Weak Reference Objects

Python supports *weak references* as first-class objects. There are two specific object types which directly implement weak references. The first is a simple reference object, and the second acts as a proxy for the original object as much as it can.

int **`PyWeakref_Check`**(ob)

Return true if *ob* is either a reference or proxy object.

int **`PyWeakref_CheckRef`**(ob)

Return true if *ob* is a reference object.

int **`PyWeakref_CheckProxy`**(ob)

Return true if *ob* is a proxy object.

PyObject* **`PyWeakref_NewRef`**(PyObject **ob*, PyObject **callback*)

Return value: New reference.

Return a weak reference object for the object *ob*. This will always return a new reference, but is not guaranteed to create a new object; an existing reference object may be returned. The second parameter, *callback*, can be a callable object that receives notification when *ob* is garbage collected; it should accept a single parameter, which will be the weak reference object itself. *callback* may also be `None` or *NULL*. If *ob* is not a weakly-referencable object, or if *callback* is not callable, `None`, or *NULL*, this will return *NULL* and raise `TypeError`.

PyObject* **`PyWeakref_NewProxy`**(PyObject **ob*, PyObject **callback*)

Return value: New reference.

Return a weak reference proxy object for the object *ob*. This will always return a new reference, but is not guaranteed to create a new object; an existing proxy object may be returned. The second parameter, *callback*, can be a callable object that receives notification when *ob* is garbage collected; it should accept a single parameter, which will be the weak reference object itself. *callback* may also be `None` or *NULL*. If *ob* is not a weakly-referencable object, or if *callback* is not callable, `None`, or *NULL*, this will return *NULL* and raise `TypeError`.

PyObject* **`PyWeakref_GetObject`**(PyObject **ref*)

Return value: Borrowed reference.

Return the referenced object from a weak reference, *ref*. If the referent is no longer live, returns `Py_None`.

> **Note:** This function returns a **borrowed reference** to the referenced object. This means that you should always call `Py_INCREF()` on the object except if you know that it cannot be destroyed while you are still using it.

PyObject* **PyWeakref_GET_OBJECT**(PyObject **ref*)

Return value: Borrowed reference.

Similar to `PyWeakref_GetObject()`, but implemented as a macro that does no error checking.