New in version 3.4.

class `weakref.`**`finalize`**(*obj*,*func*, **args*, ***kwargs*)

Return a callable finalizer object which will be called when *obj* is garbage collected. Unlike an ordinary weak reference, a finalizer will always survive until the reference object is collected, greatly simplifying lifecycle management.

A finalizer is considered *alive*until it is called (either explicitly or at garbage collection), and after that it is *dead*. Calling a live finalizer returns the result of evaluating `func(*arg,**kwargs)`, whereas calling a dead finalizer returns `None`.

Exceptions raised by finalizer callbacks during garbage collection will be shown on the standard error output, but cannot be propagated. They are handled in the same way as exceptions raised from an object's `__del__()` method or a weak reference's callback.

When the program exits, each remaining live finalizer is called unless its `atexit` attribute has been set to false. They are called in reverse order of creation.

A finalizer will never invoke its callback during the later part of the interpreter shutdown when module globals are liable to have been replaced by `None`.

`__call__`()

If *self* is alive then mark it as dead and return the result of calling`func(*args, **kwargs)`. If *self* is dead then return`None`.

`detach`()

If *self* is alive then mark it as dead and return the tuple `(obj, func, args,kwargs)`. If *self* is dead then return `None`.

`peek`()

If *self* is alive then return the tuple `(obj, func,args, kwargs)`. If *self* is dead then return `None`.

`alive`

Property which is true if the finalizer is alive, false otherwise.

`atexit`

A writable boolean property which by default is true. When the program exits, it calls all remaining live finalizers for which `atexit`is true. They are called in reverse order of creation.

> **Note** It is important to ensure that *func*, *args* and*kwargs* do not own any references to *obj*, either directly or indirectly, since otherwise *obj* will never be garbage collected. In particular, *func* should not be a bound method of *obj*.