Programmieren – Praktikum 3

Ingenieurinformatik Teil 1, Wintersemester 2026/27

David Straub

Ein simuliertes Batteriepack

Bisher hatten Sie eine einzelne, selbst gewählte Zelle. Ab heute arbeiten Sie mit einem simulierten Pack aus acht Zellen. Ein Modul liefert die Zellspannung – abhängig von Ladezustand **und** Stromstärke:

```
from zelle import lese_spannung

u = lese_spannung(3, 0.5, 1.0)   # Zelle 3, Ladezustand 50 %, 1.0 A
print(u)
```

`lese_spannung(nummer, ladezustand, strom_A)`: `nummer` von 1 bis 8, `ladezustand` von 1.0 (voll) bis 0.0 (leer). Je höher der Strom, desto stärker sackt die Spannung ein – der Innenwiderstand der Zelle. Gleiche Eingaben ergeben immer dieselbe Ausgabe.

So bekommen Sie das Modul

1. Herunterladen: https://raw.githubusercontent.com/DavidMStraub/teaching-materials/main/programmieren/zelle.py
2. Datei speichern **im selben Ordner** wie Ihr Skript – nicht in einem Unterordner
3. Testen Sie mit den zwei Zeilen von der vorigen Folie

Fehlermeldung `ModuleNotFoundError: No module named 'zelle'`? Dann liegt die Datei am falschen Ort – prüfen Sie den Ordner.

Aufgabe 1: Eine Zelle entladen

Simulieren Sie, wie die Spannung einer Zelle sinkt, während sie entladen wird. Die Zelle hat eine Kapazität von `kapazitaet_Ah = 3.5`; pro Zeitschritt `zeitschritt_h = 0.1` sinkt der Ladezustand um `strom_A * zeitschritt_h / kapazitaet_Ah`.

1. Legen Sie Variablen an: `ladezustand = 1.0` (voll), `strom_A` mit einem Wert Ihrer Wahl, sowie `kapazitaet_Ah` und `zeitschritt_h` wie oben
2. Schreiben Sie eine `while`-Schleife: Solange `ladezustand` über 0 liegt, lesen Sie die Spannung aus (`lese_spannung(1, ladezustand, strom_A)`), geben Sie Ladezustand und Spannung aus (`print`) – und verringern Sie danach `ladezustand` gemäß der Formel oben
3. Bei welchem Ladezustand fällt die Spannung am stärksten ab?

4. Verdoppeln Sie `strom_A` – was ändert sich am Verlauf, was bleibt gleich?

Aufgabe 2: Grenzwerte prüfen

1. Schreiben Sie eine Funktion `bewertung(spannung_V)`, die ein Urteil zurückgibt (`return`): unter 3,0 V ⟶ `kritisch`, sonst ⟶ `ok`
2. Bauen Sie das Urteil als dritte Spalte in Ihre Ausgabe aus Aufgabe 1 ein
3. Bei welchem Ladezustand wird Zelle 1 kritisch – bei 1,0 A? Bei 2,0 A?

Aufgabe 3: Alle acht Zellen vergleichen

1. Wählen Sie einen festen Ladezustand und einen festen Strom und lesen Sie die Spannung aller acht Zellen aus – mit einer Schleife über die Zellnummern
2. Bestimmen Sie die **niedrigste** Spannung und die **Nummer** der zugehörigen Zelle (zwei Merker, ohne Liste)
3. Wiederholen Sie mit einem deutlich höheren Strom: Limitiert dieselbe Zelle das Pack – oder eine andere?

Aufgabe 4: Die schwächste Zelle bei verschiedenen Ladezuständen

1. Packen Sie Ihre Lösung aus Aufgabe 3 in eine Funktion `schwaechste_zelle(ladezustand, strom_A)`, die Nummer und Spannung der schwächsten Zelle zurückgibt (`return`)
2. Rufen Sie die Funktion bei mehreren Ladezuständen auf (z. B. 1.0, 0.5, 0.2 und 0.05) und geben Sie jedes Ergebnis aus (`print`)
3. Bleibt es über den ganzen Verlauf dieselbe Zelle, oder wechselt es?

Aufgabe 5 (falls Sie Listen schon hatten): Der Entladeverlauf als Liste

1. Simulieren Sie die Entladung von Zelle 1 wie in Aufgabe 1, sammeln Sie die Spannungen aber in einer Liste statt sie nur auszugeben
2. Schreiben Sie eine Funktion `minimum(werte)`, die den kleinsten Wert einer Liste zurückgibt (`return`) – ohne `min()`
3. Wenden Sie sie auf Ihre Liste an: Stimmt das Ergebnis mit Ihrer Beobachtung aus Aufgabe 2 überein?

Zusatz für Schnelle

- Schreiben Sie `kapazitaet_bis_grenze(nummer, strom_A, grenze_V)` : simuliert die Entladung in kleinen Schritten und gibt zurück (`return`), bei welchem Ladezustand die Zelle die Grenze unterschreitet

- Vergleichen Sie alle acht Zellen mit dieser Funktion, bei niedrigem und bei hohem Strom: Ändert sich die Rangfolge?
- Öffnen Sie `zelle.py` und lesen Sie den Code – was verstehen Sie schon, was noch nicht?