:mod:`graphlib` --- Functionality to operate with graph-like structures

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

Unknown directive type "module".

```
.. module:: graphlib
   :synopsis: Functionality to operate with graph-like structures
```

Source code: :source:`Lib/graphlib.py`

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)graphlib.rst`, line 8); ***backlink***

Unknown interpreted text role "source".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)graphlib.rst`, line 10)

Unknown directive type "testsetup".

```
.. testsetup:: default

   import graphlib
   from graphlib import *
```

Provides functionality to topologically sort a graph of hashable nodes.

A topological order is a linear ordering of the vertices in a graph such that for every directed edge u -> v from vertex u to vertex v, vertex u comes before vertex v in the ordering. For instance, the vertices of the graph may represent tasks to be performed, and the edges may represent constraints that one task must be performed before another; in this example, a topological ordering is just a valid sequence for the tasks. A complete topological ordering is possible if and only if the graph has no directed cycles, that is, if it is a directed acyclic graph.

If the optional *graph* argument is provided it must be a dictionary representing a directed acyclic graph where the keys are nodes and the values are iterables of all predecessors of that node in the graph (the nodes that have edges that point to the value in the key). Additional nodes can be added to the graph using the :meth:`~TopologicalSorter.add` method.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)graphlib.rst`, line 31); ***backlink***

Unknown interpreted text role "meth".

In the general case, the steps required to perform the sorting of a given graph are as follows:

- Create an instance of the :class:`TopologicalSorter` with an optional initial graph.

  System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)graphlib.rst`, line 40); ***backlink***

  Unknown interpreted text role "class".

- Add additional nodes to the graph.
- Call :meth:`~TopologicalSorter.prepare` on the graph.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)graphlib.rst`, line 43); *backlink*

Unknown interpreted text role "meth".

- While :meth:`~TopologicalSorter.is_active` is `True`, iterate over the nodes returned by :meth:`~TopologicalSorter.get_ready` and process them. Call :meth:`~TopologicalSorter.done` on each node as it finishes processing.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)graphlib.rst`, line 44); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)graphlib.rst`, line 44); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)graphlib.rst`, line 44); *backlink*

Unknown interpreted text role "meth".

In case just an immediate sorting of the nodes in the graph is required and no parallelism is involved, the convenience method :meth:`TopologicalSorter.static_order` can be used directly:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)graphlib.rst`, line 49); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)graphlib.rst`, line 53)

Unknown directive type "doctest".

```
.. doctest::

    >>> graph = {"D": {"B", "C"}, "C": {"A"}, "B": {"A"}}
    >>> ts = TopologicalSorter(graph)
    >>> tuple(ts.static_order())
    ('A', 'C', 'B', 'D')
```

The class is designed to easily support parallel processing of the nodes as they become ready. For instance:

```
topological_sorter = TopologicalSorter()

# Add nodes to 'topological_sorter'...

topological_sorter.prepare()
while topological_sorter.is_active():
    for node in topological_sorter.get_ready():
        # Worker threads or processes take nodes to work on off the
        # 'task_queue' queue.
        task_queue.put(node)

    # When the work for a node is done, workers put the node in
    # 'finalized_tasks_queue' so we can get more nodes to work on.
    # The definition of 'is_active()' guarantees that, at this point, at
    # least one node has been placed on 'task_queue' that hasn't yet
    # been passed to 'done()', so this blocking 'get()' must (eventually)
    # succeed.  After calling 'done()', we loop back to call 'get_ready()'
    # again, so put newly freed nodes on 'task_queue' as soon as
    # logically possible.
    node = finalized_tasks_queue.get()
```

```
topological_sorter.done(node)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)graphlib.rst, line 85)

Unknown directive type "method".

```
.. method:: add(node, *predecessors)

   Add a new node and its predecessors to the graph. Both the *node* and all
   elements in *predecessors* must be hashable.

   If called multiple times with the same node argument, the set of
   dependencies will be the union of all dependencies passed in.

   It is possible to add a node with no dependencies (*predecessors* is not
   provided) or to provide a dependency twice. If a node that has not been
   provided before is included among *predecessors* it will be automatically
   added to the graph with no predecessors of its own.

