arguments as well.

When a foreign function is called, each actual argument is passed to the :meth:`from_param` class method of the items in the :attr:`argtypes` tuple, this method allows adapting the actual argument to an object that the foreign function accepts. For example, a :class:`c_char_p` item in the :attr:`argtypes` tuple will convert a string passed as argument into a bytes object using ctypes conversion rules.

New: It is now possible to put items in argtypes which are not ctypes types, but each item must have a :meth:`from_param` method which returns a value usable as argument (integer, string, ctypes instance). This allows defining adapters that can adapt custom objects as function parameters.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 1596)

Unknown directive type "attribute".

```
.. attribute:: errcheck
```

Assign a Python function or another callable to this attribute. The callable will be called with three or more arguments:

```
.. function:: callable(result, func, arguments)
: noindex:
: module:
```

result is what the foreign function returns, as specified by the :attr:`restype` attribute.

func is the foreign function object itself, this allows reusing the same callable object to check or post process the results of several functions.

arguments is a tuple containing the parameters originally passed to the function call, this allows specializing the behavior on the arguments used.

The object that this function returns will be returned from the foreign function call, but it can also check the result value and raise an exception if the foreign function call failed.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 1621)

Unknown directive type "exception".

```
.. exception:: ArgumentError
```

This exception is raised when a foreign function call cannot convert one of the passed arguments.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1627)

Unknown directive type "audit-event".

```
.. audit-event:: ctypes.seh_exception code foreign-functions
```

On Windows, when a foreign function call raises a system exception (for example, due to an access violation), it will be captured and replaced with a suitable Python exception. Further, an auditing event `ctypes.seh_exception` with argument code` will be raised, allowing an audit hook to replace the exception with its own.`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1635)

Unknown directive type "audit-event".

```
.. audit-event:: ctypes.call_function func_pointer,arguments foreign-functions
```

Some ways to invoke foreign function calls may raise an auditing event `ctypes.call_function` with arguments function pointer` and arguments`.`

Function prototypes

Foreign functions can also be created by instantiating function prototypes. Function prototypes are similar to function prototypes in C; they describe a function (return type, argument types, calling convention) without defining an implementation. The factory functions must be called with the desired result type and the argument types of the function, and can be used as decorator factories, and as such, be applied to functions through the `@wrapper` syntax. See [ref`ctypes-callback-functions`](#) for examples.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1645); [backlink](#)

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1654)

Unknown directive type "function".

```
.. function:: CFUNCTYPE(restype, *argtypes, use_errno=False, use_last_error=False)
```

The returned function prototype creates functions that use the standard C calling convention. The function will release the GIL during the call. If `*use_errno*` is set to true, the ctypes private copy of the system `:data:`errno` variable is exchanged with the real :data:`errno` value before and after the call; *use_last_error* does the same for the Windows error code.`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1664)

Unknown directive type "function".

```
.. function:: WINFUNCTYPE(restype, *argtypes, use_errno=False, use_last_error=False)
```

Windows only: The returned function prototype creates functions that use the `stdcall` calling convention, except on Windows CE where :func:`WINFUNCTYPE` is the same as :func:`CFUNCTYPE`. The function will release the GIL during the call. *use_errno* and *use_last_error* have the same meaning as above.`

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

Unknown directive type "function".

```
.. function:: PYFUNCTYPE(restype, *argtypes)
```

The returned function prototype creates functions that use the Python calling convention. The function will *not* release the GIL during the call.

Function prototypes created by these factory functions can be instantiated in different ways, depending on the type and number of the parameters in the call:

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

Unknown directive type "function".

```
.. function:: prototype(address)
: noindex:
: module:
```

Returns a foreign function at the specified address which must be an integer.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1689)

Unknown directive type "function".

```
.. function:: prototype(callable)
: noindex:
: module:
```

Create a C callable function (a callback function) from a Python *callable*.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1696)

Unknown directive type "function".

```
.. function:: prototype(func_spec[, paramflags])
: noindex:
: module:
```

Returns a foreign function exported by a shared library. **func_spec** must be a 2-tuple ``(name_or_ordinal, library)``. The first item is the name of the exported function as string, or the ordinal of the exported function as small integer. The second item is the shared library instance.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1706)

Unknown directive type "function".

```
.. function:: prototype(vtbl_index, name[, paramflags[, iid]])
: noindex:
: module:
```

Returns a foreign function that will call a COM method. **vtbl_index** is the index into the virtual function table, a small non-negative integer. **name** is name of the COM method. **iid** is an optional pointer to the interface identifier which is used in extended error reporting.

