DateTime Objects

Various date and time objects are supplied by the :mod:`datetime` module. Before using any of these functions, the header file :file:`datetime.h` must be included in your source (note that this is not included by :file:`Python.h`), and the macro :c: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, :c:data:`PyDateTimeAPI`, that is used by the following macros.

Macro for access to the UTC singleton:

Type-check macros:

```
.. c:function:: int PyDateTime_CheckExact(PyObject *ob)

    Return true if *ob* is of type :c:data:`PyDateTime_DateTimeType`. *ob* must not
    be ``NULL``.  This function always succeeds.
```

```
.. c:function:: int PyTime_Check(PyObject *ob)

    Return true if *ob* is of type :c:data:`PyDateTime_TimeType` or a subtype of
    :c:data:`PyDateTime_TimeType`.  *ob* must not be ``NULL``.  This function always
    succeeds.
```

```
.. c:function:: int PyTime_CheckExact(PyObject *ob)

    Return true if *ob* is of type :c:data:`PyDateTime_TimeType`. *ob* must not be
    ``NULL``.  This function always succeeds.
```

```
.. c:function:: int PyDelta_Check(PyObject *ob)

    Return true if *ob* is of type :c:data:`PyDateTime_DeltaType` or a subtype of
    :c:data:`PyDateTime_DeltaType`.  *ob* must not be ``NULL``.  This function always
    succeeds.
```

```
.. c:function:: int PyDelta_CheckExact(PyObject *ob)

    Return true if *ob* is of type :c:data:`PyDateTime_DeltaType`. *ob* must not be
    ``NULL``.  This function always succeeds.
```

```
.. c:function:: int PyTZInfo_Check(PyObject *ob)

    Return true if *ob* is of type :c:data:`PyDateTime_TZInfoType` or a subtype of
    :c:data:`PyDateTime_TZInfoType`.  *ob* must not be ``NULL``.  This function always
    succeeds.
```

```
.. c:function:: int PyTZInfo_CheckExact(PyObject *ob)

    Return true if *ob* is of type :c:data:`PyDateTime_TZInfoType`. *ob* must not be
    ``NULL``.  This function always succeeds.
```

Macros to create objects:

```
.. c:function:: PyObject* PyDate_FromDate(int year, int month, int day)

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

```
.. c:function:: PyObject* PyDateTime_FromDateAndTime(int year, int month, int day, int hour, int minute, int second, int

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

```
.. c:function:: PyObject* PyDateTime_FromDateAndTimeAndFold(int year, int month, int day, int hour, int minute, int second

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

    .. versionadded:: 3.6
```

```
.. c:function:: PyObject* PyTime_FromTime(int hour, int minute, int second, int usecond)

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

```
.. c:function:: PyObject* PyTime_FromTimeAndFold(int hour, int minute, int second, int usecond, int fold)

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

    .. versionadded:: 3.6
```

```
.. c:function:: PyObject* PyDelta_FromDSU(int days, int seconds, int useconds)

    Return a :class:`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
    :class:`datetime.timedelta` objects.
```

```
.. c:function:: PyObject* PyTimeZone_FromOffset(PyDateTime_DeltaType* offset)

    Return a :class:`datetime.timezone` object with an unnamed fixed offset
    represented by the *offset* argument.

    .. versionadded:: 3.7
```

```
.. c:function:: PyObject* PyTimeZone_FromOffsetAndName(PyDateTime_DeltaType* offset, PyUnicode* name)

    Return a :class:`datetime.timezone` object with a fixed offset represented
    by the *offset* argument and with tzname *name*.

    .. versionadded:: 3.7
```

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

```
.. c:function:: int PyDateTime_GET_YEAR(PyDateTime_Date *o)

    Return the year, as a positive int.
```

```
.. c:function:: int PyDateTime_GET_MONTH(PyDateTime_Date *o)

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

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]datetime.rst, line 165`)

Unknown directive type "c:function".

```
.. c:function:: 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 :c:data:`PyDateTime_DateTime`, including subclasses. The argument must not be NULL, and the type is not checked:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]datetime.rst, line 170`); *backlink*

Unknown interpreted text role "c:data".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]datetime.rst, line 174`)

Unknown directive type "c:function".

```
.. c:function:: int PyDateTime_DATE_GET_HOUR(PyDateTime_DateTime *o)

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

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]datetime.rst, line 179`)

Unknown directive type "c:function".

```
.. c:function:: int PyDateTime_DATE_GET_MINUTE(PyDateTime_DateTime *o)

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

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]datetime.rst, line 184`)

Unknown directive type "c:function".

```
.. c:function:: int PyDateTime_DATE_GET_SECOND(PyDateTime_DateTime *o)

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

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]datetime.rst, line 189`)

Unknown directive type "c:function".

```
.. c:function:: int PyDateTime_DATE_GET_MICROSECOND(PyDateTime_DateTime *o)

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

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]datetime.rst, line 193`)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyDateTime_DATE_GET_TZINFO(PyDateTime_DateTime *o)

   Return the tzinfo (which may be ``None``).

   .. versionadded:: 3.10
```

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

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]datetime.rst, line 199`); *backlink*

Unknown interpreted text role "c:data".

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

Unknown directive type "c:function".

```
.. c:function:: int PyDateTime_TIME_GET_HOUR(PyDateTime_Time *o)

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

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\[cpython-main][Doc][c-api]datetime.rst, line 208`)

Unknown directive type "c:function".

```
.. c:function:: int PyDateTime_TIME_GET_MINUTE(PyDateTime_Time *o)
```

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

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

Macros for the convenience of modules implementing the DB API:

```
tuple suitable for passing to :meth:`datetime.date.fromtimestamp()`.
```