DateTime Objects

Various date and time objects are supplied by the `datetime` module. Before using any of these functions, the header file `datetime.h` must be included in your source (note that this is not included by `Python.h`), and the macro `PyDateTime_IMPORT` must be invoked, usually as part of the module initialisation function. The macro puts a pointer to a C structure into a static variable, `PyDateTimeAPI`, that is used by the following macros.

Type-check macros:

int **`PyDate_Check`**(PyObject **ob*)

Return true if *ob* is of type `PyDateTime_DateType` or a subtype of `PyDateTime_DateType`. *ob* must not be *NULL*.

int **`PyDate_CheckExact`**(PyObject **ob*)

Return true if *ob* is of type `PyDateTime_DateType`. *ob* must not be *NULL*.

int **`PyDateTime_Check`**(PyObject **ob*)

Return true if *ob* is of type `PyDateTime_DateTimeType` or a subtype of `PyDateTime_DateTimeType`. *ob* must not be *NULL*.

int **`PyDateTime_CheckExact`**(PyObject **ob*)

Return true if *ob* is of type `PyDateTime_DateTimeType`. *ob* must not be *NULL*.

int **`PyTime_Check`**(PyObject **ob*)

Return true if *ob* is of type `PyDateTime_TimeType` or a subtype of `PyDateTime_TimeType`. *ob* must not be *NULL*.

int **`PyTime_CheckExact`**(PyObject **ob*)

Return true if *ob* is of type `PyDateTime_TimeType`. *ob* must not be *NULL*.

int **`PyDelta_Check`**(PyObject **ob*)

Return true if *ob* is of type `PyDateTime_DeltaType` or a subtype of `PyDateTime_DeltaType`. *ob* must not be *NULL*.

int **`PyDelta_CheckExact`**(PyObject **ob*)

Return true if *ob* is of type `PyDateTime_DeltaType`. *ob* must not be *NULL*.

int **`PyTZInfo_Check`**(PyObject **ob*)

Return true if *ob* is of type `PyDateTime_TZInfoType` or a subtype of `PyDateTime_TZInfoType`. *ob* must not be *NULL*.

int **`PyTZInfo_CheckExact`**(PyObject **ob*)

Return true if *ob* is of type `PyDateTime_TZInfoType`. *ob* must not be *NULL*.

Macros to create objects:

PyObject* **`PyDate_FromDate`**(int *year*, int *month*, int *day*)

Return value: New reference.

Return a `datetime.date` object with the specified year, month and day.

PyObject* **`PyDateTime_FromDateAndTime`**(int *year*, int *month*, int *day*, int *hour*, int *minute*, int *second*, int *usecond*)

Return value: New reference.

Return a `datetime.datetime` object with the specified year, month, day, hour, minute, second and microsecond.

PyObject* **`PyTime_FromTime`**(int *hour*, int *minute*, int *second*, int *usecond*)

Return value: New reference.

Return a `datetime.time` object with the specified hour, minute, second and microsecond.

PyObject* **`PyDelta_FromDSU`**(int *days*, int *seconds*, int *useconds*)

Return value: New reference.

Return a `datetime.timedelta` object representing the given number of days, seconds and microseconds. Normalization is performed so that the resulting number of microseconds and seconds lie in the ranges documented for `datetime.timedelta` objects.

Macros to extract fields from date objects. The argument must be an instance of `PyDateTime_Date`, including subclasses (such as `PyDateTime_DateTime`). The argument must not be *NULL*, and the type is not checked:

int **`PyDateTime_GET_YEAR`**(PyDateTime_Date **o*)

Return the year, as a positive int.

int **`PyDateTime_GET_MONTH`**(PyDateTime_Date **o*)

Return the month, as an int from 1 through 12.

int **`PyDateTime_GET_DAY`**(PyDateTime_Date **o*)

Return the day, as an int from 1 through 31.

Macros to extract fields from datetime objects. The argument must be an instance of `PyDateTime_DateTime`, including subclasses. The argument must not be *NULL*, and the type is not checked:

int **`PyDateTime_DATE_GET_HOUR`**(PyDateTime_DateTime **o*)

Return the hour, as an int from 0 through 23.

int **`PyDateTime_DATE_GET_MINUTE`**(PyDateTime_DateTime **o*)

Return the minute, as an int from 0 through 59.

int **`PyDateTime_DATE_GET_SECOND`**(PyDateTime_DateTime **o*)

Return the second, as an int from 0 through 59.

int **`PyDateTime_DATE_GET_MICROSECOND`**(PyDateTime_DateTime **o*)

Return the microsecond, as an int from 0 through 999999.

Macros to extract fields from time objects. The argument must be an instance of `PyDateTime_Time`, including subclasses. The argument must not be *NULL*, and the type is not checked:

int **`PyDateTime_TIME_GET_HOUR`**(PyDateTime_Time **o*)

Return the hour, as an int from 0 through 23.

int **`PyDateTime_TIME_GET_MINUTE`**(PyDateTime_Time **o*)

Return the minute, as an int from 0 through 59.

int **`PyDateTime_TIME_GET_SECOND`**(PyDateTime_Time **o*)

Return the second, as an int from 0 through 59.

int **`PyDateTime_TIME_GET_MICROSECOND`**(PyDateTime_Time **o*)

Return the microsecond, as an int from 0 through 999999.

Macros to extract fields from time delta objects. The argument must be an instance of `PyDateTime_Delta`, including subclasses. The argument must not be *NULL*, and the type is not checked:

int **`PyDateTime_DELTA_GET_DAYS`**(PyDateTime_Delta **o*)

Return the number of days, as an int from -999999999 to 999999999.

New in version 3.3.

int **`PyDateTime_DELTA_GET_SECONDS`**(PyDateTime_Delta **o*)

Return the number of seconds, as an int from 0 through 86399.

New in version 3.3.

int **`PyDateTime_DELTA_GET_MICROSECONDS`**(PyDateTime_Delta **o*)

Return the number of microseconds, as an int from 0 through 999999.

New in version 3.3.

Macros for the convenience of modules implementing the DB API:

PyObject* **`PyDateTime_FromTimestamp`**(PyObject **args*)

Return value: New reference.

Create and return a new `datetime.datetime` object given an argument tuple suitable for passing to `datetime.datetime.fromtimestamp()`.

PyObject* **`PyDate_FromTimestamp`**(PyObject **args*)

Return value: New reference.

Create and return a new `datetime.date` object given an argument tuple suitable for passing to `datetime.date.fromtimestamp()`.