COM methods use a special calling convention: They require a pointer to the COM interface as first argument, in addition to those parameters that are specified in the `:attr:`argtypes`` tuple.

The optional *paramflags* parameter creates foreign function wrappers with much more functionality than the features described above.

paramflags must be a tuple of the same length as `:attr:`argtypes``.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1722); [backlink](#)

Unknown interpreted text role "attr".

Each item in this tuple contains further information about a parameter, it must be a tuple containing one, two, or three items.

The first item is an integer containing a combination of direction flags for the parameter:

- 1 Specifies an input parameter to the function.
- 2 Output parameter. The foreign function fills in a value.
- 4 Input parameter which defaults to the integer zero.

The optional second item is the parameter name as string. If this is specified, the foreign function can be called with named parameters.

The optional third item is the default value for this parameter.

This example demonstrates how to wrap the Windows `MessageBoxW` function so that it supports default parameters and named arguments. The C declaration from the windows header file is this:

```
WINUSERAPI int WINAPI
MessageBoxW(
    HWND hWnd,
    LPCWSTR lpText,
    LPCWSTR lpCaption,
    UINT uType);
```

Here is the wrapping with `mod:'ctypes'`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1755); [backlink](#)

Unknown interpreted text role "mod".

```
>>> from ctypes import c_int, WINFUNCTYPE, windll
>>> from ctypes.wintypes import HWND, LPCWSTR, UINT
>>> prototype = WINFUNCTYPE(c_int, HWND, LPCWSTR, LPCWSTR, UINT)
>>> paramflags = (1, "hwnd", 0), (1, "text", "Hi"), (1, "caption", "Hello from ctypes"), (1, "flags", 0)
>>> MessageBox = prototype(("MessageBoxW", windll.user32), paramflags)
```

The `MessageBox` foreign function can now be called in these ways:

```
>>> MessageBox()
>>> MessageBox(text="Spam, spam, spam")
>>> MessageBox(flags=2, text="foo bar")
```

A second example demonstrates output parameters. The win32 `GetWindowRect` function retrieves the dimensions of a specified window by copying them into `RECT` structure that the caller has to supply. Here is the C declaration:

```
WINUSERAPI BOOL WINAPI
GetWindowRect(
    HWND hWnd,
    LPRECT lpRect);
```

Here is the wrapping with `mod:'ctypes'`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 1778); [backlink](#)

Unknown interpreted text role "mod".

```
>>> from ctypes import POINTER, WINFUNCTYPE, windll, WinError
>>> from ctypes.wintypes import BOOL, HWND, RECT
>>> prototype = WINFUNCTYPE(BOOL, HWND, POINTER(RECT))
>>> paramflags = (1, "hwnd"), (2, "lprect")
>>> GetWindowRect = prototype(("GetWindowRect", windll.user32), paramflags)
>>>
```

Functions with output parameters will automatically return the output parameter value if there is a single one, or a tuple containing the output parameter values when there are more than one, so the `GetWindowRect` function now returns a `RECT` instance, when called.

Output parameters can be combined with the `:attr:'errcheck'` protocol to do further output processing and error checking. The `win32.GetWindowRect` api function returns a `BOOL` to signal success or failure, so this function could do the error checking, and raises an exception when the api call failed:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1792); [backlink](#)

Unknown interpreted text role "attr".

```
>>> def errcheck(result, func, args):
...     if not result:
...         raise WinError()
...     return args
...
>>> GetWindowRect.errcheck = errcheck
>>>
```

If the `:attr:'errcheck'` function returns the argument tuple it receives unchanged, `:mod:'ctypes'` continues the normal processing it does on the output parameters. If you want to return a tuple of window coordinates instead of a `RECT` instance, you can retrieve the fields in the function and return them instead, the normal processing will no longer take place:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1805); [backlink](#)

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1805); [backlink](#)

Unknown interpreted text role "mod".

```
>>> def errcheck(result, func, args):
...     if not result:
...         raise WinError()
...     rc = args[1]
...     return rc.left, rc.top, rc.bottom, rc.right
...
>>> GetWindowRect.errcheck = errcheck
>>>
```

Utility functions

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1826)

Unknown directive type "function".

