🕒Python `time` Module - Colorful Study Notes

🔗Introduction

The `` **module** in Python is a **built-in module** that provides functions to work with **time-related tasks**, such as getting the current time, measuring execution duration, and creating delays. ⏳⏱️📅

🧱It's part of the **standard library** → no installation required.

```
import time
```

⏳Getting Current Time

- `` → Current time in **seconds since Epoch** (Jan 1, 1970, UTC)

```
print(time.time())  # e.g., 1693533021.123456
```

- `` → Converts seconds to a **readable string**

```
print(time.ctime())  # e.g., 'Sun Aug 31 22:15:21 2025'
```

- `` `→ Local time as` struct_time`

```
t = time.localtime()
print(t)
```

- `` `→ UTC time as` struct_time`

```
print(time.gmtime())
```

Pausing Execution

- `` → Pause program for given seconds.

```
print("Start")
time.sleep(3)  # waits for 3 seconds
print("End")
```

🔗Useful for automation, API rate-limiting, or animations.

⏱️Measuring Execution Time

- Using `time.time()`:

```
start = time.time()
for i in range(1000000):
    pass
end = time.time()
print("Execution time:", end - start, "seconds")
```

- Using **high-precision** timers:

```
start = time.perf_counter()
# code block
end = time.perf_counter()
print("Precise execution time:", end - start)
```

🛠️Formatting and Parsing Time

- **Formatting (``)** → struct_time → string

```
t = time.localtime()
print(time.strftime("%Y-%m-%d %H:%M:%S", t))
# e.g., 2025-08-31 22:20:00
```

- **Parsing (``)** → string → struct_time

```
t = time.strptime("2025-08-31 22:20:00", "%Y-%m-%d %H:%M:%S")
print(t)
```

🕐UTC vs Local Time

```
print(time.localtime())  # Local time 💻
print(time.gmtime())     # UTC time 🕐
```

🔎Working with `struct_time`

Many functions return a `` object.

```
t = time.localtime()
print(t)
print(t.tm_year, t.tm_mon, t.tm_mday)
```

Attributes of `struct_time`

- `tm_year` → Year 📅
- `tm_mon` → Month 📅
- `tm_mday` → Day 🗓️
- `tm_hour` → Hour ⏰
- `tm_min` → Minute ⏱️
- `tm_sec` → Second 🕐

🕐Conversions

```
print(time.asctime(time.localtime()))  # struct_time → String
print(time.mktime(time.localtime()))   # struct_time → Timestamp
```

📌Other Useful Functions

- `time.monotonic()` → Monotonic clock (never goes backward) 🔦
- `time.process_time()` → CPU time used by process 💻
- `time.time_ns()` → Current time in nanoseconds 🧾

📝Summary

The `` **module** is mainly used for:

1. ⏳Getting current time.
2. 🛠️Formatting & parsing dates.
3. 🚽Pausing execution.
4. ⏱️Measuring execution duration.
5. 🕐Conversions between timestamp/string/struct_time.
6. 🔎Inspecting date-time parts with `struct_time`.

🧱With these tools, you can **track, delay, format, and analyze time easily** in Python!