:mod:`gc` --- Garbage Collector interface

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 1); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 4)

Unknown directive type "module".

```
.. module:: gc
   :synopsis: Interface to the cycle-detecting garbage collector.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 7)

Unknown directive type "moduleauthor".

```
.. moduleauthor:: Neil Schemenauer <nas@arctrix.com>
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 8)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Neil Schemenauer <nas@arctrix.com>
```

This module provides an interface to the optional garbage collector. It provides the ability to disable the collector, tune the collection frequency, and set debugging options. It also provides access to unreachable objects that the collector found but cannot free. Since the collector supplements the reference counting already used in Python, you can disable the collector if you are sure your program does not create reference cycles. Automatic collection can be disabled by calling `gc.disable()`. To debug a leaking program call `gc.set_debug(gc.DEBUG_LEAK)`. Notice that this includes `gc.DEBUG_SAVEALL`, causing garbage-collected objects to be saved in gc.garbage for inspection.

The :mod:`gc` module provides the following functions:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 23); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 26)

Unknown directive type "function".

```
.. function:: enable()

   Enable automatic garbage collection.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 31)

Unknown directive type "function".

```
.. function:: disable()

   Disable automatic garbage collection.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 36)

Unknown directive type "function".

```
.. function:: isenabled()

   Return ``True`` if automatic collection is enabled.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 41)

Unknown directive type "function".

```
.. function:: collect(generation=2)

   With no arguments, run a full collection.  The optional argument *generation*
   may be an integer specifying which generation to collect (from 0 to 2).  A
   :exc:`ValueError` is raised if the generation number  is invalid. The number of
   unreachable objects found is returned.

   The free lists maintained for a number of built-in types are cleared
   whenever a full collection or collection of the highest generation (2)
   is run.  Not all items in some free lists may be freed due to the
   particular implementation, in particular :class:`float`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 54)

Unknown directive type "function".

```
.. function:: set_debug(flags)

   Set the garbage collection debugging flags. Debugging information will be
   written to ``sys.stderr``.  See below for a list of debugging flags which can be
   combined using bit operations to control debugging.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 61)

Unknown directive type "function".

```
.. function:: get_debug()

   Return the debugging flags currently set.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 66)

Unknown directive type "function".

```
.. function:: get_objects(generation=None)

   Returns a list of all objects tracked by the collector, excluding the list
   returned. If *generation* is not None, return only the objects tracked by
   the collector that are in that generation.

   .. versionchanged:: 3.8
      New *generation* parameter.

   .. audit-event:: gc.get_objects generation gc.get_objects
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 77)

Unknown directive type "function".

```
.. function:: get_stats()

   Return a list of three per-generation dictionaries containing collection
   statistics since interpreter start.  The number of keys may change
   in the future, but currently each dictionary will contain the following
   items:

   * ``collections`` is the number of times this generation was collected;

   * ``collected`` is the total number of objects collected inside this
     generation;

   * ``uncollectable`` is the total number of objects which were found
     to be uncollectable (and were therefore moved to the :data:`garbage`
     list) inside this generation.

   .. versionadded:: 3.4
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 96)

Unknown directive type "function".

```
.. function:: set_threshold(threshold0[, threshold1[, threshold2]])

   Set the garbage collection thresholds (the collection frequency). Setting
   *threshold0* to zero disables collection.

   The GC classifies objects into three generations depending on how many
   collection sweeps they have survived.  New objects are placed in the youngest
   generation (generation ``0``).  If an object survives a collection it is moved
   into the next older generation.  Since generation ``2`` is the oldest
   generation, objects in that generation remain there after a collection.  In
   order to decide when to run, the collector keeps track of the number object
   allocations and deallocations since the last collection.  When the number of
   allocations minus the number of deallocations exceeds *threshold0*, collection
   starts.  Initially only generation ``0`` is examined.  If generation ``0`` has
   been examined more than *threshold1* times since generation ``1`` has been
   examined, then generation ``1`` is examined as well.
   With the third generation, things are a bit more complicated,
   see `Collecting the oldest generation <https://devguide.python.org/garbage_collector/#collecting-the-oldest-generation>
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 116)

Unknown directive type "function".

```
.. function:: get_count()

   Return the current collection  counts as a tuple of ``(count0, count1,
   count2)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 122)

Unknown directive type "function".

```
.. function:: get_threshold()

   Return the current collection thresholds as a tuple of ``(threshold0,
   threshold1, threshold2)``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 128)

Unknown directive type "function".

```
.. function:: get_referrers(*objs)

   Return the list of objects that directly refer to any of objs. This function
   will only locate those containers which support garbage collection; extension
   types which do refer to other objects but do not support garbage collection will
   not be found.

   Note that objects which have already been dereferenced, but which live in cycles
   and have not yet been collected by the garbage collector can be listed among the
   resulting referrers.  To get only currently live objects, call :func:`collect`
   before calling :func:`get_referrers`.

   .. warning::
      Care must be taken when using objects returned by :func:`get_referrers` because
      some of them could still be under construction and hence in a temporarily
      invalid state. Avoid using :func:`get_referrers` for any purpose other than
      debugging.

   .. audit-event:: gc.get_referrers objs gc.get_referrers
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 149)

Unknown directive type "function".

```
.. function:: get_referents(*objs)

   Return a list of objects directly referred to by any of the arguments. The
   referents returned are those objects visited by the arguments' C-level
   :c:member:`~PyTypeObject.tp_traverse` methods (if any), and may not be all objects actually
   directly reachable.  :c:member:`~PyTypeObject.tp_traverse` methods are supported only by objects
   that support garbage collection, and are only required to visit objects that may
   be involved in a cycle.  So, for example, if an integer is directly reachable
   from an argument, that integer object may or may not appear in the result list.

