**1. datetime Module**

**Why we use it**

The datetime module is used to work with **dates** and **times** in Python. It allows you to:

* Get the current date and time.
* Create specific date/time objects.
* Perform date arithmetic (add/subtract days, hours, etc.).
* Format and parse dates.

**Key Classes**

* datetime.date: Work with dates only (year, month, day).
* datetime.time: Work with times only (hour, minute, second).
* datetime.datetime: Work with both date and time.
* datetime.timedelta: Work with time intervals.

**Example**

python

CopyEdit

from datetime import datetime, date, time, timedelta

# Current date and time

now = datetime.now()

print("Now:", now)

# Specific date

d = date(2024, 7, 25)

print("Date:", d)

# Specific time

t = time(14, 30)

print("Time:", t)

# Formatting and parsing

formatted = now.strftime("%Y-%m-%d %H:%M:%S")

print("Formatted:", formatted)

parsed = datetime.strptime("2025-07-26 10:30:00", "%Y-%m-%d %H:%M:%S")

print("Parsed:", parsed)

# Adding days using timedelta

new\_date = now + timedelta(days=7)

print("One week later:", new\_date)

**2. time Module**

**Why we use it**

The time module is used for **measuring execution time**, **pausing programs**, and working with timestamps (epoch time).

**Functions**

* time.time(): Get current time in seconds since epoch.
* time.sleep(seconds): Pause program for given seconds.
* time.ctime(timestamp): Convert timestamp to readable format.

**Example**

python

CopyEdit

import time

# Current time in seconds since epoch

current\_time = time.time()

print("Epoch time:", current\_time)

# Human-readable time

print("Readable time:", time.ctime(current\_time))

# Sleep for 2 seconds

print("Sleeping...")

time.sleep(2)

print("Awake now!")

**3. calendar Module**

**Why we use it**

The calendar module lets you work with entire calendars and weekdays. Useful for **finding specific days** or **printing calendars**.

**Functions**

* calendar.month(year, month): Show calendar of a month.
* calendar.isleap(year): Check if a year is a leap year.
* calendar.weekday(year, month, day): Get weekday (0=Monday).

**Example**

python

CopyEdit

import calendar

# Print July 2025 calendar

print(calendar.month(2025, 7))

# Leap year check

print("Is 2024 leap year?", calendar.isleap(2024))

# Day of week

print("Weekday (0=Mon):", calendar.weekday(2025, 7, 26))

**4. timedelta (from datetime)**

**Why we use it**

timedelta represents a duration or difference between two dates/times.

**Example**

python

CopyEdit

from datetime import datetime, timedelta

today = datetime.now()

print("Today:", today)

# Add 10 days

future = today + timedelta(days=10)

print("10 days later:", future)

# Subtract hours

past = today - timedelta(hours=5)

print("5 hours ago:", past)

# Difference between dates

difference = future - today

print("Difference:", difference.days, "days")

**5. arrow Module (External Library)**

**Why we use it**

arrow is an external library that **simplifies datetime handling**:

* Timezones are easier to work with.
* Easier formatting and shifting dates.

**Installation**

bash

CopyEdit

pip install arrow

**Example**

python

CopyEdit

import arrow

# Current time

now = arrow.now()

print("Now:", now)

# Format

print("Formatted:", now.format('YYYY-MM-DD HH:mm'))

# Shift dates

print("3 days ago:", now.shift(days=-3))

# Parse string

parsed = arrow.get('2025-07-26', 'YYYY-MM-DD')

print("Parsed:", parsed)

**6. dateutil Module (External Library)**

**Why we use it**

dateutil extends the functionality of datetime.

* Handles **timezones**, **relative deltas**, and **parsing any date string**.

