:mod:`contextvars` --- Context Variables

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

Unknown directive type "module".

```
.. module:: contextvars
   :synopsis: Context Variables
```

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

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Yury Selivanov <yury@magic.io>
```

This module provides APIs to manage, store, and access context-local state. The :class:`~contextvars.ContextVar` class is used to declare and work with *Context Variables*. The :func:`~contextvars.copy_context` function and the :class:`~contextvars.Context` class should be used to manage the current context in asynchronous frameworks.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 11); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 11); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 11); *backlink*

Unknown interpreted text role "class".

Context managers that have state should use Context Variables instead of :func:`threading.local()` to prevent their state from bleeding to other code unexpectedly, when used in concurrent code.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 17); *backlink*

Unknown interpreted text role "func".

See also PEP 567 for additional details.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 23)

Unknown directive type "versionadded".

```
.. versionadded:: 3.7
```

Context Variables

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 29)

Invalid class attribute value for "class" directive: "ContextVar(name, [*, default])".

```
.. class:: ContextVar(name, [*, default])

   This class is used to declare a new Context Variable, e.g.::

       var: ContextVar[int] = ContextVar('var', default=42)

   The required *name* parameter is used for introspection and debug
   purposes.

   The optional keyword-only *default* parameter is returned by
   :meth:`ContextVar.get` when no value for the variable is found
   in the current context.

   **Important:** Context Variables should be created at the top module
   level and never in closures.  :class:`Context` objects hold strong
   references to context variables which prevents context variables
   from being properly garbage collected.

   .. attribute:: ContextVar.name

      The name of the variable.  This is a read-only property.

      .. versionadded:: 3.7.1

   .. method:: get([default])

      Return a value for the context variable for the current context.

      If there is no value for the variable in the current context,
      the method will:

      * return the value of the *default* argument of the method,
        if provided; or

      * return the default value for the context variable,
        if it was created with one; or

      * raise a :exc:`LookupError`.

   .. method:: set(value)

      Call to set a new value for the context variable in the current
      context.

      The required *value* argument is the new value for the context
      variable.

      Returns a :class:`~contextvars.Token` object that can be used
      to restore the variable to its previous value via the
      :meth:`ContextVar.reset` method.

   .. method:: reset(token)

      Reset the context variable to the value it had before the
      :meth:`ContextVar.set` that created the *token* was used.

      For example::

          var = ContextVar('var')

          token = var.set('new value')
          # code that uses 'var'; var.get() returns 'new value'.
          var.reset(token)

          # After the reset call the var has no value again, so
          # var.get() would raise a LookupError.
```

Token objects are returned by the :meth:`ContextVar.set` method. They can be passed to the :meth:`ContextVar.reset` method to revert the value of the variable to what it was before the corresponding *set*.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 99); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 99); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 104)

Unknown directive type "attribute".

```
.. attribute:: Token.var

   A read-only property.  Points to the :class:`ContextVar` object
   that created the token.
```

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

Unknown directive type "attribute".

```
.. attribute:: Token.old_value

   A read-only property.  Set to the value the variable had before
   the :meth:`ContextVar.set` method call that created the token.
   It points to :attr:`Token.MISSING` is the variable was not set
   before the call.
```

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

Unknown directive type "attribute".

```
.. attribute:: Token.MISSING

   A marker object used by :attr:`Token.old_value`.
```

Manual Context Management

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 124)

Unknown directive type "function".

```
.. function:: copy_context()

   Returns a copy of the current :class:`~contextvars.Context` object.

   The following snippet gets a copy of the current context and prints
   all variables and their values that are set in it::

      ctx: Context = copy_context()
      print(list(ctx.items()))

   The function has an O(1) complexity, i.e. works equally fast for
   contexts with a few context variables and for contexts that have
   a lot of them.
```

A mapping of :class:`ContextVars <ContextVar>` to their values.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 141); *backlink*

Unknown interpreted text role "class".

`Context()` creates an empty context with no values in it. To get a copy of the current context use the

:func:`~contextvars.copy_context` function.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 143); *backlink*

Unknown interpreted text role "func".

Context implements the :class:`collections.abc.Mapping` interface.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 147); *backlink*

Unknown interpreted text role "class".

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

Unknown directive type "method".

