

LU01.A12 - Trace Table für die Collatz-Folge

Erstellen Sie einen Trace Table für einen Algorithmus mit `while`-Schleife, bei dem sich der Parameter selbst laufend verändert – und vergleichen Sie ihn mit der rekursiven Fassung desselben Algorithmus.

Erklärung der Collatz-Folge

Die Collatz-Folge (auch $(3n+1)$ -Problem) folgt einer verblüffend einfachen Regel. Man startet mit einer positiven ganzen Zahl und wiederholt:

- Ist die Zahl **gerade**, halbiere sie.
- Ist die Zahl **ungerade**, multipliziere sie mit 3 und addiere 1.

Man hört auf, sobald die Zahl 1 erreicht ist. Ob das für *jede* Startzahl passiert, ist bis heute unbewiesen – ausprobieren kann man es trotzdem.

Algorithmus

```
def collatz(n):
    schritte = 0
    while n != 1:
        if n % 2 == 0:
            n = n // 2
        else:
            n = 3 * n + 1
        schritte += 1
    return schritte

if __name__ == '__main__':
    start = 6
    print(f'Von {start} bis 1 sind es {collatz(start)} Schritte.')
```

Aufgabe

- Analysieren Sie den Code Schritt für Schritt für den Startwert $n = 6$.
- Erstellen Sie einen Trace Table mit dem Wert von **n** **vor** und **nach** dem Schritt, dem Ergebnis der Bedingung und dem Zählerstand.
- Verwenden Sie die folgende Struktur:

Schritt	n (vorher)	n % 2 == 0	angewandte Regel	n (nachher)	schritte
1	6	Ja	$n // 2$	3	1

Schritt	n (vorher)	n % 2 == 0	angewandte Regel	n (nachher)	schritte
2					

- Führen Sie die Tabelle fort, bis die Schleifenbedingung $n \neq 1$ nicht mehr erfüllt ist.

Beispielinput

```
start = 6
```

Beispieloutput

Von 6 bis 1 sind es 8 Schritte.

Teil 2: Dieselbe Berechnung ohne veränderliche Variablen

Die folgende Fassung berechnet dasselbe, verändert aber keine einzige Variable. Der Zustand wandert stattdessen durch die Parameter der rekursiven Aufrufe.

```
def collatz_rekursiv(n, schritte=0):  
    if n == 1:  
        return schritte  
    if n % 2 == 0:  
        return collatz_rekursiv(n // 2, schritte + 1)  
    return collatz_rekursiv(3 * n + 1, schritte + 1)
```

Erstellen Sie auch dafür einen Trace Table:

Aufruf	n	schritte	nächster Aufruf	Rückgabewert
1	6	0	collatz_rekursiv(3, 1)	
2				

Hinweis zur Rückgabespalte

Füllen Sie die Spalte *Rückgabewert* erst von unten nach oben aus. Der innerste Aufruf liefert seinen Wert zurück, und dieser Wert wird von jedem darüberliegenden Aufruf unverändert weitergereicht.

Auswertungsfragen

- Welche Spalten des ersten Trace Tables werden im zweiten überflüssig? Was sagt das über den Unterschied zwischen veränderlichem Zustand und Parameterübergabe aus?
- Im ersten Algorithmus bezeichnet n nacheinander acht verschiedene Zahlen. Im zweiten hat jedes n innerhalb seines Aufrufs genau einen Wert und behält ihn. Warum ist die zweite

Variante leichter nachzuvollziehen?

3. Was passiert bei `collatz(0)`? Erklären Sie, warum – und welche der beiden Fassungen den Fehler früher sichtbar macht.
4. Prüfen Sie Ihre Tabelle mit dem Startwert $n = 7$. Wie viele Schritte sind es? Warum braucht eine so kleine Zahl so viele Schritte?

Abgabe

Geben Sie beide ausgefüllten Trace Tables und die beantworteten Auswertungsfragen in Moodle ab.

 © Kevin Maurizi

From:

<https://wiki.bzz.ch/> - **BZZ - Modulwiki**

Permanent link:

<https://wiki.bzz.ch/modul/m323/learningunits/lu01/aufgaben/tracetable3?rev=1787040656>

Last update: **2026/08/18 10:10**

