LU06a - Collections - Grundlagen

Eine Sammlung (engl. Collection) gruppiert mehrere Variablen, die einen logischen Zusammenhang haben.

- Jedes Element einer Collection hat eine verwandte Aufgabe.
- Die Anzahl der Elemente kann während der Programmausführung ändern.
- Das Programm will die Elemente durchsuchen oder sortieren

https://www.techopedia.com/definition/25317/collection

Verwendung von Collections

Sobald ein Programm mit einer grossen Datenmenge arbeitet, ist die Verwendung von einzelnen Variablen nicht mehr sinnvoll. Anhand eines Beispiels siehst du den Unterschied zwischen Einzelvariablen und einer Liste.

Kleinste Zahl finden

Ein Programm erhält als Input fünf Zahlen. Es soll aus diesen fünf Zahlen die kleinste Zahl finden. Um das Problem ohne eine Liste zu lösen, musst du eine ganze Reihe von `if` -Bedingungen schreiben.

Pseudocode: Kleinste Zahl finden

```
def smallest_number():
    number1 = 5
    number2 = 15
    number3 = 2
    number4 = 37
    number5 = 3
    smallest = 0

    if number1 <= number2 and number1 <= number3 and number1 <= number4 and
number1 <= number5:
        smallest = number1
    elif number2 <= number1 and number2 <= number3 and number2 <= number4
and number2 <= number5:
        smallest = number2
    elif number3 <= number1 and number3 <= number2 and number3 <= number4
and number3 <= number5:
```

```
        smallest = number3
    elif number4 <= number1 and number4 <= number2 and number4 <= number3
and number4 <= number5:
        smallest = number4
    else:
        smallest = number5
    print(f'Kleinste Zahl: {smallest}')

if __name__ == "__main__":
    smallest_number()
```

Mehr als 5 Variablen

Vielleicht muss dieses Programm erweitert werden, damit es neu neun Zahlen verarbeiten kann. Damit die `if`-Bedingungen nicht unendlich lang werden, wollen wir eine andere Technik anwenden:

1. Speichere die grösstmögliche Zahl in die Variable `smallest`.
2. Prüfe ob die n.-Zahl kleiner ist als `smallest`.
    - Falls Ja: Überschreibe den Wert von `smallest` mit dem Wert der n.-Zahl
3. Wiederhole Schritt 2 für die nächste Zahl

```
    ...
    smallest = float('inf')  # Kleinste bisher gefundene Zahl, Annahme:
Grösstmögliche Zahl zum Start

    if number1 < smallest:
        smallest = number1
    if number2 < smallest:
        smallest = number2
    if number3 < smallest:
        smallest = number3
    if number4 < smallest:
        smallest = number4
    if number5 < smallest:
        smallest = number5
    # ...
    # if number99 < smallest:
    #     smallest = number99
    print(f'Kleinste Zahl: {smallest}')
```

Diese Programmvariante ist *ganz nett*, solange sie nicht mit tausenden von Zahlen arbeiten müssen. Bei grossen Datenmengen wäre es von Vorteil, man könnte mittels einer Schleife die Zahlenreihen abarbeiten. Genau an diesem Punkt kommen Collections zum Einsatz.

Kleinste Zahl mittels Liste finden

Anstatt für jede Zahl eine einzelne Variable zu definieren, definieren wir eine Liste von Zahlen. Eine

Liste kann beliebig viele Werte (Elemente) enthalten.

Mit einer `for … in …`-Schleife können wir alle Elemente der Liste verarbeiten. Bei jedem Durchlauf der Schleife wird das nächste Element aus der Liste genommen und in einer Variable gespeichert.

```
def smallest_list():
    smallest = float('inf')
    numbers = [5, 15, 2, 37, 3, 18, 21]
    for number in numbers:
        print(f'Prüfe die Zahl {number}')
        if number < smallest:
            smallest = number
    print(f'Kleinste Zahl: {smallest}')
```

Diese Lösung benötigt nicht nur weniger Codezeilen, sie funktioniert auch mit beliebig vielen Zahlen.

M319-LU06, M319-C1E