`.. function:: addressof(obj)`

Returns the address of the memory buffer as integer. `*obj*` must be an instance of a `ctypes` type.

`.. audit-event:: ctypes.addressof obj ctypes.addressof`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1834)

Unknown directive type "function".

`.. function:: alignment(obj_or_type)`

Returns the alignment requirements of a `ctypes` type. `*obj_or_type*` must be a `ctypes` type or instance.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1840)

Unknown directive type "function".

```
.. function:: byref(obj[, offset])
```

Returns a light-weight pointer to **obj**, which must be an instance of a ctypes type. **offset** defaults to zero, and must be an integer that will be added to the internal pointer value.

`byref(obj, offset)` corresponds to this C code::

```
((char *)&obj) + offset)
```

The returned object can only be used as a foreign function call parameter. It behaves similar to `pointer(obj)`, but the construction is a lot faster.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 1854)

Unknown directive type "function".

```
.. function:: cast(obj, type)
```

This function is similar to the cast operator in C. It returns a new instance of **type** which points to the same memory block as **obj**. **type** must be a pointer type, and **obj** must be an object that can be interpreted as a pointer.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 1862)

Unknown directive type "function".

```
.. function:: create_string_buffer(init_or_size, size=None)
```

This function creates a mutable character buffer. The returned object is a ctypes array of :class:`c_char`.

init_or_size must be an integer which specifies the size of the array, or a bytes object which will be used to initialize the array items.

If a bytes object is specified as first argument, the buffer is made one item larger than its length so that the last element in the array is a NUL termination character. An integer can be passed as second argument which allows specifying the size of the array if the length of the bytes should not be used.

```
.. audit-event:: ctypes.create_string_buffer init,size ctypes.create_string_buffer
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 1878)

Unknown directive type "function".

```
.. function:: create_unicode_buffer(init_or_size, size=None)
```

This function creates a mutable unicode character buffer. The returned object is a ctypes array of :class:`c_wchar`.

init_or_size must be an integer which specifies the size of the array, or a string which will be used to initialize the array items.

If a string is specified as first argument, the buffer is made one item larger than the length of the string so that the last element in the array is a NUL termination character. An integer can be passed as second argument which allows specifying the size of the array if the length of the string should not be used.

```
.. audit-event:: ctypes.create_unicode_buffer init,size ctypes.create_unicode_buffer
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 1895)

Unknown directive type "function".

```
.. function:: DllCanUnloadNow()
```

Windows only: This function is a hook which allows implementing in-process COM servers with ctypes. It is called from the DllCanUnloadNow function that the _ctypes extension dll exports.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 1902)

Unknown directive type "function".

```
.. function:: DllGetClassObject()
```

Windows only: This function is a hook which allows implementing in-process COM servers with ctypes. It is called from the DllGetClassObject function that the ``_ctypes`` extension dll exports.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 1909)

Unknown directive type "function".

```
.. function:: find_library(name)
:module: ctypes.util
```

Try to find a library and return a pathname. *name* is the library name without any prefix like ``lib``, suffix like ``.so``, ``.dylib`` or version number (this is the form used for the posix linker option :option:`-l`). If no library can be found, returns ``None``.

The exact functionality is system dependent.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 1920)

Unknown directive type "function".

```
.. function:: find_msvcr()
:module: ctypes.util
```

Windows only: return the filename of the VC runtime library used by Python, and by the extension modules. If the name of the library cannot be determined, ``None`` is returned.

If you need to free memory, for example, allocated by an extension module with a call to the ``free(void *)``, it is important that you use the function in the same library that allocated the memory.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 1932)

Unknown directive type "function".

```
.. function:: FormatError([code])
```

Windows only: Returns a textual description of the error code *code*. If no error code is specified, the last error code is used by calling the Windows api function GetLastError.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 1939)

Unknown directive type "function".

```
.. function:: GetLastError()
```

Windows only: Returns the last error code set by Windows in the calling thread.

This function calls the Windows `GetLastError()` function directly, it does not return the ctypes-private copy of the error code.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 1945)

Unknown directive type "function".

```
.. function:: get_errno()
```

Returns the current value of the ctypes-private copy of the system :data:`errno` variable in the calling thread.

```
.. audit-event:: ctypes.get_errno "" ctypes.get_errno
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 1952)

Unknown directive type "function".

```
.. function:: get_last_error()
```

Windows only: returns the current value of the ctypes-private copy of the system :data:`LastError` variable in the calling thread.

