Datenstrukturen + Kommentare

Variablen und Datentypen

- Eine Variable ist ein Speicherplatz für Werte.
- Ein Wert hat einen Datentyp.

```
a = 7
b = 9.2
string = 'Text'
c = True
```

Datentyp	Wertebereich	Beispiel
`int`	Ganze Zahlen	`11`
`float`	Reelle Zahlen	`3.14`
`str`	Zeichenfolgen	`"Katze"`, `"starcode*"`
`boolean`	`True`, `False`	`True`

Listen

- Eine Liste ist ein Speicherplatz für mehrere Werte.

> 💡 Python gibt dem 1. Element der Liste den Index 0.

```
liste = ["H", "A", "L", "L", "O"]

print(len(liste))

liste[1] = "E"
print(liste)

liste.append("!")
print(liste)
```

```
5
['H', 'E', 'L', 'L', 'O']
['H', 'E', 'L', 'L', 'O', '!']
```

Kommentare

- Ein Kommentar ist ein Text-Abschnitt im Code, der nicht ausgeführt wird.

```
# Das ist ein einzeiliger Kommentar
```

```
# Diese Funktion berechnet die Summe von a und b
...
```

```
"""
Dies ist ein mehrzeiliger Kommentar
"""
```

Lösungen zum Kurs unter:
github.com/Perimora/Starcode
↓
README
↓
Lösungen
↓
launch binder

Funktionen

print

- Die print() Anweisung gibt Werte auf der Konsole aus.

```
print('Hallo')
```

```
Hallo
```

```
print('42')
```

```
42
```

```
print('Ich bin ' + str(17) + ' Jahre alt')
```

```
Ich bin 17 Jahre alt
```

input

- Die input() Anweisung erlaubt dem Benutzer, eine Eingabe zu machen.

```
zahl = input('Bitte gib eine Zahl ein.')
```

```
Bitte gib eine Zahl ein.
```

```
bst = input('Bitte gib einen Buchstaben ein.')
```

```
Bitte gib einen Buchstaben ein.
```

Funktionen

- Eine Funktion erlaubt das Strukturieren in einen wiederverwendbaren Code-Block.

```
def addition(zahl1, zahl2):
      ergebnis = zahl1 + zahl2
      return ergebnis
```

```
print(addition(3,7))
print(addition(12,49))
```

```
10
61
```

```
def verabschieden(name):
  print('Auf wiedersehen ' + name + '!')
```

```
verabschieden('Weihnachtsmann')
verabschieden('Osterhase')
```

```
Auf wiedersehen Weihnachtsmann!
Auf wiedersehen Osterhase!
```

Kontrollstrukturen

if else

- Die if-else Kontrollanweisung ermöglicht es, verschiedene Code-Blöcke abhängig von einer Bedingung auszuführen.

```
regnet = False

if regnet == True:
    print('Du brauchst du einen Regenschirm.')
else:
    print('Du brauchst KEINEN Regenschirm!')
```

```
Du brauchst KEINEN Regenschirm!
```

```
alter = 20

if alter < 18:
    print("Du bezahlst den ermäßigten Preis.")
elif alter < 65:
    print("Du bezahlst den regulären Preis.")
else:
    print("Du bezahlst den Senioren-Preis.")
```

```
Du bezahlst den regulären Preis.
```

Vergleichsoperator	Bedeutung
`>`	größer als
`<`	kleiner als
`>=`	größer oder gleich
`<=`	kleiner oder gleich
`==`	gleich
`!=`	ungleich

while

- Die while-Schleife ermöglicht das wiederholte Ausführen eines Code-Blocks, solange eine Bedingung erfüllt ist.

```
antwort = "ja"
while antwort != "nein":
    antwort = input("Weitermachen? (ja/nein): ")
```

```
Weitermachen? (ja/nein):
```

```
x = 1
while x <= 3:
    x += 1
    print(x)
```

```
2
3
4
```

for

- Die for-Schleife ermöglicht das wiederholte Ausführen eines Code-Blocks, bis eine bestimmte Anzahl an Durchgängen erreicht ist.

> 💡 Python beginnt mit i = 0.

```
for i in range(4):
    print(i)
```

```
0
1
2
3
```

```
wort = ["H", "A", "L", "L", "O"]

for i in range(len(wort)):
    print(i,". Buchstabe: ", wort[i])
```

```
0 . Buchstabe:  H
1 . Buchstabe:  A
2 . Buchstabe:  L
3 . Buchstabe:  L
4 . Buchstabe:  O
```

```
wort = ["H", "A", "L", "L", "O"]

for buchstabe in wort:
    print(buchstabe)
```

```
H
A
L
L
O
```

```
zahlen_liste = [1,2,3,4]

for i in range(len(zahlen_liste)):
    zahlen_liste[i] = zahlen_liste[i] + 1

print(zahlen_liste)
```

```
[2, 3, 4, 5]
```

```
fruechte = ["Apfel", "Banane", "Kirsche"]
for frucht in fruechte:
    print("Ich mag", frucht)
```

```
Ich mag Apfel
Ich mag Banane
Ich mag Kirsche
```