



EINFÜHRUNG IN PROGRAMMIERUNG UND DATENBANKEN

joern ploennigs

grundlagen

Motivation

Computer und
Architekturen

Programmierung
und Datentypen

Verzweigungen und
Schleifen

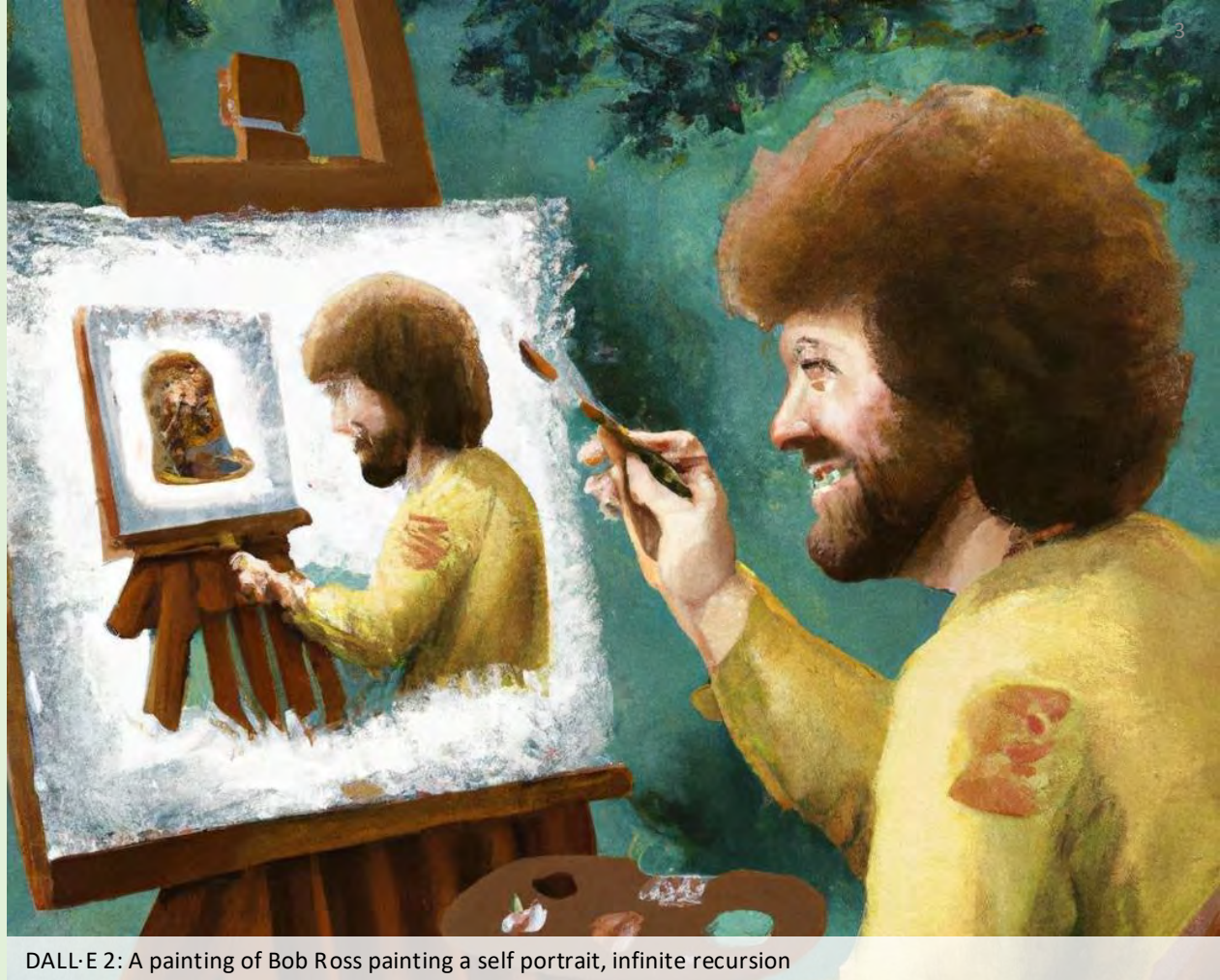
modellierung

Fehler und
Debugging

Objektorientierung u.
Softwareentwurf

Funktionen und
Rekursion

VERZWEIGUNGEN



DALL·E 2: A painting of Bob Ross painting a self portrait, infinite recursion

ZIELSETZUNG

- Heute wollen wir uns anschauen wie man den Ablauf der Statements dynamisch verändern kann
– speziell wie man bedingte Verzweigungen nutzen kann.
- Außerdem: Wie plant und konzeptioniert man den Ablauf eines Programmes?

PROGRAMMABLAUF

- Programme können als Abfolge von durchnummerierten Statements betrachtet werden.
- Diese werden Zeile für Zeile ausgeführt.
- Erinnerung: Statements in Funktionen werden erst ausgeführt wenn die Funktion gerufen wird.

