Type Objects

PyTypeObject

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

PyObject* **PyType_Type**

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

int **PyType_Check**(PyObject *o)

Return true if the object *o* is a type object, including instances of types derived from the standard type object. Return false in all other cases.

int **PyType_CheckExact**(PyObject *o)

Return true if the object *o* is a type object, but not a subtype of the standard type object. Return false in all other cases.

unsigned int **PyType_ClearCache**()

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

long **PyType_GetFlags**(PyTypeObject* *type*)

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

New in version 3.2.

void **PyType_Modified**(PyTypeObject *type)

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

int **PyType_HasFeature**(PyTypeObject *o, int *feature*)

Return true if the type object *o* sets the feature *feature*. Type features are denoted by single bit flags.

int **PyType_IS_GC**(PyTypeObject *o)

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

int **PyType_IsSubtype**(PyTypeObject *a, PyTypeObject *b)

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

This function only checks for actual subtypes, which means that `__subclasscheck__()` is not called on *b*. Call `PyObject_IsSubclass()` to do the same check that `issubclass()` would do.

PyObject* **PyType_GenericAlloc**(PyTypeObject **type*, Py_ssize_t *nitems*)

Return value: New reference.

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

PyObject* **PyType_GenericNew**(PyTypeObject **type*, PyObject **args*, PyObject **kwds*)

Return value: New reference.

Generic handler for the `tp_new` slot of a type object. Create a new instance using the type's `tp_alloc` slot.

int **PyType_Ready**(PyTypeObject **type*)

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

PyObject* **PyType_FromSpec**(PyType_Spec **spec*)

Creates and returns a heap type object from the *spec* passed to the function.

PyObject* **PyType_FromSpecWithBases**(PyType_Spec **spec*, PyObject **bases*)

Creates and returns a heap type object from the *spec*. In addition to that, the created heap type contains all types contained by the *bases* tuple as base types. This allows the caller to reference other heap types as base types.

New in version 3.3.

void* **PyType_GetSlot**(PyTypeObject **type*, int *slot*)

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

New in version 3.4.