```
.. method:: run(callable, *args, **kwargs)

   Execute ``callable(*args, **kwargs)`` code in the context object
   the *run* method is called on.  Return the result of the execution
   or propagate an exception if one occurred.

   Any changes to any context variables that *callable* makes will
   be contained in the context object::

     var = ContextVar('var')
     var.set('spam')

     def main():
         # 'var' was set to 'spam' before
         # calling 'copy_context()' and 'ctx.run(main)', so:
         # var.get() == ctx[var] == 'spam'

         var.set('ham')

         # Now, after setting 'var' to 'ham':
         # var.get() == ctx[var] == 'ham'

     ctx = copy_context()

     # Any changes that the 'main' function makes to 'var'
     # will be contained in 'ctx'.
     ctx.run(main)

     # The 'main()' function was run in the 'ctx' context,
     # so changes to 'var' are contained in it:
     # ctx[var] == 'ham'

     # However, outside of 'ctx', 'var' is still set to 'spam':
     # var.get() == 'spam'

   The method raises a :exc:`RuntimeError` when called on the same
   context object from more than one OS thread, or when called
   recursively.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 188)

Unknown directive type "method".

```
.. method:: copy()

   Return a shallow copy of the context object.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 192)

Unknown directive type "describe".

```
.. describe:: var in context
```

```
    Return ``True`` if the *context* has a value for *var* set;
    return ``False`` otherwise.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 197)

Unknown directive type "describe".

```
.. describe:: context[var]

   Return the value of the *var* :class:`ContextVar` variable.
   If the variable is not set in the context object, a
   :exc:`KeyError` is raised.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 203)

Unknown directive type "method".

```
.. method:: get(var, [default])

   Return the value for *var* if *var* has the value in the context
   object.  Return *default* otherwise.  If *default* is not given,
   return ``None``.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 209)

Unknown directive type "describe".

```
.. describe:: iter(context)

   Return an iterator over the variables stored in the context
   object.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 214)

Unknown directive type "describe".

```
.. describe:: len(proxy)

   Return the number of variables set in the context object.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 218)

Unknown directive type "method".

```
.. method:: keys()

   Return a list of all variables in the context object.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 222)

Unknown directive type "method".

```
.. method:: values()

   Return a list of all variables' values in the context object.
```

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

Unknown directive type "method".

```
.. method:: items()

   Return a list of 2-tuples containing all variables and their
   values in the context object.
```

asyncio support

Context variables are natively supported in :mod:`asyncio` and are ready to be used without any extra configuration. For example, here is a simple echo server, that uses a context variable to make the address of a remote client available in the Task that handles that client:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]contextvars.rst`, line 236); ***backlink***

Unknown interpreted text role "mod".

```
import asyncio
import contextvars

client_addr_var = contextvars.ContextVar('client_addr')

def render_goodbye():
    # The address of the currently handled client can be accessed
    # without passing it explicitly to this function.

    client_addr = client_addr_var.get()
    return f'Good bye, client @ {client_addr}\n'.encode()

async def handle_request(reader, writer):
    addr = writer.transport.get_extra_info('socket').getpeername()
    client_addr_var.set(addr)

    # In any code that we call is now possible to get
    # client's address by calling 'client_addr_var.get()'.

    while True:
        line = await reader.readline()
        print(line)
        if not line.strip():
            break
        writer.write(line)

    writer.write(render_goodbye())
    writer.close()

async def main():
    srv = await asyncio.start_server(
        handle_request, '127.0.0.1', 8081)

    async with srv:
        await srv.serve_forever()

asyncio.run(main())

# To test it you can use telnet:
#     telnet 127.0.0.1 8081
```