Kapitel 4: Algorithmische Grundstrukturen und ihre Darstellung

Bisher haben wir hauptsächlich Programme gesehen, bei denen der Programmcode Zeile für Zeile nacheinander ausgewertet wird. Mit Hilfe **algorithmischer Grundstrukturen** können wir diesem Verhalten abweichen.

Algorithmische Grundstrukturen steuern dabei den Ablauf eines Programms. Mit ihnen könnt ihr Anweisungsblöcke definieren, die nur unter einer bestimmten Bedingung ausgeführt werden oder auch solche, deren Ausführung mehrfach erfolgt. Hierfür unterscheiden wir **Sequenzen**, **Verzweigungen** und **Schleifen**, welche wir uns - inklusiv dreier **Darstellungsformen** - nachfolgend genauer anschauen.

4.8.3. Fußgesteuerte Schleife

Bei der **fußgesteuerten Schleife** wird eine Sequenz so lange wiederholt ausgeführt, **bis eine Bedingung (am Ende der Schleife) erfüllt ist**.

Im **Pseudocode** könnte man exemplarisch folgendes formulieren:

```
WIEDERHOLE
    ergebnis = ergebnis + 1
    zaehler = zaehler + 1
BIS zaehler >= 5
```

Am Ende eines jeden Schleifendurchlaufs wird nun überprüft, ob die Schleifenbedingung gültig ist.

Ist dies **nicht der Fall**, wird die Schleife erneut durchlaufen. **Ist die Bedingung jedoch gültig, wird die Wiederholung abgebrochen.**

Da die Bedingung erst am Ende jedes Schleifendurchlaufs überprüft wird, wird so eine Schleife **immmer mindestens einmal ausgeführt**, selbst wenn die Bedingung bereits zu Beginn erfüllt war.

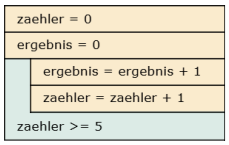
An sich gibt es diese Struktur in Python nicht. Allerdings können wir sie uns konstruieren: Mit Hilfe einer **Verzweigung** überprüfen wir, ob die Schleifenbedingung gültig ist. Wenn wir mit der Verzweigung feststellen, dass die Schleifenbedingung endlich gültig ist, können wir mit dem Schlüsselwort `break` die wiederholte Ausführung der Schleife verhindern und diese abbrechen.

Damit die Schleife allerdings überhaupt wiederholt ausgeführt werden kann, ist die eigentliche Bedingung hinter dem `WHILE`-Schlüsselwort `True`.

Programmcode:

In [ ]:
```
zaehler = 0
ergebnis = 0
while True:
    ergebnis = ergebnis + 1
    zaehler = zaehler + 1
    if zaehler >= 5:
        break
```

Struktogramm:



Die Darstellung ist ähnlich zur kopfgesteuerten Schleife, allerdings steht die Bedingung nun unten.

Hinweise zum Code einer fußgesteuerten Schleife:

- Jede fußgesteuerte Schleife beginnt mit dem Schlüsselwort `while`.
- Statt einer Bedingung folgt nach `while` das Schlüsselwort `True`. Dies entspricht in `PYTHON` dem Wahrheitswert (Boolean) wahr, d. h. statt eine Bedingung zu überprüfen, wird hier sofort das Ergebnis einer solchen Bedingung angegeben: `True`. Theoretisch würde diese Schleife also nie abbrechen.
- Der Schleifenkörper funktioniert analog zu einer normalen kopfgesteuerten Schleife: alle innerhalb eines Schleifendurchlaufs auszuführenden Anweisungen müssen gleich weit eingerückt werden.
- Die Überprüfung der Endbedingung erfolgt mittels einer Verzweigung. Ist die angegebene Bedingung erfüllt, wird die Anweisung break ausgeführt. Dies bewirkt einen Abbruch der gesamten Schleife. Nach dem Abbruch wird wie gewohnt die erste nicht eingerückte Anweisung ausgeführt.

Beispiel 1 - Wiederholte Satzausgabe:

In [ ]:
```
satz = input("Welcher Satz soll wiederholt ausgegeben werden? ")
eingabe = input("Wie oft soll der Satz ausgegeben werden? ")
anzahl = int(eingabe)
i = 1

if anzahl > 0:
    while True:
        print(satz)
        i += 1
        if i > anzahl:
            break

print("Damit ist das Programm beendet.")

# FÜHRE MICH AUS! :)
```

Beispiel 2 - Alle gerade Zahlen von 0 bis 10:

In [ ]:
```
n = 0
while True:
    print(n)
    n = n + 2
    if n > 10:
        break

# FÜHRE MICH AUS! :)
```

Das Schlüsselwort `break`:

Das Schlüsselwort `break` wird nicht nur in fußgesteuerten Schleifen genutzt. Mit Hilfe dieses Schlüsselworts kann **jede beliebige Art von Schleife** abgebrochen werden. Nachfolgend ist dies anhand der kopfgesteuerten Schleife visualisiert.

