PyTime C API

- Types
- Clock Functions
- Raw Clock Functions
- Conversion functions

3.13

The clock C API provides access to system clocks. It is similar to the Python `time` module.

For C API related to the `datetime` module, see `datetimeobjects`.

Types

> A timestamp or duration in nanoseconds, represented as a signed 64-bit integer.
>
> The reference point for timestamps depends on the clock used. For example, `PyTime_Time` returns timestamps relative to the UNIX epoch.
>
> The supported range is around [-292.3 years; +292.3 years]. Using the Unix epoch (January 1st, 1970) as reference, the supported date range is around [1677-09-21; 2262-04-11]. The exact limits are exposed as constants:

> Minimum value of `PyTime_t`.

> Maximum value of `PyTime_t`.

Clock Functions

The following functions take a pointer to a `PyTime_t` that they set to the value of a particular clock. Details of each clock are given in the documentation of the corresponding Python function.

The functions return `0` on success, or `-1` (with an exception set) on failure.

On integer overflow, they set the `PyExc_OverflowError` exception and set `*result` to the value clamped to the `[PyTime_MIN; PyTime_MAX]` range. (On current systems, integer overflows are likely caused by misconfigured system time.)

As any other C API (unless otherwise specified), the functions must be called with an `attached thread state`.

> Read the monotonic clock. See `time.monotonic` for important details on this clock.

> Read the performance counter. See `time.perf_counter` for important details on this clock.

> Read the “wall clock” time. See `time.time` for details important on this clock.

Raw Clock Functions

Similar to clock functions, but don't set an exception on error and don't require the caller to have an `attached thread state`.

On success, the functions return `0`.

On failure, they set `*result` to `0` and return `-1`, *without* setting an exception. To get the cause of the error, `attach <attached thread state>` a `thread state`, and call the regular (non-`Raw`) function. Note that the regular function may succeed after the `Raw` one failed.

>

> Similar to `PyTime_Monotonic`, but don't set an exception on error and don't require an `attached thread state`.

> Similar to `PyTime_PerfCounter`, but don't set an exception on error and don't require an `attached thread state`.

> Similar to `PyTime_Time`, but don't set an exception on error and don't require an `attached thread state`.

Conversion functions

> Convert a timestamp to a number of seconds as a C `double`.
>
> The function cannot fail, but note that `double` has limited accuracy for large values.

□□□ #2
□ □□□ □□□ 2026-10-31 16:54:45 UTC
□ □□□ □□□ 2026-10-31 16:54:46 UTC