```
.. audit-event:: ctypes.get_last_error "" ctypes.get_last_error
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 1959)

Unknown directive type "function".

```
.. function:: memmove(dst, src, count)
```

Same as the standard C memmove library function: copies **count** bytes from **src** to **dst**. **dst** and **src** must be integers or ctypes instances that can be converted to pointers.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 1966)

Unknown directive type "function".

```
.. function:: memset(dst, c, count)
```

Same as the standard C memset library function: fills the memory block at address **dst** with **count** bytes of value **c**. **dst** must be an integer specifying an address, or a ctypes instance.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 1973)

Unknown directive type "function".

```
.. function:: POINTER(type)
```

This factory function creates and returns a new ctypes pointer type. Pointer types are cached and reused internally, so calling this function repeatedly is cheap. **type** must be a ctypes type.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 1980)

Unknown directive type "function".

```
.. function:: pointer(obj)
```

This function creates a new pointer instance, pointing to *obj*. The returned object is of the type ``POINTER(type(obj))``.

Note: If you just want to pass a pointer to an object to a foreign function call, you should use ``byref(obj)`` which is much faster.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1989)

Unknown directive type "function".

```
.. function:: resize(obj, size)
```

This function resizes the internal memory buffer of *obj*, which must be an instance of a ctypes type. It is not possible to make the buffer smaller than the native size of the objects type, as given by ``sizeof(type(obj))``, but it is possible to enlarge the buffer.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 1997)

Unknown directive type "function".

```
.. function:: set_errno(value)
```

Set the current value of the ctypes-private copy of the system :data:`errno` variable in the calling thread to *value* and return the previous value.

```
.. audit-event:: ctypes.set_errno errno ctypes.set_errno
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2005)

Unknown directive type "function".

```
.. function:: set_last_error(value)
```

Windows only: set the current value of the ctypes-private copy of the system :data:`LastError` variable in the calling thread to *value* and return the previous value.

```
.. audit-event:: ctypes.set_last_error error ctypes.set_last_error
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2014)

Unknown directive type "function".

```
.. function:: sizeof(obj_or_type)
```

Returns the size in bytes of a ctypes type or instance memory buffer. Does the same as the C ``sizeof`` operator.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2020)

Unknown directive type "function".

```
.. function:: string_at(address, size=-1)
```

This function returns the C string starting at memory address *address* as a bytes object. If size is specified, it is used as size, otherwise the string is assumed to be zero-terminated.

```
.. audit-event:: ctypes.string_at address,size ctypes.string_at
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2029)

Unknown directive type "function".

```
.. function:: WinError(code=None, descr=None)
```

Windows only: this function is probably the worst-named thing in ctypes. It creates an instance of OSError. If **code** is not specified, `GetLastError` is called to determine the error code. If **descr** is not specified, `FormatError` is called to get a textual description of the error.

```
.. versionchanged:: 3.3
   An instance of :exc:`WindowsError` used to be created.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2041)

Unknown directive type "function".

```
.. function:: wstring_at(address, size=-1)
```

This function returns the wide character string starting at memory address **address** as a string. If **size** is specified, it is used as the number of characters of the string, otherwise the string is assumed to be zero-terminated.

```
.. audit-event:: ctypes.wstring_at address,size ctypes.wstring_at
```

Data types

This non-public class is the common base class of all ctypes data types. Among other things, all ctypes type instances contain a memory block that hold C compatible data; the address of the memory block is returned by the `func:addressof` helper function. Another instance variable is exposed as `attr: _objects`; this contains other Python objects that need to be kept alive in case the memory block contains pointers.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2059); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2059); [backlink](#)

Unknown interpreted text role "attr".

Common methods of ctypes data types, these are all class methods (to be exact, they are methods of the `term:metaclass`):

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2066); [backlink](#)

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2069)

Unknown directive type "method".

```
.. method:: _CData.from_buffer(source[, offset])
```

This method returns a ctypes instance that shares the buffer of the **source** object. The **source** object must support the writeable buffer interface. The optional **offset** parameter specifies an offset into the source buffer in bytes; the default is zero. If the source buffer is not large enough a `exc:ValueError` is raised.

```
.. audit-event:: ctypes.cdata/buffer pointer,size,offset ctypes._CData.from_buffer
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 2079)

Unknown directive type "method".

