Python: Woche 2

Eingabe, Verzweigungen, Schleifen

Michael Stambach
Stellvertretung Ivo Blöchliger

Themen Heute

- Eingabe
- Vergleiche
- Verzweigungen
- Schleifen

Aufgaben letzte Woche

- **Mehrheitlich gut gelöst**
- **Nicht ganz fertig – kein Problem**
- **Dateien speichern!**

Verwechslungsgefahr

Dateiformat:

- Wie ist eine Datei aufgebaut?
- Inhalt der Datei
- z.B. Textdatei

Dateierweiterung:

- Was steht nach dem Punkt im Dateinamen?
- Teil des Namens: gibt Format an, stimmt aber nicht zwingend!
- z.B. ".py", ".txt", ".pptx", ".zip", ...

Datentypen:

- Welcher Kategorie gehört ein Wert an?
- Variablen in Programmiersprachen haben immer einen bestimmten Datentyp
- z.B. "int", "float", "str", "bool", etc.

Eine Pythondatei

- **...ist als eine gewöhnliche Textdatei *formatiert***
- **...hat als *Dateiendung* ".py"**
- **...verwendet *Variablen* welche verschiedene *Datentypen* haben können**

Umgang mit Strings

```
print("Hello" + "World")
```

```
HelloWorld
```

- **Beim Zusammenketten gibt es keinen Abstand!**

```
a = 12
print(f"Das Resultat ist {a}")
```

```
Das Resultat ist 12
```

- **Bei f-Strings ist ein explizites Konvertieren mit `str()` nicht nötig!**

Eingabe

- Das Gegenteil von `print()` ist `input()`
- Der angegebene Text wird ausgegeben, auf dem Rest der Zeile kann der Nutzer etwas schreiben
- Die Eingabe wird immer als String zurückgegeben!

```
a = input("Geben sie etwas ein: ")
print(f"Sie haben {a} eingegeben!")
```

```
Geben sie etwas ein: hallo
Sie haben hallo eingegeben!
```

Eingabe von Zahlen

```
a = int(input("Geben sie eine Zahl ein: "))
print(f"{a} im Quadrat ist {a*a}")
```

```
Geben sie eine Zahl ein: 5
5 im Quadrat ist 25
```

Will man Zahlen einlesen, muss man sie explizit konvertieren!

Vergleiche

- **Mit diesen Operatoren lassen sich Variablen und Werte vergleichen**
- **Es wird dabei immer ein Wahrheitswert (bool) zurückgegeben**

Operator	Bedeutung
==	gleich
!=	ungleich
>	grösser
<	kleiner
>=	grösser oder gleich
<=	kleiner oder gleich

Vergleiche: Beispiele

```
a = 100
b = 111
c = "100"
```

Was ergeben diese Vergleiche?

`a == b`	**False**
`b >= a`	**True**
`(a+b) < 200`	**False**
`a != c`	**True**
`c > b`	**Error**
`a <= 100`	**True**

Logische Operatoren

- **Logische Werte (bools) lassen sich verknüpfen**
- **x and y: True falls x und y True sind**
- **x or y: True falls mindestens eines von x und y True ist**
- **not x: True falls x False ist**

Wahrheitstabelle

x	y	x and y	x or y	not x
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

Verzweigungen

- **Mit Wahrheitswerten können wir Code bedingt ausführen**
- **If-Statements**
- **Der eingerückte Code wird nur ausgeführt, wenn die Bedingung wahr (True) ist.**
- **Die Einrückung muss konsistent sein!**

```
a = 14
if a > 10:
  print("a ist grösser als 10")
  a -= 10
print(f"a={a}")
```

```
a ist grösser als 10
a=4
```

Ausgabe mit a=8:

```
a=8
```

"Sonst" in Verzweigungen

- **Mit else können wir "mache ... falls ... sonst mache ..." umsetzten.**
- **Der eingerückte Code unter else wird nur ausgeführt, wenn die Bedingung des zugehörigen ifs nicht wahr war.**

```
a = 14
if a > 10:
  print("a ist grösser als 10")
else:
  print("a ist kleiner oder gleich 10")
print(f"a={a}")
```

Beispiel: ist eine Zahl gerade?

- Mit dem Modulo-Operator erhält man den Divisionsrest.
- Wenn der Divisionsrest mit 2 gleich 0 ist, ist die Zahl gerade.
- *Was ist der Unterschied zwischen den beiden Programmen? Weshalb ist das else hier nötig?*

```
a = 14
if a % 2 == 0:
  print("a ist gerade")
else:
  print("a ist ungerade")
print(f"a={a}")
```

```
a = 14
if a % 2 == 0:
  print("a ist gerade")
print("a ist ungerade")
print(f"a={a}")
```

elif: Eine Mischung aus else und if

- Mit elif lassen sich zusätzliche Bedingungen mit einem else verknüpfen
- "elif" ist kurz für "else if", also "sonst falls"
- Wird ausgeführt falls die vorherigen Bedingungen False waren und die eigene True ist
- In einer Kette von ifs, elifs und else wird immer nur genau ein Abschnitt ausgeführt

elif: ein Beispiel

```
a = ??

if a > 20:
  print("A")
elif a >= 10:
  print("B")
else:
  print("C")
```

Was muss auf 'a' zutreffen, damit folgende Ausgaben erfolgen?

A: 'a' ist grösser als 20

B: 'a' ist zwischen 10 und 20 (inklusive)

C: 'a' ist kleiner als 10

Wiederholungen mit Schleifen

- **Mit einem ähnlichen Syntax wie bei den Verzweigungen lassen sich Schleifen erstellen**
- **Schleifen können einen Abschnitt mehrmals ausführen**
- **Es gibt zwei Arten von Schleifen**
  - while-Schleife: Wiederholt, so lange eine Bedingung wahr ist
  - for-Schleife: Wiederholt für jedes Element in einer Sammlung, die Anzahl Wiederholungen ist zu Beginn bekannt

while-Schleife: Ein wiederholtes if

- **Eine while-Schleife ist sehr ähnlich wie ein 'if' aufgebaut**
- **Unterschied: So lange die Bedingung zutrifft, wird der Abschnitt ausgeführt**
- **Gefahr von Endlosschleifen**
- *Was gibt der Code rechts aus?*

```
a = 5
while a > 0:
  print(a)
  a -= 1
print("fertig!")
```

```
5
4
3
2
1
fertig!
```

for-Schleife

- **Aufbau:**
  for <Element> in <Sammlung>
- **Der eingerückte Code wird für jedes Element in der Sammlung ausgeführt**
- **range(n) entspricht Zahlen von 0 bis n-1**
- **Die Sammlung könnte z.B. auch eine Liste sein**

```
for i in range(4):
  print(f"Wiederholung {i}")
print("fertig!")
```

```
Wiederholung 0
Wiederholung 1
Wiederholung 2
Wiederholung 3
fertig!
```

Aufgaben

- Diese und nächste Woche Programmieraufgaben
- Erstmal Code selber ausführen
- Kleine Programme die vervollständigt werden sollen
- Aufgaben: Kommentare in den Programmen
- Zusammenfassung auf dem Wiki
- Einrichten von VS Code für Python