Python Date and Time Modules Guide

1. time Module Overview

The time module in Python is used to work with time-related functions at a low level. It provides access to time-related functions that are based on the C standard library.

Key Functions:

- time.time(): Returns the current time in seconds since the Epoch (January 1, 1970).
- time.sleep(seconds): Pauses execution for a specified number of seconds.
- time.localtime([secs]): Converts seconds since the Epoch to a struct_time in local time.
- time.gmtime([secs]): Converts seconds since the Epoch to a struct_time in UTC.
- time.strftime(format[, t]): Formats a struct_time into a string. If 't' is omitted, defaults to localtime().
- time.strptime(string, format): Parses a string into a struct_time using the specified format.
- time.ctime([secs]): Converts a time expressed in seconds to a string.
- time.mktime(t): Converts a struct_time to seconds since the Epoch.

Example:

```
import time
print(time.strftime('%Y-%m-%d %H:%M:%S'))  # Current time formatted
print(time.strftime('%A', time.localtime()))  # Current day name
```

2. datetime Module Overview

The datetime module supplies classes for manipulating dates and times in a more object-oriented way.

Main Classes:

- datetime.datetime: Combines both date and time into one object.
- datetime.date: Represents just the date (year, month, day).
- datetime.time: Represents time (hour, minute, second, microsecond).
- datetime.timedelta: Represents the difference between two datetime or date objects.
- datetime.tzinfo: Base class for dealing with time zones.

Common Methods:

Python Date and Time Modules Guide

- datetime.now(): Returns the current local date and time.
- datetime.utcnow(): Returns the current UTC date and time.
- datetime.strftime(format): Formats a datetime object into a string.
- datetime.strptime(date_string, format): Parses a string into a datetime object.

Example:

```
from datetime import datetime
now = datetime.now()
print(now.strftime('%Y-%m-%d %H:%M:%S'))
date_obj = datetime.strptime('2025-07-11', '%Y-%m-%d')
```

3. strftime/strptime Format Codes

These format codes can be used with both datetime.strftime() and time.strftime():

%Y - Year with century (2025)
%y - Year without century (25)
%m - Month as a number (01-12)
%B - Full month name (July)
%b - Abbreviated month name (Jul)
%d - Day of the month (01-31)
%A - Full weekday name (Saturday)
%a - Abbreviated weekday name (Sat)
%H - Hour (00-23)
%I - Hour (01-12)
%p - AM/PM
%M - Minute (00-59)
%S - Second (00-59)
%f - Microsecond (000000-999999)
%z - UTC offset (+0530)
%Z - Time zone (IST)

%j - Day of the year (001-366)

%U - Week number, Sunday as the first day (00-53)

%W - Week number, Monday as the first day (00-53)

%c - Locale's date and time

%x - Locale's date

%X - Locale's time

%% - Literal '%' character

4. Key Differences Between time.strftime and datetime.strftime

1. time.strftime() is a module-level function:

- Can be called directly without an object.

- Defaults to time.localtime() if no second argument is passed.

Example:

```
import time
print(time.strftime('%c'))
print(time.strftime('%c', time.localtime()))
```

2. datetime.strftime() is an instance method:

- Must be called on a datetime object.

- Cannot be used directly from the class.

Example:

```
from datetime import datetime
now = datetime.now()
print(now.strftime('%c'))
# datetime.strftime('%c') -> This would raise a TypeError
```

Use time.strftime() for struct_time objects.

Use datetime.strftime() for datetime objects.