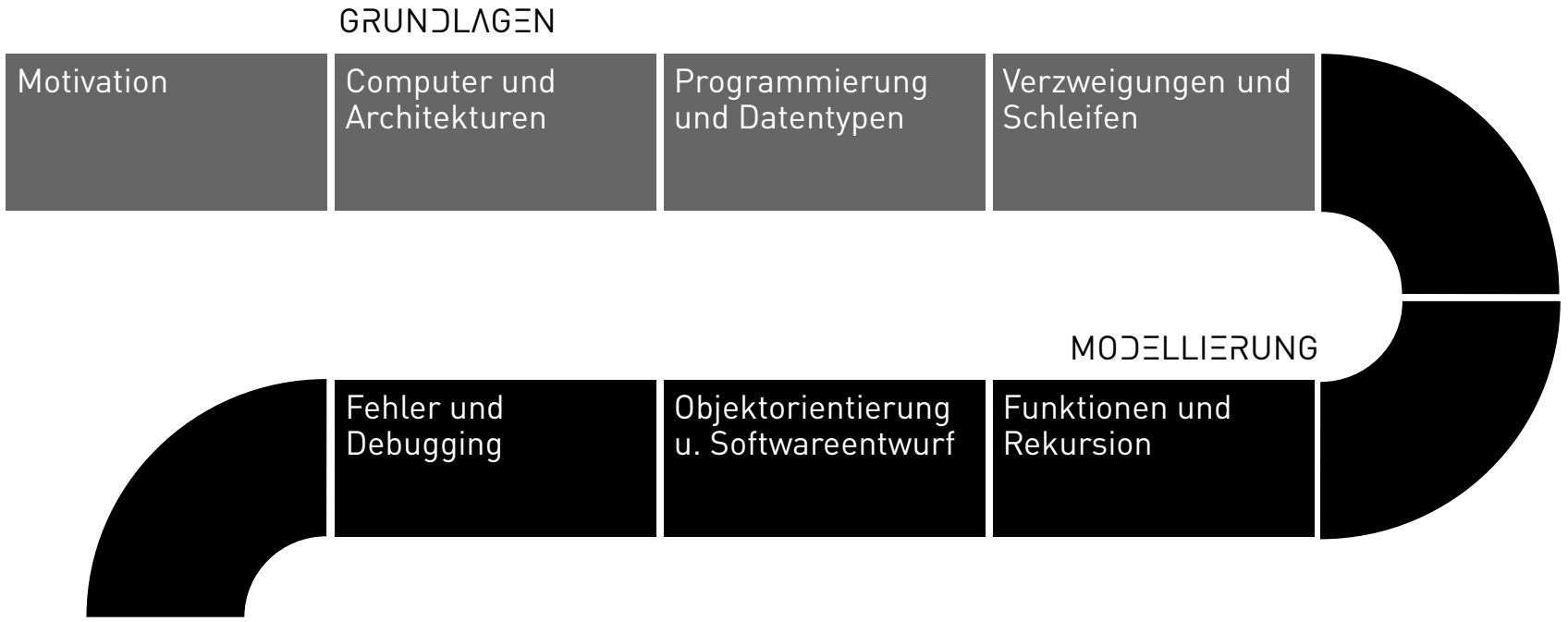
programmierung und datenbanken

Joern Ploennigs

Schleifen

MIDJOURNEY: LOOPS, REF. ROBERT DELAUNAY

Ablauf


GRUNDLAGEN
Motivation
Computer und Architekturen
Programmierung und Datentypen
Verzweigungen und Schleifen
MODELLIERUNG
Funktionen und Rekursion
Objektorientierung u. Softwareentwurf
Fehler und Debugging

Schleifen

Schleifen sind ein wichtiger Ansatz in der Programmierung, um wiederholende oder inkrementelle Lösungen zu realisieren.
In Python sind verfügbar:

