Context Variables Objects

3.7.1

> In Python 3.7.1 the signatures of all context variables C APIs were **changed** to use `PyObject` pointers instead of `PyContext`, `PyContextVar`, and `PyContextToken`, e.g.:
>
> ```
> // in 3.7.0:
> PyContext *PyContext_New(void);
>
> // in 3.7.1+:
> PyObject *PyContext_New(void);
> ```
>
> See `34762` for more details.

This section details the public C API for the `contextvars` module.

> The C structure used to represent a `contextvars.Context` object.

> The C structure used to represent a `contextvars.ContextVar` object.

> The C structure used to represent a `contextvars.Token` object.

> The type object representing the *context* type.

> The type object representing the *context variable* type.

> The type object representing the *context variable token* type.

Type-check macros:

> Return true if *o* is of type `PyContext_Type`. *o* must not be `NULL`. This function always succeeds.

> Return true if *o* is of type `PyContextVar_Type`. *o* must not be `NULL`. This function always succeeds.

> Return true if *o* is of type `PyContextToken_Type`. *o* must not be `NULL`. This function always succeeds.

Context object management functions:

> Create a new empty context object. Returns `NULL` if an error has occurred.

> Create a shallow copy of the passed *ctx* context object. Returns `NULL` if an error has occurred.

> Create a shallow copy of the current thread context. Returns `NULL` if an error has occurred.

> Set *ctx* as the current context for the current thread. Returns `0` on success, and `-1` on error.

> Deactivate the *ctx* context and restore the previous context as the current context for the current thread. Returns `0` on success, and `-1` on error.

> Register *callback* as a context object watcher for the current interpreter. Return an ID which may be passed to `PyContext_ClearWatcher`. In case of error (e.g. no more watcher IDs available), return `-1` and set an exception.
>
> 3.14

> Clear watcher identified by *watcher_id* previously returned from `PyContext_AddWatcher` for the current interpreter. Return `0` on success, or `-1` and set an exception on error (e.g. if the given *watcher_id* was never registered.)
>
> 3.14

> Enumeration of possible context object watcher events:
>
> - `Py_CONTEXT_SWITCHED`: The `current context` has switched to a different context. The object passed to the watch callback is the now-current `contextvars.Context` object, or None if no context is current.
>
> 3.14

> Context object watcher callback function. The object passed to the callback is event-specific; see `PyContextEvent` for details.
>
> If the callback returns with an exception set, it must return `-1`; this exception will be printed as an unraisable exception using `PyErr_FormatUnraisable`. Otherwise it should return `0`.
>
> There may already be a pending exception set on entry to the callback. In this case, the callback should return `0` with the same exception still set. This means the callback may not call any other API that can set an exception unless it saves and clears the exception state first, and restores it before returning.
>
> 3.14

Context variable functions:

>

> Create a new `ContextVar` object. The *name* parameter is used for introspection and debug purposes. The *def* parameter specifies a default value for the context variable, or `NULL` for no default. If an error has occurred, this function returns `NULL`.

> Get the value of a context variable. Returns `-1` if an error has occurred during lookup, and `0` if no error occurred, whether or not a value was found.
>
> If the context variable was found, *value* will be a pointer to it. If the context variable was *not* found, *value* will point to:
>
> - *default_value*, if not `NULL`;
> - the default value of *var*, if not `NULL`;
> - `NULL`
>
> Except for `NULL`, the function returns a new reference.

> Set the value of *var* to *value* in the current context. Returns a new token object for this change, or `NULL` if an error has occurred.

> Reset the state of the *var* context variable to that it was in before `PyContextVar_Set` that returned the *token* was called. This function returns `0` on success and `-1` on error.

□□□ #2
□ □□□ □□□ 2026-10-31 16:50:23 UTC
□ □□□ □□□ 2026-10-31 16:50:25 UTC