Capsules

Refer to Providing a C API for an Extension Module for more information on using these objects.

New in version 3.1.

PyCapsule

This subtype of `PyObject` represents an opaque value, useful for C extension modules who need to pass an opaque value (as a `void*` pointer) through Python code to other C code. It is often used to make a C function pointer defined in one module available to other modules, so the regular import mechanism can be used to access C APIs defined in dynamically loaded modules.

PyCapsule_Destructor

The type of a destructor callback for a capsule. Defined as:

```
typedef void (*PyCapsule_Destructor)(PyObject *);
```

See `PyCapsule_New()` for the semantics of PyCapsule_Destructor callbacks.

int **PyCapsule_CheckExact**(PyObject *p)

Return true if its argument is a `PyCapsule`.

PyObject* **PyCapsule_New**(void *pointer, const char *name, PyCapsule_Destructor destructor)

Return value: New reference.

Create a `PyCapsule` encapsulating the *pointer*. The *pointer* argument may not be *NULL*.

On failure, set an exception and return *NULL*.

The *name* string may either be *NULL* or a pointer to a valid C string. If non-*NULL*, this string must outlive the capsule. (Though it is permitted to free it inside the *destructor*.)

If the *destructor* argument is not *NULL*, it will be called with the capsule as its argument when it is destroyed.

If this capsule will be stored as an attribute of a module, the *name* should be specified as `modulename.attributename`. This will enable other modules to import the capsule using `PyCapsule_Import()`.

void* **PyCapsule_GetPointer**(PyObject *capsule, const char *name)

Retrieve the *pointer* stored in the capsule. On failure, set an exception and return *NULL*.

The *name* parameter must compare exactly to the name stored in the capsule. If the name stored in the capsule is *NULL*, the *name* passed in must also be *NULL*. Python uses the C function `strcmp()` to compare capsule names.

PyCapsule_Destructor **PyCapsule_GetDestructor**(PyObject **capsule*)

Return the current destructor stored in the capsule. On failure, set an exception and return *NULL*.

It is legal for a capsule to have a *NULL* destructor. This makes a *NULL* return code somewhat ambiguous; use `PyCapsule_IsValid()` or `PyErr_Occurred()` to disambiguate.

void* **PyCapsule_GetContext**(PyObject **capsule*)

Return the current context stored in the capsule. On failure, set an exception and return *NULL*.

It is legal for a capsule to have a *NULL* context. This makes a *NULL* return code somewhat ambiguous; use `PyCapsule_IsValid()` or `PyErr_Occurred()` to disambiguate.

const char* **PyCapsule_GetName**(PyObject **capsule*)

Return the current name stored in the capsule. On failure, set an exception and return *NULL*.

It is legal for a capsule to have a *NULL* name. This makes a *NULL* return code somewhat ambiguous; use `PyCapsule_IsValid()` or `PyErr_Occurred()` to disambiguate.

void* **PyCapsule_Import**(const char **name*, int *no_block*)

Import a pointer to a C object from a capsule attribute in a module. The *name* parameter should specify the full name to the attribute, as in `module.attribute`. The *name* stored in the capsule must match this string exactly. If *no_block* is true, import the module without blocking (using `PyImport_ImportModuleNoBlock()`). If *no_block* is false, import the module conventionally (using `PyImport_ImportModule()`).

Return the capsule's internal *pointer* on success. On failure, set an exception and return *NULL*.

int **PyCapsule_IsValid**(PyObject **capsule*, const char **name*)

Determines whether or not *capsule* is a valid capsule. A valid capsule is non-*NULL*, passes `PyCapsule_CheckExact()`, has a non-*NULL* pointer stored in it,

and its internal name matches the *name* parameter. (See `PyCapsule_GetPointer()` for information on how capsule names are compared.)

In other words, if `PyCapsule_IsValid()` returns a true value, calls to any of the accessors (any function starting with `PyCapsule_Get()`) are guaranteed to succeed.

Return a nonzero value if the object is valid and matches the name passed in. Return `0` otherwise. This function will not fail.

int **PyCapsule_SetContext**(PyObject **capsule*, void **context*)

Set the context pointer inside *capsule* to *context*.

Return `0` on success. Return nonzero and set an exception on failure.

int **PyCapsule_SetDestructor**(PyObject **capsule*, PyCapsule_Destructor *destructor*)

Set the destructor inside *capsule* to *destructor*.

Return `0` on success. Return nonzero and set an exception on failure.

int **PyCapsule_SetName**(PyObject **capsule*, const char **name*)

Set the name inside *capsule* to *name*. If non-*NULL*, the name must outlive the capsule. If the previous *name* stored in the capsule was not *NULL*, no attempt is made to free it.

Return `0` on success. Return nonzero and set an exception on failure.

int **PyCapsule_SetPointer**(PyObject **capsule*, void **pointer*)

Set the void pointer inside *capsule* to *pointer*. The pointer may not be *NULL*.

Return `0` on success. Return nonzero and set an exception on failure.