Lösungsalgorithmen & Programmieren

- Block 06 /Abend 06-

Repetition und Vertiefung

🎯 Leistungs-/Lernziele dieses Blocks:

Die Studierenden …

repetieren und vertiefen die bisher erlernten Elemente

01) Eingabeschleife mit Abbruch (while + if)
Schreibe ein Programm, das den Benutzer so lange nach einer ganzen Zahl fragt, bis eine Zahl im Bereich von 1 bis und mit 10 eingegeben wird. Gib am Schluss die gültige Zahl aus.

02) Schleife mit break und continue
Lies fortlaufend ganze Zahlen ein

- negative Zahlen werden ignoriert (mit continue)
- bei 0 wird die Eingabe abgebrochen (mit break)
- alle positiven Zahlen sollen zu einer Summe addiert werden

Am Ende gibst du die Summe aus.

03) Wochentag prüfen – Version mit vielen Bedingungen
Schreibe eine Eingabeschleife, die so lange nach einem Wochentag fragt, bis einer der folgenden Kürzel eingegeben wird: "MO", "DI", "MI", "DO", "FR", "SA", "SO"
Verwende dazu keine Liste, sondern eine Bedingung mit or: if tag == "MO" or tag == "DI" or …
Am Ende gibst du "Gültiger Tag" aus.

04) Wochentag prüfen – Version mit Liste
Schreibe die gleiche Funktionalität wie in Aufgabe 3, aber jetzt mit einer Liste gueltige = [...] und einer Bedingung: if tag in gueltige:

05) Nur einmal ausgeben – ohne Set
Gegeben ist: zahlen = [3, 1, 3, 2, 1, 4, 3]
Gib jede Zahl höchstens einmal aus – aber ohne set() zu benutzen. Verwende eine zusätzliche Liste gesehen = [] und prüfe mit if zahl not in gesehen:

06) Nur einmal ausgeben – mit Set
Löse die gleiche Aufgabe wie in 5) diesmal aber mit einem Set.
Hinweis:Mit set(zahlen) bekommst du die eindeutigen Werte.

07) Listentricks & Alias vs. Kopie

Gegeben ist: noten = [5.0, 4.5, 5.5, 4.0]

1. Erstelle eine Alias-Variable alias und eine Kopie kopie.
2. Füge bei alias mit append() die Note 6.0 hinzu.
3. Gib noten und kopie aus.

Frage: Welche Liste hat sich verändert, welche nicht? (kurze Textantwort)

08) Arbeiten mit Listen-Methoden

Gegeben ist: werte = [9, 3, 7, 1, 8]

Schreibe Code, der:

1. die Liste aufsteigend sortiert,
2. die Liste danach umkehrt,
3. das erste Element entfernt,
4. die Länge der Liste ausgibt.
5. Den Durchschnitt der Zahlen ausgibt

09) Verschachtelte Strukturen

Gegeben ist: daten = ("Luzern", [12, 14, 16], ("CH", "EU"))

Schreibe eine Zeile, die die Zahl 14 ausgibt.

Schreibe eine zweite Zeile, die das Kürzel "EU" ausgibt.

Schreibe eine dritte Zeile, welche die Bezeichnung “Luzern” in

10) Sets-Methoden anwenden

Erzeuge aus der Liste namen = ["Anna", "Ben", "Anna", "Clara"] ein Set.

Füge "Tom" mit add() hinzu.

Entferne "Ben" mit remove().

Entferne "Max" so, dass kein Fehler entsteht (verwende discard()).

11) Dictionary mit Listen als Value

Gegeben ist:

```
noten = {
   "Anna": [5.0, 5.5],
   "Ben":  [4.0],
   "Clara":[5.5, 5.0, 5.5]
}
```

Schreibe Code, der:

1. für jede Person den Notendurchschnitt berechnet,
2. die Ausgabe z.B. so macht:

```
Anna: 5.25
Ben: 4.0
Clara: 5.33
```

12) Dictionary – get() und pop()

Gegeben ist: staedte = {"Luzern": 25, "Zug": 23, "Bern": 22}

1. Lies die Temperatur von "Zug" mit get() aus.
2. Versuche, die Temperatur von "Basel" mit get() auszulesen, gib 0 zurück, falls nicht vorhanden.
3. Entferne "Bern" mit pop() und speichere den entfernten Wert in einer Variable.

