datetime Module in Python

The `datetime` module in Python is used to work with dates and times. It allows you to represent and manipulate both date and time in your programs. It's useful for tasks like logging, event scheduling, age calculation, and more.

---

datetime.now()

Purpose: Returns the current local date and time as a `datetime` object.

You can use this method to get the exact current moment when your program is running.

Examples:

```
from datetime import datetime

print(datetime.now())          # Returns full date and time, e.g., 2025-06-30 10:15:42.123456
print(datetime.now().date())   # Returns only the date part, e.g., 2025-06-30
print(datetime.now().time())   # Returns only the time part, e.g., 10:15:42.123456
```

---

datetime.strptime(string, format)

Purpose: Converts a date/time string into a `datetime` object using the given format.

This is useful when you get a date as text (like from user input or a file) and want to convert it into a `datetime` object to work with.

Examples:

```
from datetime import datetime

print(datetime.strptime("2025-06-30", "%Y-%m-%d"))   # Converts ISO-style string
print(datetime.strptime("30/06/25", "%d/%m/%y"))     # Converts custom format
print(datetime.strptime("June 30, 2025", "%B %d, %Y"))  # Full month name
```

---

datetime.strftime(format)

Format Codes Used with strftime and strptime

Here is a list of common format symbols used in both `strftime()` and `strptime()`:

Code	Meaning	Example
%Y	Year (4 digits)	2025
%y	Year (last 2 digits)	25
%m	Month number (01 to 12)	06
%B	Full month name	June
%b	Abbreviated month name	Jun
%d	Day of the month (01 to 31)	30
%A	Full weekday name	Monday
%a	Abbreviated weekday name	Mon
%H	Hour (24-hour clock)	14
%I	Hour (12-hour clock)	02
%p	AM or PM	AM
%M	Minutes	15
%S	Seconds	45
%%	Literal % character	%

Purpose: Converts a `datetime` object into a string using a specified format.

This is useful for displaying the date in a user-friendly or required format.

Examples:

```
from datetime import datetime

now = datetime.now()
print(now.strftime("%Y-%m-%d"))    # Outputs 2025-06-30
print(now.strftime("%A"))          # Outputs full weekday name like 'Monday'
print(now.strftime("%I:%M %p"))    # Outputs time in 12-hour format like '10:15
AM'
```

datetime.replace()

Purpose: Creates a new `datetime` object with specific values replaced (like year, hour, etc).

You can use this to modify part of a datetime without creating a new one from scratch.

Examples:

```
from datetime import datetime

dt = datetime(2025, 6, 30, 10, 15)
print(dt.replace(year=2030))        # Changes the year to 2030
print(dt.replace(month=12, day=25)) # Changes the date to December 25
print(dt.replace(hour=0, minute=0)) # Changes the time to midnight
```

---

datetime.weekday()

Purpose: Returns the weekday as an integer (0 = Monday, 6 = Sunday).

You can use this method to find out which day of the week a particular date falls on.

Examples:

```
from datetime import datetime

print(datetime(2025, 6, 30).weekday())  # Returns 0 (Monday)
print(datetime(2025, 7, 1).weekday())   # Returns 1 (Tuesday)
print(datetime.now().weekday())        # Returns current weekday number
```

---

datetime.isoweekday()

Purpose: Returns the weekday as an integer (1 = Monday, 7 = Sunday).

Same as `weekday()`, but starts counting from 1 instead of 0.

Examples:

```
from datetime import datetime

```

```
print(datetime(2025, 6, 30).isoweekday())  # Returns 1 (Monday)
print(datetime(2025, 7, 1).isoweekday())   # Returns 2 (Tuesday)
print(datetime.now().isoweekday())        # Returns current weekday number
```

---

timedelta

Purpose: Represents a difference between two dates/times. Used for arithmetic.

You can use it to add or subtract days, hours, or minutes to/from a datetime.

Examples:

```
from datetime import datetime, timedelta

print(datetime.now() + timedelta(days=5))     # 5 days later
print(datetime.now() - timedelta(days=10))    # 10 days ago
print(datetime.now() + timedelta(hours=3))    # 3 hours later
```

---

datetime.isoformat()

Purpose: Returns the date/time in ISO 8601 format (standardized format).

This is useful for storing or sending date/time values across systems.

Examples:

```
from datetime import datetime

now = datetime.now()
print(now.isoformat())            # 2025-06-30T10:15:00.123456
print(now.date().isoformat())     # 2025-06-30
print(now.time().isoformat())     # 10:15:00.123456
```

---

datetime.date() and datetime.time()

Purpose: Extracts only the date or time from a `datetime` object.

This is helpful if you want to work with just the date or just the time.

Examples:

```python
from datetime import datetime

now = datetime.now()
print(now.date())    # 2025-06-30
print(now.time())    # 10:15:00.123456
print(type(now.date()))  # <class 'datetime.date'>
```