   .. audit-event:: gc.get_referents objs gc.get_referents
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 161)

Unknown directive type "function".

```
.. function:: is_tracked(obj)

   Returns ``True`` if the object is currently tracked by the garbage collector,
   ``False`` otherwise.  As a general rule, instances of atomic types aren't
   tracked and instances of non-atomic types (containers, user-defined
   objects...) are.  However, some type-specific optimizations can be present
   in order to suppress the garbage collector footprint of simple instances
   (e.g. dicts containing only atomic keys and values)::

      >>> gc.is_tracked(0)
      False
      >>> gc.is_tracked("a")
      False
      >>> gc.is_tracked([])
      True
      >>> gc.is_tracked({})
      False
      >>> gc.is_tracked({"a": 1})
      False
      >>> gc.is_tracked({"a": []})
      True

   .. versionadded:: 3.1
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 186)

Unknown directive type "function".

```
.. function:: is_finalized(obj)

   Returns ``True`` if the given object has been finalized by the
   garbage collector, ``False`` otherwise. ::

      >>> x = None
      >>> class Lazarus:
      ...     def __del__(self):
      ...         global x
      ...         x = self
      ...
      >>> lazarus = Lazarus()
      >>> gc.is_finalized(lazarus)
      False
      >>> del lazarus
      >>> gc.is_finalized(x)
      True

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 207)

Unknown directive type "function".

```
.. function:: freeze()

   Freeze all the objects tracked by gc - move them to a permanent generation
   and ignore all the future collections. This can be used before a POSIX
   fork() call to make the gc copy-on-write friendly or to speed up collection.
```

```
Also collection before a POSIX fork() call may free pages for future
allocation which can cause copy-on-write too so it's advised to disable gc
in parent process and freeze before fork and enable gc in child process.

.. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)gc.rst, line 219)

Unknown directive type "function".

```
.. function:: unfreeze()

   Unfreeze the objects in the permanent generation, put them back into the
   oldest generation.

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)gc.rst, line 227)

Unknown directive type "function".

```
.. function:: get_freeze_count()

   Return the number of objects in the permanent generation.

   .. versionadded:: 3.7
```

The following variables are provided for read-only access (you can mutate the values but should not rebind them):

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)gc.rst, line 237)

Unknown directive type "data".

```
.. data:: garbage

   A list of objects which the collector found to be unreachable but could
   not be freed (uncollectable objects).  Starting with Python 3.4, this
   list should be empty most of the time, except when using instances of
   C extension types with a non-``NULL`` ``tp_del`` slot.

   If :const:`DEBUG_SAVEALL` is set, then all unreachable objects will be
   added to this list rather than freed.

   .. versionchanged:: 3.2
      If this list is non-empty at :term:`interpreter shutdown`, a
      :exc:`ResourceWarning` is emitted, which is silent by default.  If
      :const:`DEBUG_UNCOLLECTABLE` is set, in addition all uncollectable objects
      are printed.

   .. versionchanged:: 3.4
      Following :pep:`442`, objects with a :meth:`__del__` method don't end
      up in :attr:`gc.garbage` anymore.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)gc.rst, line 257)

Unknown directive type "data".

```
.. data:: callbacks

   A list of callbacks that will be invoked by the garbage collector before and
   after collection.  The callbacks will be called with two arguments,
   *phase* and *info*.

   *phase* can be one of two values:

      "start": The garbage collection is about to start.

      "stop": The garbage collection has finished.

   *info* is a dict providing more information for the callback.  The following
   keys are currently defined:

      "generation": The oldest generation being collected.

      "collected": When *phase* is "stop", the number of objects
      successfully collected.

      "uncollectable": When *phase* is "stop", the number of objects
      that could not be collected and were put in :data:`garbage`.

   Applications can add their own callbacks to this list.  The primary
   use cases are:

      Gathering statistics about garbage collection, such as how often
      various generations are collected, and how long the collection
      takes.

      Allowing applications to identify and clear their own uncollectable
      types when they appear in :data:`garbage`.

   .. versionadded:: 3.3
```

The following constants are provided for use with :func:`set_debug`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)gc.rst, line 293); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 296)

Unknown directive type "data".

```
.. data:: DEBUG_STATS

   Print statistics during collection.  This information can be useful when tuning
   the collection frequency.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 302)

Unknown directive type "data".

```
.. data:: DEBUG_COLLECTABLE

   Print information on collectable objects found.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 307)

Unknown directive type "data".

```
.. data:: DEBUG_UNCOLLECTABLE

   Print information of uncollectable objects found (objects which are not
   reachable but cannot be freed by the collector).  These objects will be added
   to the ``garbage`` list.

   .. versionchanged:: 3.2
      Also print the contents of the :data:`garbage` list at
      :term:`interpreter shutdown`, if it isn't empty.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 317)

Unknown directive type "data".

```
.. data:: DEBUG_SAVEALL

   When set, all unreachable objects found will be appended to *garbage* rather
   than being freed.  This can be useful for debugging a leaking program.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)gc.rst, line 323)

Unknown directive type "data".

```
.. data:: DEBUG_LEAK

   The debugging flags necessary for the collector to print information about a
   leaking program (equal to ``DEBUG_COLLECTABLE | DEBUG_UNCOLLECTABLE |
   DEBUG_SAVEALL``).
```