In [ ]:
```
eingabe = -1
i = 0
n = 2
while i < n:
    if eingabe < i:
        break
    eingabe = eingabe + (i * n)
    i = i + 1
print(eingabe)

# FÜHRE MICH AUS! :)
```

4.8.4. Übungen zur fußgesteuerten Schleife

Aufgabe 1

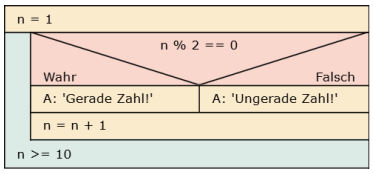
Implementiere den folgenden Pseudocode.

```
Eingabe einer Zahl n
Setze die Variable Zähler auf 1
WIEDERHOLE
    Erhöhe Zähler um 0,5
BIS der Zähler größer n ist
Ausgabe des Zählers
```

In [ ]:
```
# HIER IST PLATZ FÜR DEINE LÖSUNG! :)
```

Aufgabe 2

Implementiere das folgende Struktogramm.



Zu beachten ist, dass die letzte Anweisung `n = n + 1` über die gesamte Breite geht. Das heißt, dass diese Anweisung **NICHT MEHR** zur Verzweigung gehört!

In [ ]:
```
# HIER IST PLATZ FÜR DEINE LÖSUNG! :)
```

Aufgabe 3

Implementiere ein Programm, welches nach Eingabe einer oberen Grenze n, alle Zahlen von 0 bis n aufaddiert.

Bei der Eingabe der Zahl 3 ergibt sich beispielsweise das Ergebnis 6.

Gerechnet wird dabei: $0 + 1 + 2 + 3 = 6$.

Implementiere deine Lösung **zwingend** mit Hilfe einer **fußgesteuerten Schleife**!

In [ ]:
```
# HIER IST PLATZ FÜR DEINE LÖSUNG! :)
```

Aufgabe 4

Implementiere ein Programm, welches den Benutzer auffordert, eine Zahl zu erraten.

Die fußgesteuerte Schleife soll so lange fortgesetzt werden, bis der Benutzer die richtige Zahl erraten hat.

Gib dem Benutzer nach jeder Eingabe eine Rückmeldung, ob die Zahl zu hoch oder zu niedrig ist.

Hinweis: Verwende eine feste Zahl im Code, die der Benutzer erraten muss.

In [ ]:
```
# HIER IST PLATZ FÜR DEINE LÖSUNG! :)
```

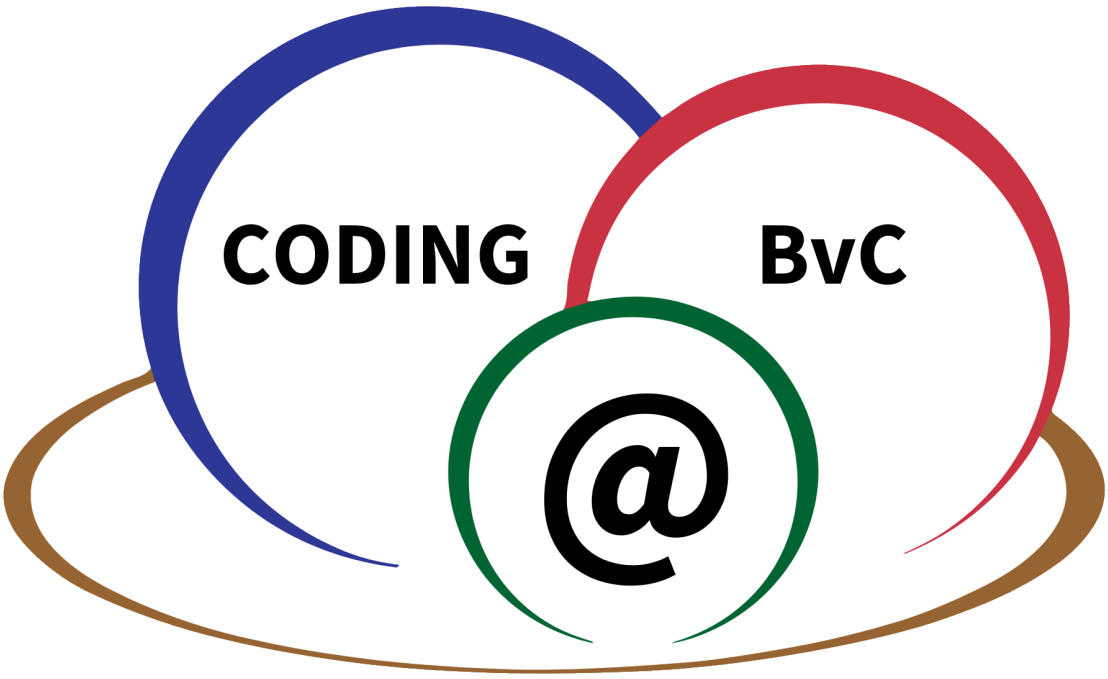
Aufgabe 5

Implementiere ein Programm, welches die Zahlwörter eins, zwei, ... , sechs so oft ausgibt, wie es das Wort aussagt, also eins einmal, zwei zweimal.

Implementiere deine Lösung mit Hilfe einer **fußgesteuerten Schleife**! Ein **korrekt funktionierendes Programm** erzeugt die folgende Ausgabe:

```
eins
zwei zwei
drei drei drei
vier vier vier vier
fünf fünf fünf fünf fünf
sechs sechs sechs sechs sechs sechs
```

In [ ]:
```
# HIER IST PLATZ FÜR DEINE LÖSUNG! :)
```


CODING
BvC
@