

Reference Counting

The macros in this section are used for managing reference counts of Python objects.

void **Py_INCREF**(PyObject *o)

Increment the reference count for object *o*. The object must not be *NULL*; if you aren't sure that it isn't *NULL*, use `Py_XINCREF()`.

void **Py_XINCREF**(PyObject *o)

Increment the reference count for object *o*. The object may be *NULL*, in which case the macro has no effect.

void **Py_DECREF**(PyObject *o)

Decrement the reference count for object *o*. The object must not be *NULL*; if you aren't sure that it isn't *NULL*, use `Py_XDECREF()`. If the reference count reaches zero, the object's type's deallocation function (which must not be *NULL*) is invoked.

Warning: The deallocation function can cause arbitrary Python code to be invoked (e.g. when a class instance with a `__del__()` method is deallocated). While exceptions in such code are not propagated, the executed code has free access to all Python global variables. This means that any object that is reachable from a global variable should be in a consistent state before `Py_DECREF()` is invoked. For example, code to delete an object from a list should copy a reference to the deleted object in a temporary variable, update the list data structure, and then call `Py_DECREF()` for the temporary variable.

void **Py_XDECREF**(PyObject *o)

Decrement the reference count for object *o*. The object may be *NULL*, in which case the macro has no effect; otherwise the effect is the same as for `Py_DECREF()`, and the same warning applies.

void **Py_CLEAR**(PyObject *o)

Decrement the reference count for object *o*. The object may be *NULL*, in which case the macro has no effect; otherwise the effect is the same as for `Py_DECREF()`, except that the argument is also set to *NULL*. The warning for `Py_DECREF()` does not apply with respect to the object passed because the macro carefully uses a temporary variable and sets the argument to *NULL* before decrementing its reference count.

It is a good idea to use this macro whenever decrementing the value of a variable that might be traversed during garbage collection.

The following functions are for runtime dynamic embedding of Python: `Py_IncRef(PyObject *o)`, `Py_DecRef(PyObject *o)`. They are simply exported function versions of `Py_XINCRF()` and `Py_XDECREF()`, respectively.

The following functions or macros are only for use within the interpreter core: `_Py_Dealloc()`, `_Py_ForgetReference()`, `_Py_NewReference()`, as well as the global variable `_Py_RefTotal`.