```
.. method:: _CData.from_buffer_copy(source[, offset])
```

This method creates a ctypes instance, copying the buffer from the *source* object buffer which must be readable. The optional *offset* parameter specifies an offset into the source buffer in bytes; the default is zero. If the source buffer is not large enough a :exc:`ValueError` is raised.

```
.. audit-event:: ctypes.cdata/buffer pointer,size,offset ctypes._CData.from_buffer_copy
```

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

Unknown directive type "method".

```
.. method:: from_address(address)
```

This method returns a ctypes type instance using the memory specified by *address* which must be an integer.

```
.. audit-event:: ctypes.cdata address ctypes._CData.from_address
```

This method, and others that indirectly call this method, raises an :ref:`auditing event <auditing>` ``ctypes.cdata`` with argument ``address``.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 2100)

Unknown directive type "method".

```
.. method:: from_param(obj)
```

This method adapts *obj* to a ctypes type. It is called with the actual object used in a foreign function call when the type is present in the foreign function's :attr:`argtypes` tuple; it must return an object that can be used as a function call parameter.

All ctypes data types have a default implementation of this classmethod that normally returns *obj* if that is an instance of the type. Some types accept other objects as well.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main [Doc] [library]ctypes.rst, line 2111)

Unknown directive type "method".

```
.. method:: in_dll(library, name)
```

This method returns a ctypes type instance exported by a shared library. *name* is the name of the symbol that exports the data, *library* is the loaded shared library.

Common instance variables of ctypes data types:

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

Unknown directive type "attribute".

```
.. attribute:: _b_base_
```

Sometimes ctypes data instances do not own the memory block they contain, instead they share part of the memory block of a base object. The :attr:`_b_base_` read-only member is the root ctypes object that owns the memory block.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2126)

Unknown directive type "attribute".

```
.. attribute:: _b_needsfree_
```

This read-only variable is true when the ctypes data instance has allocated the memory block itself, false otherwise.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2131)

Unknown directive type "attribute".

```
.. attribute:: _objects
```

This member is either ``None`` or a dictionary containing Python objects that need to be kept alive so that the memory block contents is kept valid. This object is only exposed for debugging; never modify the contents of this dictionary.

Fundamental data types

This non-public class is the base class of all fundamental ctypes data types. It is mentioned here because it contains the common attributes of the fundamental ctypes data types. `:class: '_SimpleCData'` is a subclass of `:class: '_CData'`, so it inherits their methods and attributes. ctypes data types that are not and do not contain pointers can now be pickled.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2146); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2146); [backlink](#)

Unknown interpreted text role "class".

Instances have a single attribute:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2154)

Unknown directive type "attribute".