- `for` (Entspricht For-Each-Loop)
- `while` (Entspricht While-Loop)

Wichtige Eigenschaften:

- Funktionieren genau wie alle anderen Steueranweisungen durch Einrückungen nach einer einzeiligen Definition
- Anders als in Funktionen sind Variablen aus Schleifen auch außerhalb der Schleife verfügbar!
- Die anderen Loop-Varianten sind zwar nicht explizit vorhanden, aber funktionell nachbildbar

Schleifen - FOR-Loop

Der for-Loop wiederholt ein Block an Statements für alle Elemente in einer Sequenz. Element ist dabei eine Variable die immer mit dem aktuellen Element aus der Sequenz belegt wird.

```
for Element in Sequenz:
    # Statement
```

Schleifen - FOR-Each-Loop Beispiel

Der for-Loop in Python ist ein For-Each-Loop. Er iteriert immer durch eine Sequenz an Werten, wie z.B. eine Liste. Die Variable des Iterators `e` enthält dabei den aktuellen Wert der Liste `seq`.

```
seq = ['a','b','c']
for e in seq:
    print(e)
```

Ausgabe:

```
a
b
c
```

Schleifen - FOR-Loop Beispiel

Der for-Loop in Python kann auch als klassischer For-Loop verwendet werden, bei der man n-mal etwas wiederholt. Dazu erzeugt man mit der `range(n)` -Funktion eine Folge an Zahlen, durch die dann das For-Each-Loop iteriert wird.

```
n = 3
seq = range(n)
for e in seq:
    print(e)
```

Ausgabe:

```
0
1
2
```

Schleifen - FOR-Loop Beispiel

Damit kann man dann zum Beispiel eine Liste an Messwerten aus imperialen Fuß in metrische Meter umwandeln.

```
measurements_feet = [4.2, 2.3, 6.2, 10.5]   # Eingabe
measurements_meter = []   # Ergebnisse

for feets in measurements_feet:
    measurements_meter.append(0.3048 * feets)

print(measurements_meter)
```

Ausgabe:

```
[1.2801600000000002, 0.70104, 1.88976, 3.2004]
```

Schleifen - WHILE-Loop

Die while-Loop-Schleife wird so lange ausgeführt so lange bis eine Bedingung wahr ist. Da diese am Anfang geprüft wird und von Anfang an falsch sein kann, muss der Inhalt nicht ausgeführt werden.

```
while Bedingung:
    # Statement
```

Wie beim if wird die Bedingung auf seinen Wahrheitswert geprüft:

- Ist dieser `True` läuft die Schleife einmal durch, woraufhin wieder geprüft wird
- Ist dieser `False` endet die Schleife und der Code wird nicht (noch einmal) ausgeführt

Schleifen - While-Loop vs. Do-While-Loop vs. Repeat-Until

While-Loop

Im While-Loop kann die Bedingung schon am Anfang falsch sein. Die Schleife muss also nicht ausgeführt werden.

```
Bedingung = True/False
while Bedingung:
    # Statement
    Bedingung = False
```

Do-While-Loop

Im Do-While-Loop wird die Bedingung erst am Ende geprüft. Die Schleife wird also immer mindestens einmal durchlaufen.

```
Bedingung = True
while Bedingung:
    # Statement
    Bedingung = False
```

Repeat-Until-Loop

Beim Repeat-Until-Loop wird die Bedingung auch erst am Ende geprüft. Die Schleife wird wiederholt bis die Bedingung wahr wird.

```
Bedingung = False
while not Bedingung:
    # Statement
    Bedingung = True
```

Programmflusskontrolle

Wir kennen nun zwei Werkzeuge:

Bedingte Ausführung:

- `if`, `else`, `elif`

Mehrfachausführung:

- `while`, `for`

Jedes Programm, was jemals auf einem Computer ausgeführt wurde, könnte mit diesen Mitteln geschrieben werden.

PROGRAMMIERUNG UND DATENBANKEN