13) Code → PAP / Struktogramm (Reverse Engineering)

Gegeben ist:

```
x = 0
while x < 5:
    x = x + 1
    if x % 2 == 0:
        print("gerade:", x)
    else:
        print("ungerade:", x)
```

Aufgabe:

- Beschreibe in 1–3 Sätzen, was der Algorithmus macht.
- Zeichne dazu ein Nassi-Shneiderman-Struktogramm.

14) Funktion mit Parameter und Rückgabewert

Erstelle die Funktion: def ist_gerade(n):
Die Funktion soll True zurückgeben, wenn n gerade ist, sonst False. Teste die Funktion mit ein paar Werten.

15) Funktion mit optionalem Parameter

Erstelle die Funktion: def begruessung(name, formell):
Wenn formell == True: gibt die Funktion aus: "Guten Abend, " + name, sonst: "Hoi " + name
Wenn man den zweiten Parameter "formell" nicht übergibt, soll er per Standard True sein.
Teste z.B.: begruessung("Anna")
begruessung("Ben", formell=False)

16) Funktion mit variabler Parameterliste (*args)

Erstelle die Funktion produkt() so, dass folgende Aufrufe möglich sind:

```
print(produkt(2, 3))
print(produkt(2, 3, 4))
print(produkt(2, 3, 4, 5, 6, 7, 8))
```

Die Funktion soll das Produkt aller übergebenen Zahlen berechnen (also Multiplikation, nicht Summe).

17) Anonyme Funktionen (lambda)

Gegeben ist: woerter = ["Banane", "Kiwi", "Apfel", "Melone"]
Sortiere die Liste nach der Länge der Wörter mit sorted() und key=lambda .... Gib die sortierte Liste aus.

18) Lambda als Parameter
Erstelle die Funktion: def anwenden(funktion, wert):

Die Funktion soll funktion(wert) ausführen und das Resultat zurückgeben.
Teste mit:

- einer lambda, die eine Zahl verdoppelt
- einer lambda, die zu einem String "!!!" anhängt.

19) Mischung: for + if + Listen + Funktionen
Erstelle die Funktion: def filter_gerade_und_quadrat(liste):
Die Funktion soll:

1. alle geraden Zahlen aus liste auswählen
2. von diesen Zahlen die Quadrate berechnen (n ** 2)
3. die neue Liste mit den Quadraten zurückgeben

Beispiel:
print(filter_gerade_und_quadrat([1,2,3,4])) # liefert zurück [4, 16]

20) Programmcode lesen und verstehen
Lösen sie ohne Hilfe von Software: Was gibt dieses Programm aus?

```
x = 4
y = 7
if x < y:
    y = x
elif y == 7:
    x = 0
print(x, y)
```

21) Programmcode lesen und verstehen
Lösen sie ohne Hilfe von Software: Was gibt dieses Programm aus?

```
noten = [4.5, 5.0, 3.5, 5.5]
a = noten
b = noten.copy()

a.append(6.0)
b.remove(3.5)
b.sort()

print(noten)
print(b)
```

22) Programmcode lesen und verstehen

Lösen sie ohne Hilfe von Software: Was gibt dieses Programm aus?

```
werte = (1, 2, 3, 4)
liste = list(werte)

liste[1] = liste[1] * 10
liste.append(2)

s = set(liste)

print(liste)
print(len(s))
```

23) Programmcode lesen und verstehen

Lösen sie ohne Hilfe von Software: Was gibt dieses Programm aus?

```
daten = {
    "Anna": [5.0, 5.5],
    "Ben":  [4.0, 4.5]
}

def bonus(noten):
    kopie = noten.copy()
    for i in range(len(kopie)):
        if kopie[i] < 5.0:
            kopie[i] += 0.5
    return kopie

neu = {}
for name, liste in daten.items():
    neu[name] = bonus(liste)

print(daten["Ben"])
print(neu["Ben"])
```