PROGRAMMABLAUF - BEISPIEL

```
1. x = 3
2. y = 4
3.
4. def pythagoras(a, b):
5.     return sqrt((a**2)+(b**2))
6.
7. z = pythagoras (x, y)
8. print(z)
```

```
1. # Zuweisung
2. # Zuweisung
3.
4. # Abspeichern der Funktion
5.
6.
7. # Funktionsaufruf
5. # Berechnung
7. # Zuweisung zu z
8. # Variablenausgabe
```

Output: 5

VERZWEIGUNGEN - BEDINGTES VERZWEIGEN

- Wir können Steueranweisungen benutzen, die basierend auf einer Bedingung ein bestimmtes Statement „ansteuern“.
- Diese Bedingungen sind Statements deren Ausgabe ein Boolean-Wert (True oder False) ist.
- Beispiele für solche Statements:
 - True # Logische Werte
 - a # Variablen Werte, wenn a mit einem Bool belegt wurde, bzw. das Bool-Äquivalent von a
 - 2 < 5 # Alle Vergleichsoperatoren

VERZWEIGUNGEN – WENN ... DANN ...

- Die Grundform solcher Verzweigungen ist also immer:
„Wenn die Bedingung zutrifft, dann führe folgende Statements aus“
- Oft folgt außerdem ein „sonst führe stattdessen folgendes Statement aus“
- Python kennt drei Anweisungen zur Verzweigung:
 - `if` `# wenn`
 - `else` `# sonst`
 - `elif` `# kurz für else-if`

VERZWEIGUNGEN - IF-STATEMENT

- Die If-Anweisung ist nötig um eine Verzweigung zu beginnen.
- Die Notation folgt demselben Prinzip wie bei Funktionsdefinitionen, aber ohne Namen:

if Bedingung:

dann führe den folgenden Block nur aus wenn die Bedingung wahr ist

Einrücken →

Statement1

Statement2 ...

- Ist die Bedingung True, werden die folgenden eingerückten Statements ausgeführt.

VERZWEIGUNGEN - BEISPIEL: IF-STATEMENT ZUM ÜBERSPRINGEN VON CODE

```
activateOutput = True
```

```
a = 10
```

```
b = 13
```

```
c = pythagoras(a, b)
```

```
if activateOutput:
```

```
    print("Seitenlänge:" + c)
```

```
umfang = a + b + c
```

```
...
```

VERZWEIGUNGEN - ELSE-STATEMENT

- else folgt nur auf ein if.
- Wird ausgeführt wenn die im if angegebene Bedingung false zurückgibt.
- Funktioniert ebenso über Einrückung:

if **Bedingung:**

dann führe den folgenden Block nur aus wenn Bedingung wahr ist

Statement1

Statement2 ...

else:

Statement1b ...

VERZWEIGUNGEN - BEISPIEL: DIVISION DURCH 0 ABFANGEN (EINFACHE VERZWEIGUNG)

a = 5

b = 0

if b: # der Wahrheitswert eines int-Wertes ist false bei 0 und Wahr für != 0

 c = a / b

else:

 print("Division by Zero")

 c = None

VERZWEIGUNGEN - BEISPIEL: DIVISION DURCH 0 ABFANGEN (EINFACHE VERZWEIGUNG)

14

```
a = 5
```

```
b = 0
```

```
if b != 0:
```

```
    c = a / b
```

```
else:
```

```
    print("Division by Zero")
```

```
    c = None
```

VERZWEIGUNGEN - VERSCHACHTELTE VERZWEIGUNGEN

- Durch mehrfaches Einrücken können wir Verzweigungen in Verzweigungen definieren

```
if Condition1:
```

```
    StatementA
```

```
else:
```

```
    if Condition2:
```

```
        StatementB
```

```
    else:
```

```
        StatementC
```

```
def sign(a)
    if a < 0:
        print("Number is negative")
    else:
        if a > 0:
            print("Number is positive")
        else:
            print("Number is zero")
```

VERZWEIGUNGEN - MEHRFACHVERZWEIGUNG: ELIF-STATEMENT

- Anstatt einer else-Anweisung können wir die elif-Anweisung verwenden, welche wie if eine Bedingung hat.
- Auf eine elif-Anweisung können folgen:
 - Eine weitere elif-Anweisung
 - Else-Anweisung - wird aufgerufen wenn keine der bisherigen Anweisungen zutraf
 - Keine Anweisung – zweite Variante vom optionalen Code


```
def sign(a)
    if a < 0:
        print("Number is negative")
    elif a > 0:
        print("Number is positive")
    else:
        print("Number is zero")
```

VERZWEIGUNGEN - NEU IN PYTHON 3.10: MATCH-STATEMENTS

- Programmiersprachen entwickeln sich stetig weiter.
- Anstatt viele `elif`-Anweisungen mit dem `==` Operator, geben wir eine Variable ein, deren Wert gegen mehrere Optionen geprüft wird.

```
if Variable == ValueA:  
    StatementA  
elif Variable == ValueB:  
    StatementB  
else:  
    StatementC
```



```
match(Variable)  
    case ValueA:  
        StatementA  
    case ValueB:  
        StatementB  
    case _:  
        StatementC
```

Hörsaalfrage

FRAGEN?



DALL·E 2: A psychedelic DJ with a question mark for a head