```
.. attribute:: value
```

This attribute contains the actual value of the instance. For integer and pointer types, it is an integer, for character types, it is a single character bytes object or string, for character pointer types it is a Python bytes object or string.

When the ``value`` attribute is retrieved from a ctypes instance, usually a new object is returned each time. `:mod: 'ctypes'` does *not* implement original object return, always a new object is constructed. The same is true for all other ctypes object instances.

Fundamental data types, when returned as foreign function call results, or, for example, by retrieving structure field members or array items, are transparently converted to native Python types. In other words, if a foreign function has a `:attr: 'restype'` of `:class: 'c_char_p'`, you will always receive a Python bytes object, *not* a `:class: 'c_char_p'` instance.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2167); [backlink](#)

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2167); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2167); [backlink](#)

Unknown interpreted text role "class".

Subclasses of fundamental data types do *not* inherit this behavior. So, if a foreign functions `:attr:'restype'` is a subclass of `:class:'c_void_p'`, you will receive an instance of this subclass from the function call. Of course, you can get the value of the pointer by accessing the `value` attribute.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2175); [backlink](#)

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2175); [backlink](#)

Unknown interpreted text role "class".

These are the fundamental ctypes data types:

Represents the C `:type:'signed char'` datatype, and interprets the value as small integer. The constructor accepts an optional integer initializer; no overflow checking is done.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2184); [backlink](#)

Unknown interpreted text role "c:type".

Represents the C `:type:'char'` datatype, and interprets the value as a single character. The constructor accepts an optional string initializer, the length of the string must be exactly one character.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2191); [backlink](#)

Unknown interpreted text role "c:type".

Represents the C `:type:'char *'` datatype when it points to a zero-terminated string. For a general character pointer that may also point to binary data, `POINTER(c_char)` must be used. The constructor accepts an integer address, or a bytes object.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2198); [backlink](#)

Unknown interpreted text role "c:type".

Represents the C `:type:'double'` datatype. The constructor accepts an optional float initializer.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2206); [backlink](#)

Unknown interpreted text role "c:type".

Represents the C `:type:'long double'` datatype. The constructor accepts an optional float initializer. On platforms where `sizeof(long double) == sizeof(double)` it is an alias to `:class:'c_double'`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2212); [backlink](#)

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2212); [backlink](#)

Unknown interpreted text role "class".

Represents the C `:type:'float'` datatype. The constructor accepts an optional float initializer.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2218); [backlink](#)

Unknown interpreted text role "c.type".

Represents the C `:type:'signed int'` datatype. The constructor accepts an optional integer initializer; no overflow checking is done. On platforms where `sizeof(int) == sizeof(long)` it is an alias to `:class:'c_long'`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2224); [backlink](#)

Unknown interpreted text role "c.type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2224); [backlink](#)

Unknown interpreted text role "class".

Represents the C 8-bit `:type:'signed int'` datatype. Usually an alias for `:class:'c_byte'`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2231); [backlink](#)

Unknown interpreted text role "c.type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2231); [backlink](#)

Unknown interpreted text role "class".

Represents the C 16-bit `:type:'signed int'` datatype. Usually an alias for `:class:'c_short'`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2237); [backlink](#)

Unknown interpreted text role "c.type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2237); [backlink](#)

Unknown interpreted text role "class".

Represents the C 32-bit `:type:'signed int'` datatype. Usually an alias for `:class:'c_int'`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2243); [backlink](#)

Unknown interpreted text role "c.type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2243); [backlink](#)

Unknown interpreted text role "class".

Represents the C 64-bit `:type:'signed int'` datatype. Usually an alias for `:class:'c_longlong'`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2249); [backlink](#)

Unknown interpreted text role "c.type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2249); [backlink](#)

Unknown interpreted text role "class".

Represents the C `:type:'signed long'` datatype. The constructor accepts an optional integer initializer; no overflow checking is done.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2255); [backlink](#)

Unknown interpreted text role "c.type".

Represents the C `:type:'signed long long'` datatype. The constructor accepts an optional integer initializer; no overflow checking is done.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2261); [backlink](#)

Unknown interpreted text role "c.type".

Represents the C `:type:'signed short'` datatype. The constructor accepts an optional integer initializer; no overflow checking is done.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2267); [backlink](#)

Unknown interpreted text role "c.type".

Represents the C `:type:'size_t'` datatype.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2273); [backlink](#)

Unknown interpreted text role "c.type".

Represents the C `:type:'ssize_t'` datatype.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2278); [backlink](#)

Unknown interpreted text role "c.type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2280)

Unknown directive type "versionadded".

