Programmierung mit Python

Heutige Lernziele:

- Bekanntes aus Bash in Python umsetzen
  - Variablen (+ was sind Datentypen?)
  - if-Strukturen
  - for-Schleifen

TECH STARTER

Variablen

- Speichern einen Wert
- Wir können keine Variablen nutzen, die wir vorher nicht definiert haben (in Bash: undefinierte Variablen sind einfach leer)
- Python ist case-sensitive, auch bei Variablennamen
- Erfordern in Python keine Dollarzeichen!

Variablen

Python hat 33 reservierte Wörter

```
and
del
from
None
True
as
elif
global
nonlocal
try
assert
else
if
not
while
break
except
import
or
with
class
False
in
pass
yield
continue
finally
is
raise
def
for
lambda
return
```

Datentypen

- Jeder Wert in Python hat einen Datentypen

```
zahl   = 8                    # int = Integer (Ganze Zahlen)
minus  = -395234

komma  = 3.5                  # float = Float   (Kommazahlen)

text   = "endlich Freitag"    # str = String  (Zeichenketten)
wort   = 'Wochenende'         # Kein Unterschied der Anführungszeichen

wahr   = True                 # bool = Boolean (Wahrheitswert)
falsch = False
```

Funktionen

- Wir nutzen in Python typischerweise keine Befehle wie in der Bash
- Stattdessen nutzen wir Funktionen
- Aktuell nutzen wir nur Built-In Funktionen
- Morgen schreiben wir eigene Funktionen

TECH STARTER

Funktionen - Beispiele

```
user_input = input("Schreibe ein Wort: ") # Nimmt Nutzereingabe an
input_type = type(user_input) # Ermittelt den Datentypen von user_input

print("Typ:", input_type)
print("Nachricht:", user_input)
```

- input() : Nimmt Nutzereingaben an
- type() : Ermittelt den Datentypen eines Wert/einer Variable
- print() : Gibt etwas in der Konsole aus

Funktionen - Datentypen Casting

```
pi         = 3.141    # Eine Kommazahl, also von Typ Float
pi_string = str(pi)   # str() : Zu String casten
pi_int     = int(pi)  # int() : Zu Integer casten

print("Name Typ Value")
print("pi", type(pi), pi)
print("pi_string", type(pi_string), pi_string)
print("pi_int", type(pi_int), pi_int)
```

Mit beispielsweise str() und int() können wir den Typen einer Variable ändern, wenn die Änderung sinnvoll ist.

TECH STARTER

Listen

```
leere_liste = []
einkaufsliste = ["Äpfel", "Zucker", "Mehl"]
wirre_liste = [ 2, True, False, -5.5, "Sushi"]

print(einkaufsliste[1]) # Zugriff auf einzelne Felder wie in Bash
print(wirre_liste[3])
print( len(leere_liste) ) # len() ermittelt Länge von Liste
```

Listen sind sehr ähnlich zu Arrays in Bash.

Wir können einzelne Felder abrufen, oder die Länge abfragen.

TECH STARTER

if-Abfragen

```
if 5 == "5":
    print("Ja, die sind gleich!")
    # ...
else:
    print("Nein, sind verschieden")
    # ...
```

Kommentare:

- Genau wie in Bash

- Keine besonderen Klammern mehr
- Doppelpunkt kündigt tiefere Einrückung an

TECH STARTER

for-each-Schleifen

```
list = [ 0, 1, 2, 3, 4]

print("Erste Schleife")
for zahl in list:
    print(zahl)
```

- Wie in Bash können wir über die Elemente einer Liste bzw. Eines Arrays durchiterieren und für jedes Element eine bestimmte Aktion ausführen, hier: print()

TECH STARTER

range() - Funktion

```
print("\nZweite Schleife") # "\n erzeugt Zeilenumbruch"
for i in range(0,5):
    print(i)
```

- Mit range() können wir Listen mit Zahlen einfach erzeugen
- range(A,B) erzeugt eine Art Liste von A bis B - 1
- range(0,5) erzeugt also die Zahlen 0, 1, 2, 3, 4