**Installation**

bash

CopyEdit

pip install python-dateutil

**Example**

python

CopyEdit

from datetime import datetime

from dateutil.relativedelta import relativedelta

from dateutil import parser

# Relative delta: Add months/years

now = datetime.now()

print("Now:", now)

future = now + relativedelta(months=+2)

print("Two months later:", future)

# Parse any date string automatically

parsed\_date = parser.parse("July 26th, 2025 10:30 AM")

print("Parsed:", parsed\_date)

**Why We Use Them**

1. **datetime**: Built-in and most versatile for date/time manipulation.
2. **time**: Used when you need to work with epoch time, delays, or measure execution time.
3. **calendar**: Useful for printing or calculating entire calendars.
4. **timedelta**: For adding or subtracting time intervals.
5. **arrow**: Simplifies timezone handling and formatting compared to datetime.
6. **dateutil**: Automatically parses various date formats and handles complex intervals.

**Do you want me to combine all these modules in one big practical example (like a mini project: meeting scheduler with time zones and reminders)? Or should I make a clean PDF-style document with tables and diagrams?**

**You said:**

Do you want me to combine all these modules in one big practical example (like a mini project: meeting scheduler with time zones and reminders)? ok give me

**ChatGPT said:**

Great! Here is a **practical mini-project** that combines datetime, time, calendar, timedelta, arrow, and dateutil into a **Meeting Scheduler with Time Zones and Reminders**.

**Project: Meeting Scheduler**

**Features**

1. Schedule a meeting with a date and time (user input).
2. Automatically handle **time zones**.
3. Calculate how much time is left until the meeting (using timedelta).
4. Add the meeting date to a **calendar view**.
5. Set a **reminder** before the meeting using time.sleep().
6. Support flexible date input using dateutil.

**Code**

python

CopyEdit

from datetime import datetime, timedelta

import time

import calendar

import arrow

from dateutil import parser, tz

# Step 1: Get meeting details from the user

meeting\_input = input("Enter your meeting date & time (e.g., '2025-07-28 14:30'): ")

meeting = parser.parse(meeting\_input) # Parses any format

# Step 2: Handle time zones using arrow

local\_meeting = arrow.get(meeting).to('Asia/Karachi') # Convert to local timezone

print(f"Meeting scheduled at: {local\_meeting.format('YYYY-MM-DD HH:mm')} (Asia/Karachi)")

# Step 3: Calculate how much time is left until the meeting

now = datetime.now(tz=tz.gettz('Asia/Karachi'))

time\_left = local\_meeting.datetime - now

if time\_left.total\_seconds() > 0:

print(f"Time left until the meeting: {time\_left.days} days and {time\_left.seconds//3600} hours")

else:

print("Meeting time has already passed!")

# Step 4: Show the calendar of that month

print("\nHere is the calendar for your meeting month:")

print(calendar.month(local\_meeting.year, local\_meeting.month))

# Step 5: Set a reminder 10 seconds before the meeting (for demo)

reminder\_time = local\_meeting.datetime - timedelta(seconds=10)

time\_to\_wait = (reminder\_time - now).total\_seconds()

if time\_to\_wait > 0:

print(f"Reminder set for: {reminder\_time}")

print("Waiting for reminder...")

time.sleep(time\_to\_wait)

print("⏰ Reminder: Your meeting is about to start in 10 seconds!")

else:

print("Reminder time already passed.")

**How It Uses All Modules**

* **datetime** → Handles date and time calculations.
* **timedelta** → Calculates how much time is left before the meeting.
* **time** → Waits and sends reminders using sleep().
* **calendar** → Displays the monthly calendar of the meeting date.
* **arrow** → Converts meeting date into local time zones.
* **dateutil.parser** → Parses flexible user input for date and time.

**Sample Run**

yaml

CopyEdit

Enter your meeting date & time (e.g., '2025-07-28 14:30'): 2025-07-28 14:30

Meeting scheduled at: 2025-07-28 14:30 (Asia/Karachi)

Time left until the meeting: 1 days and 2 hours

Here is the calendar for your meeting month:

July 2025

Mo Tu We Th Fr Sa Su

1 2 3 4 5 6

7 8 9 10 11 12 13

14 15 16 17 18 19 20

21 22 23 24 25 26 27

28 29 30 31

Reminder set for: 2025-07-28 14:29:50

Waiting for reminder...

⏰ Reminder: Your meeting is about to start in 10 seconds!