Klar! Hier kommt dein Lernzettel zum Python-Modul **`time`**, mit Fokus auf die **Stoppuhr-Funktionalität** — also Zeit messen, Intervalle stoppen, etc.

Lernzettel: Python-Modul `time` — Stoppuhr & Zeitmessung

---

1. Überblick `time` für Zeitmessung

- Das Modul `time` bietet Funktionen, um Zeiten zu messen, zu pausieren und Zeitstempel zu bekommen.
- Für Stoppuhren nutzt man hauptsächlich:
  - `time.time()` — aktuelle Zeit in Sekunden seit Epoch (1.1.1970)
  - `time.perf_counter()` — sehr genaue Zeitmessung für Intervalle
  - `time.sleep(sekunden)` — Programm für x Sekunden pausieren

---

2. Warum `perf_counter()` statt `time()`?

- `time.time()` kann durch Systemzeitänderungen verfälscht werden.
- `time.perf_counter()` ist eine hochauflösende, monotone Uhr, ideal für Zeitmessung (z.B. Stoppuhr).
- Verwende also für Stoppuhren immer `perf_counter()`.

---

3. Stoppuhr einfach mit `perf_counter()`

```
import time

start = time.perf_counter()  # Stoppuhr starten

# Beispiel: Code block, dessen Laufzeit du messen willst
for i in range(1000000):
    pass

ende = time.perf_counter()   # Stoppuhr stoppen

print(f"Laufzeit: {ende - start:.5f} Sekunden")
```

Erklärung:

- `perf_counter()` gibt Zeit in Sekunden als Fließkommazahl zurück.

- Differenz `ende - start` ist die gemessene Zeit.

---

4. Stoppuhr mit Pause und Fortsetzen

```
import time

start = time.perf_counter()
time.sleep(1.5)  # Simuliere Arbeit

pause_start = time.perf_counter()
time.sleep(2)  # Pause (z.B. Benutzer wartet)
pause_ende = time.perf_counter()

zeit_in_pause = pause_ende - pause_start

# Fortsetzen
time.sleep(1)  # weitere Arbeit

ende = time.perf_counter()

gesamtzeit = (ende - start) - zeit_in_pause
print(f"Effektive Laufzeit ohne Pause: {gesamtzeit:.3f} Sekunden")
```

---

5. Stoppuhr als Klasse für wiederverwendbare Nutzung

```
class Stoppuhr:
    def __init__(self):
        self.startzeit = None
        self.gestoppte_zeit = 0
        self.laeuft = False

    def start(self):
        if not self.laeuft:
            self.startzeit = time.perf_counter()
            self.laeuft = True

    def stop(self):
        if self.laeuft:
            self.gestoppte_zeit += time.perf_counter() - self.startzeit
            self.laeuft = False

    def reset(self):
        self.startzeit = None
        self.gestoppte_zeit = 0
        self.laeuft = False

    def zeit(self):
        if self.laeuft:
            return self.gestoppte_zeit + (time.perf_counter() - self.startzeit)
        return self.gestoppte_zeit

# Beispiel Nutzung
uhr = Stoppuhr()
uhr.start()
time.sleep(1)
uhr.stop()
print(f"Gestoppte Zeit: {uhr.zeit():.3f} Sekunden")
```

---

6. `time.sleep()` — Pause einlegen

- Nutze `time.sleep(sekunden)`, um das Programm bewusst zu pausieren (z.B. Wartezeit zwischen Messungen).
- Akzeptiert auch Fließkommazahlen für Millisekunden:

```
time.sleep(0.5)  # 500 Millisekunden warten
```

---

7. Zusammenfassung wichtige Funktionen für Stoppuhr

Funktion	Beschreibung
`time.perf_counter()`	Hochauflösende, monotone Zeit in Sek. (Stoppuhr ideal)
`time.time()`	Aktuelle Zeit seit Epoch (nicht ideal für Stoppuhr)
`time.sleep(sek)`	Programm pausieren für `sek` Sekunden

Wenn du möchtest, kann ich dir noch Beispielcodes geben, wie du mit `time` regelmäßig Zeitintervalle misst (z.B. jede Sekunde was tun) oder eine Countdown-Uhr baust.

Möchtest du das?