```
.. versionadded:: 3.2
```

Represents the C `:type:'unsigned char'` datatype, it interprets the value as small integer. The constructor accepts an optional integer initializer; no overflow checking is done.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2285); [backlink](#)

Unknown interpreted text role "c.type".

Represents the C `:type:'unsigned int'` datatype. The constructor accepts an optional integer initializer; no overflow checking is done. On platforms where `sizeof(int) == sizeof(long)` it is an alias for `:class:'c_ulong'`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2292); [backlink](#)

Unknown interpreted text role "c.type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library]ctypes.rst, line 2292); [backlink](#)

Unknown interpreted text role "class".

Represents the C 8-bit `:type:'unsigned int'` datatype. Usually an alias for `:class:'c_ubyte'`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2299); [backlink](#)

Unknown interpreted text role "c.type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2299); [backlink](#)

Unknown interpreted text role "class".

Represents the C 16-bit `:type:'unsigned int'` datatype. Usually an alias for `:class:'c_ushort'`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2305); [backlink](#)

Unknown interpreted text role "c.type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2305); [backlink](#)

Unknown interpreted text role "class".

Represents the C 32-bit `:type:'unsigned int'` datatype. Usually an alias for `:class:'c_uint'`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2311); [backlink](#)

Unknown interpreted text role "c.type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2311); [backlink](#)

Unknown interpreted text role "class".

Represents the C 64-bit `:type:'unsigned int'` datatype. Usually an alias for `:class:'c_ulonglong'`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2317); [backlink](#)

Unknown interpreted text role "c.type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2317); [backlink](#)

Unknown interpreted text role "class".

Represents the C `:type:'unsigned long'` datatype. The constructor accepts an optional integer initializer; no overflow checking is done.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2323); [backlink](#)

Unknown interpreted text role "c.type".

Represents the C `:type:'unsigned long long'` datatype. The constructor accepts an optional integer initializer; no overflow checking is done.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2329); [backlink](#)

Unknown interpreted text role "c.type".

Represents the C `:type:'unsigned short'` datatype. The constructor accepts an optional integer initializer; no overflow checking is done.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2335); [backlink](#)

Unknown interpreted text role "c.type".

Represents the C `:type:'void *'` type. The value is represented as integer. The constructor accepts an optional integer initializer.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2341); [backlink](#)

Unknown interpreted text role "c.type".

Represents the C `:type:'wchar_t'` datatype, and interprets the value as a single character unicode string. The constructor accepts an optional string initializer, the length of the string must be exactly one character.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2347); [backlink](#)

Unknown interpreted text role "c.type".

Represents the C `:type:'wchar_t *'` datatype, which must be a pointer to a zero-terminated wide character string. The constructor accepts an integer address, or a string.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2354); [backlink](#)

Unknown interpreted text role "c.type".

Represent the C `:type:'bool'` datatype (more accurately, `:type:'_Bool'` from C99). Its value can be `True` or `False`, and the constructor accepts any object that has a truth value.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2361); [backlink](#)

Unknown interpreted text role "c.type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2361); [backlink](#)

Unknown interpreted text role "c.type".

Windows only: Represents a `:type:'HRESULT'` value, which contains success or error information for a function or method call.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2368); [backlink](#)

Unknown interpreted text role "c.type".

Represents the C `:type:'PyObject *'` datatype. Calling this without an argument creates a `NULL :type:'PyObject *'` pointer.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2374); [backlink](#)

Unknown interpreted text role "c.type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2374); [backlink](#)

Unknown interpreted text role "c.type".

The `:mod:'ctypes.wintypes'` module provides quite some other Windows specific data types, for example `:type:'HWND'`, `:type:'WPARAM'`, or `:type:'DWORD'`. Some useful structures like `:type:'MSG'` or `:type:'RECT'` are also defined.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2377); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2377); [backlink](#)

Unknown interpreted text role "c.type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2377); [backlink](#)

Unknown interpreted text role "c.type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2377); [backlink](#)

Unknown interpreted text role "c.type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2377); [backlink](#)

Unknown interpreted text role "c.type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2377); [backlink](#)

Unknown interpreted text role "c.type".

Structured data types

Abstract base class for unions in native byte order.

Abstract base class for unions in *big endian* byte order.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2397)

Unknown directive type "versionadded".

```
.. versionadded:: 3.11
```

Abstract base class for unions in *little endian* byte order.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2403)

Unknown directive type "versionadded".

```
.. versionadded:: 3.11
```

Abstract base class for structures in *big endian* byte order.

Abstract base class for structures in *little endian* byte order.

Structures and unions with non-native byte order cannot contain pointer type fields, or any other data types containing pointer type fields.

Abstract base class for structures in *native* byte order.

Concrete structure and union types must be created by subclassing one of these types, and at least define a `attr: _fields_` class variable. `mod: ctypes` will create `term: descriptor`'s which allow reading and writing the fields by direct attribute accesses. These are the

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2422); [backlink](#)

Unknown interpreted text role "attr".

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

main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 2422); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 2422); [backlink](#)

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 2428)

Unknown directive type "attribute".

```
.. attribute:: _fields_
```

A sequence defining the structure fields. The items must be 2-tuples or 3-tuples. The first item is the name of the field, the second item specifies the type of the field; it can be any ctypes data type.

For integer type fields like :class:`c_int`, a third optional item can be given. It must be a small positive integer defining the bit width of the field.

Field names must be unique within one structure or union. This is not checked, only one field can be accessed when names are repeated.

It is possible to define the :attr:`_fields_` class variable *after* the class statement that defines the Structure subclass, this allows creating data types that directly or indirectly reference themselves::

```
class List(Structure):
    pass
List._fields_ = [("pNext", POINTER(List)),
    ...
]
```

The :attr:`_fields_` class variable must, however, be defined before the type is first used (an instance is created, :func:`sizeof` is called on it, and so on). Later assignments to the :attr:`_fields_` class variable will raise an `AttributeError`.

It is possible to define sub-subclasses of structure types, they inherit the fields of the base class plus the :attr:`_fields_` defined in the sub-subclass, if any.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 2461)

Unknown directive type "attribute".