   Raises :exc:`ValueError` if called after :meth:`~TopologicalSorter.prepare`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)graphlib.rst, line 100)

Unknown directive type "method".

```
.. method:: prepare()

   Mark the graph as finished and check for cycles in the graph. If any cycle
   is detected, :exc:`CycleError` will be raised, but
   :meth:`~TopologicalSorter.get_ready` can still be used to obtain as many
   nodes as possible until cycles block more progress. After a call to this
   function, the graph cannot be modified, and therefore no more nodes can be
   added using :meth:`~TopologicalSorter.add`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)graphlib.rst, line 109)

Unknown directive type "method".

```
.. method:: is_active()

   Returns ``True`` if more progress can be made and ``False`` otherwise.
   Progress can be made if cycles do not block the resolution and either
   there are still nodes ready that haven't yet been returned by
   :meth:`TopologicalSorter.get_ready` or the number of nodes marked
   :meth:`TopologicalSorter.done` is less than the number that have been
   returned by :meth:`TopologicalSorter.get_ready`.

   The :meth:`~TopologicalSorter.__bool__` method of this class defers to
   this function, so instead of::

       if ts.is_active():
           ...

   it is possible to simply do::

       if ts:
           ...

   Raises :exc:`ValueError` if called without calling
   :meth:`~TopologicalSorter.prepare` previously.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)graphlib.rst, line 132)

Unknown directive type "method".

```
.. method:: done(*nodes)

   Marks a set of nodes returned by :meth:`TopologicalSorter.get_ready` as
   processed, unblocking any successor of each node in *nodes* for being
   returned in the future by a call to :meth:`TopologicalSorter.get_ready`.
```

```
      Raises :exc:`ValueError` if any node in *nodes* has already been marked as
      processed by a previous call to this method or if a node was not added to
      the graph by using :meth:`TopologicalSorter.add`, if called without
      calling :meth:`~TopologicalSorter.prepare` or if node has not yet been
      returned by :meth:`~TopologicalSorter.get_ready`.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)graphlib.rst`, line 144)

Unknown directive type "method".

```
   .. method:: get_ready()

      Returns a ``tuple`` with all the nodes that are ready. Initially it
      returns all nodes with no predecessors, and once those are marked as
      processed by calling :meth:`TopologicalSorter.done`, further calls will
      return all new nodes that have all their predecessors already processed.
      Once no more progress can be made, empty tuples are returned.

      Raises :exc:`ValueError` if called without calling
      :meth:`~TopologicalSorter.prepare` previously.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)graphlib.rst`, line 155)

Unknown directive type "method".

```
   .. method:: static_order()

      Returns an iterator object which will iterate over nodes in a topological
      order. When using this method, :meth:`~TopologicalSorter.prepare` and
      :meth:`~TopologicalSorter.done` should not be called. This method is
      equivalent to::

          def static_order(self):
              self.prepare()
              while self.is_active():
                  node_group = self.get_ready()
                  yield from node_group
                  self.done(*node_group)

      The particular order that is returned may depend on the specific order in
      which the items were inserted in the graph. For example:

      .. doctest::

          >>> ts = TopologicalSorter()
          >>> ts.add(3, 2, 1)
          >>> ts.add(1, 0)
          >>> print([*ts.static_order()])
          [2, 0, 1, 3]

          >>> ts2 = TopologicalSorter()
          >>> ts2.add(1, 0)
          >>> ts2.add(3, 2, 1)
          >>> print([*ts2.static_order()])
          [0, 2, 1, 3]

      This is due to the fact that "0" and "2" are in the same level in the
      graph (they would have been returned in the same call to
      :meth:`~TopologicalSorter.get_ready`) and the order between them is
      determined by the order of insertion.


      If any cycle is detected, :exc:`CycleError` will be raised.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main) (Doc) (library)graphlib.rst`, line 194)

Unknown directive type "versionadded".

```
   .. versionadded:: 3.9
```

Exceptions

The :mod:`graphlib` module defines the following exception classes:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)graphlib.rst`, line 199); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\(cpython-main)(Doc)(library)graphlib.rst`, line 201)

Unknown directive type "exception".

```
.. exception:: CycleError

   Subclass of :exc:`ValueError` raised by :meth:`TopologicalSorter.prepare` if cycles exist
   in the working graph. If multiple cycles exist, only one undefined choice among them will
   be reported and included in the exception.

   The detected cycle can be accessed via the second element in the :attr:`~CycleError.args`
   attribute of the exception instance and consists in a list of nodes, such that each node is,
   in the graph, an immediate predecessor of the next node in the list. In the reported list,
   the first and the last node will be the same, to make it clear that it is cyclic.
```