



A `date` is a day, a `datetime` is a day plus a clock time, and a `timedelta` is a length of time: subtract two datetimes and you get one. Examples use `d = datetime(2026, 3, 14, 9, 30)`, a Saturday.

CREATE

```
from datetime import date, time,
datetime, timedelta, timezone
the names you'll use most

date(2026, 3, 14)
→ datetime.date(2026, 3, 14)
a day, no time

time(9, 30) → datetime.time(9, 30)
a clock time, no day

d.date() → datetime.date(2026, 3, 14)
drop the time part

datetime.combine(
    date(2026, 3, 14), time(18))
→ datetime.datetime(2026, 3, 14, 18, 0)
join a date and a time

datetime.now()
now, local clock, no time zone attached

datetime.now(timezone.utc)
now in UTC: the safe choice for logs

date.today() today's date
```

PARTS

```
d.year, d.month, d.day → (2026, 3, 14)
d.hour, d.minute → (9, 30)
d.weekday() → 5
Monday is 0, Sunday is 6
d.isoweekday() → 6
Monday is 1, Sunday is 7
d.isocalendar().week → 11
ISO week number

d.replace(day=1).date()
→ datetime.date(2026, 3, 1)
change one part: first of the month
```

DATE TO TEXT

```
d.strftime("%Y-%m-%d") → '2026-03-14'
d.strftime("%d/%m/%Y") → '14/03/2026'
d.strftime("%a %d %b %Y")
→ 'Sat 14 Mar 2026'
d.strftime("%A, %B %d")
→ 'Saturday, March 14'
d.strftime("%I:%M %p") → '09:30 AM'
12-hour clock

f"{d:%d %b, %H:%M}" → '14 Mar, 09:30'
the same codes work in f-strings

d.isoformat() → '2026-03-14T09:30:00'
the standard form for files and APIs

d.isoformat(" ", "minutes")
→ '2026-03-14 09:30'
```

FORMAT CODES

%Y	%y	2026, 26	
%m	%B	%b	03, March, Mar
%d	%j	14, day of the year 073	
%A	%a	Saturday, Sat	
%H	%I	%p	09 (24-hour), 09 (12-hour), AM
%M	%S	%f	minute, second, microsecond
%z	%Z	+0530, IST (aware datetimes only)	

TEXT TO DATE

```
datetime.strptime(
    "14/03/2026", "%d/%m/%Y")
→ datetime.datetime(2026, 3, 14, 0, 0)
the format must match the text exactly

datetime.strptime(
    "9:05 PM", "%I:%M %p").hour
→ 21
%p reads AM or PM

date.fromisoformat("2026-03-14")
→ datetime.date(2026, 3, 14)
fastest for ISO text

datetime.fromisoformat(
    "2026-03-14T09:30Z").tzinfo
→ datetime.timezone.utc
Z and offsets accepted since 3.11

try:
    datetime.strptime(text, "%d/%m/%Y")
except ValueError:
    print("not a date")
bad text and impossible dates (31/02) raise ValueError
```

DATE MATHS

```
d + timedelta(days=30)
→ datetime.datetime(2026, 4, 13, 9, 30)
30 days later

d - timedelta(hours=10)
→ datetime.datetime(2026, 3, 13, 23, 30)
rolls back across midnight

(date(2026, 12, 25) - d.date()).days
→ 286
days until a date

abs(date(2026, 1, 1) - d.date()).days
→ 72
abs() when either could be later

d.replace(hour=0, minute=0)
→ datetime.datetime(2026, 3, 14, 0, 0)
start of the day

timedelta(hours=36)
→ datetime.timedelta(days=1,
seconds=43200)
stored as days, seconds, µs

timedelta(hours=27).total_seconds()
→ 97200.0
the whole length in seconds

str(timedelta(minutes=95)) → '1:35:00'
readable duration

timedelta(weeks=2) // timedelta(days=1)
→ 14
how many days fit

d < d + timedelta(seconds=1) → True
datetimes compare and sort

date(2026, 3, 1) - timedelta(days=1)
→ datetime.date(2026, 2, 28)
last day of the month before
```

TIME ZONES

```
pip install tzdata
Windows needs this once for ZoneInfo

ist = ZoneInfo("Asia/Kolkata")
ny = ZoneInfo("America/New_York")
from zoneinfo import ZoneInfo

t = d.replace(tzinfo=ist)
attach a zone: same clock time, now aware

f"{t:%H:%M %Z}" → '09:30 IST'

f"{t.astimezone(ny):%H:%M %Z}"
→ '00:00 EDT'
astimezone converts: same moment

t.astimezone(timezone.utc).isoformat()
→ '2026-03-14T04:00:00+00:00'
store and compare in UTC

t.utcoffset()
→ datetime.timedelta(seconds=19800)
+5:30

timezone(timedelta(hours=5, minutes=30))
a fixed offset, no daylight saving rules

t.timestamp() → 1773460800.0
seconds since 1970 UTC

datetime.fromtimestamp(
    0, timezone.utc).year
→ 1970
and back again

calendar.monthrange(2026, 2)[1] → 28
days in a month; import calendar

calendar.isleap(2028) → True

ahead = (0 - d.weekday()) % 7 or 7
d.date() + timedelta(days=ahead)
→ datetime.date(2026, 3, 16)
next Monday (0), never today; 4 for Friday

age = t.year - b.year - (
    (t.month, t.day) < (b.month, b.day))
age in whole years, t today, b birthday

sorted(['2026-03-14', '2025-12-01'])
→ ['2025-12-01', '2026-03-14']
ISO text sorts in date order
```

WATCH OUT

```
d < d.replace(tzinfo=timezone.utc)
→ raises TypeError
naive and aware can't mix: give both a zone

d.strftime("%Y-%M-%d") → '2026-30-14'
%M is minutes, %m is the month

timedelta(days=1, seconds=5).seconds → 5
.seconds skips the days: use total_seconds()

date(2026, 1, 31) + timedelta(days=30)
→ datetime.date(2026, 3, 2)
no months in timedelta: a month isn't 30 days

date(2024, 2, 29).replace(year=2025)
→ raises ValueError
29 Feb doesn't exist every year

d == date(2026, 3, 14) → False
a datetime never equals a date: use d.date()

datetime.utcnow()
deprecated, and returns naive: use now(timezone.utc)
```