```
.. attribute:: _pack_
```

An optional small integer that allows overriding the alignment of structure fields in the instance. :attr:`_pack_` must already be defined when :attr:`_fields_` is assigned, otherwise it will have no effect.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library]ctypes.rst, line 2468)

Unknown directive type "attribute".

```
.. attribute:: _anonymous_
```

An optional sequence that lists the names of unnamed (anonymous) fields. :attr:`_anonymous_` must be already defined when :attr:`_fields_` is assigned, otherwise it will have no effect.

The fields listed in this variable must be structure or union type fields. :mod:`ctypes` will create descriptors in the structure type that allows accessing the nested fields directly, without the need to create the structure or union field.

Here is an example type (Windows)::


```

class _U(Union):
    _fields_ = [("lptdesc", POINTER(TYPEDESC)),
                ("lpadesc", POINTER(ARRAYDESC)),
                ("hreftype", HREFTYPE)]

class TYPEDESC(Structure):
    _anonymous_ = ("u",)
    _fields_ = [("u", _U),
                ("vt", VARTYPE)]

```

The ``TYPEDESC`` structure describes a COM data type, the ``vt`` field specifies which one of the union fields is valid. Since the ``u`` field is defined as anonymous field, it is now possible to access the members directly off the TYPEDESC instance. ``td.lptdesc`` and ``td.u.lptdesc`` are equivalent, but the former is faster since it does not need to create a temporary union instance::

```

td = TYPEDESC()
td.vt = VT_PTR
td.lptdesc = POINTER(some_type)
td.u.lptdesc = POINTER(some_type)

```

It is possible to define sub-subclasses of structures, they inherit the fields of the base class. If the subclass definition has a separate `attr: _fields_` variable, the fields specified in this are appended to the fields of the base class.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 2504); [backlink](#)

Unknown interpreted text role "attr".

Structure and union constructors accept both positional and keyword arguments. Positional arguments are used to initialize member fields in the same order as they appear in `attr: _fields_`. Keyword arguments in the constructor are interpreted as attribute assignments, so they will initialize `attr: _fields_` with the same name, or create new attributes for names not present in `attr: _fields_`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 2509); [backlink](#)

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 2509); [backlink](#)

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 2509); [backlink](#)

Unknown interpreted text role "attr".

Arrays and pointers

Abstract base class for arrays.

The recommended way to create concrete array types is by multiplying any `mod: ctypes` data type with a non-negative integer. Alternatively, you can subclass this type and define `attr: _length_` and `attr: _type_` class variables. Array elements can be read and written using standard subscript and slice accesses; for slice reads, the resulting object is *not* itself an `class: Array`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 2526); [backlink](#)

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes.rst, line 2526); [backlink](#)

Unknown interpreted text role "attr".

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

main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 2526); [backlink](#)

Unknown interpreted text role "attr".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 2526); [backlink](#)

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 2534)

Unknown directive type "attribute".

```
.. attribute:: _length_
```

A positive integer specifying the number of elements in the array.
Out-of-range subscripts result in an :exc:`IndexError`. Will be
returned by :func:`len`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 2541)

Unknown directive type "attribute".

```
.. attribute:: _type_
```

Specifies the type of each element in the array.

Array subclass constructors accept positional arguments, used to initialize the elements in order.

Private, abstract base class for pointers.

Concrete pointer types are created by calling :func:`POINTER` with the type that will be pointed to; this is done automatically by :func:`pointer`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 2554); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 2554); [backlink](#)

Unknown interpreted text role "func".

If a pointer points to an array, its elements can be read and written using standard subscript and slice accesses. Pointer objects have no size, so :func:`len` will raise :exc:`TypeError`. Negative subscripts will read from the memory *before* the pointer (as in C), and out-of-range subscripts will probably crash with an access violation (if you're lucky).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 2558); [backlink](#)

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 2558); [backlink](#)

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ [cpython-main] [Doc] [library] ctypes .rst, line 2566)

Unknown directive type "attribute".

```
.. attribute:: _type_
```

Specifies the type pointed to.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main] [Doc] [library] ctypes.rst, line 2570)

Unknown directive type "attribute".

```
.. attribute:: contents
```

Returns the object to which to pointer points. Assigning to this attribute changes the pointer to point to the assigned object.