ZUSAMMENFASSUNG Kontrollstrukturen

EINFÜHRUNG

Alles Skripte, die wir in diesem Kurs bisher ausgeführt haben, wurden der Reihe nach von oben nach unten abgearbeitet. Hierbei wurde die eine Anweisung ausgelassen oder wiederholt.
Doch in Python besteht diese Möglichkeit, indem man mit Fallunterscheidungen und Schleifen arbeitet. Diese Kontrollstrukturen helfen uns Entscheidungen im Programm zu treffen.
Hierfür arbeiten wir mit den Logischen- und Vergleichsoperatoren. Denn diese entscheiden darüber, ob ein bestimmter Code-Blick ausgeführt wird oder nicht oder ob er mehrmals wiederholt wird.

FALLUNTERSCHEIDUNGEN MIT IF UND ELSE

Eine if-Anweisung (deutsch = „Wenn“), startet immer mit dem Schlüsselwort if und wird gefolgt von der Bedingung in Form eines beliebigen Ausdrucks. Die Bedingung muss immer mit einem Wahrheitswert zu beantworten sein. Dann folgt ein Doppelpunkt mit einem zugehörigen Codeblock der eingerückt wird.
Alles was nach der if-Anweisung eingerückt steht, wird ausgeführt falls die Bedingung „True“ ergibt. Falls sie „False“ ergibt, wird die Anweisung nicht ausgeführt und einfach übersprungen und der Code läuft normal weiter.

Zum Beispiel:

```
number = int(input("Bitte gebe eine Ganzzahl ein: "))

if number < 10:
    print("Die eingegebene Zahl ist kleiner als 10 ...")
```

In diesem Beispiel ist die Bedingung der if-Anweisung, dass die eingegebene Zahl kleiner ist als 10. Falls diese Bedingung zutrifft, also „True“ ist, wird die eingerückte print-Anweisung ausgeführt.

Die Kombination von if und else

Die if-Anweisung kann nun durch den else-Zweig erweitert werden. Dieser wird ausgeführt, wenn die Bedingung der if-Anweisung False ist.

Verwenden wir hierfür nochmal unser Beispiel von oben. Wenn die eingegeben Zahl also nicht 10 ist, die Bedingung also „False“ ist, kann man nun noch definieren was in diesem Fall passieren soll, indem man eine else-Anweisung schreibt.

```
number = int(input("Bitte gebe eine Ganzzahl ein: "))

if number < 10:
   print("Die eingegebene Zahl ist kleiner als 10 ...")
else:
  print („Die eingegeben Zahl ist 10 oder größer“)
```

In diesem Beispiel wird nun durch die else-Anweisung definiert, dass „Die eingegeben Zahl ist 10 oder größer“ ausgegeben wird, wenn die eingegebene Zahl nicht kleiner also 10 ist.

Generell wird der else-Zweig ausgeführt, wenn die Bedingung der if-Anweisung false ist.

Die Kombination von if, elif und else

Nun kann unser Beispiel noch weiter um einen elif-Zweig erweiter werden. Elif ist eine Kombination aus else und if. Hiermit ist es möglich noch eine weitere Bedingung abzufragen.

Zum Beispiel:

```
if number == 1:
    print(„Die eingegebene Zahl ist die 1“)
elif number == 2:
    print(„Die eingegebene Zahl ist die 2“)
elif number == 3:
     print(„…“)
```

Man kann also so viele elif-Zweige wie man möchte erzeugen. Außerdem kann man am Ende auch einen else-Zweig setzen, der alles andere abfängt was nicht durch die Bedingung abgedeckt ist.

Bedinge Ausdrücke

Mit Hilfe eines bedingten Ausdrucks (englisch: conditional expression) kann eine if else Anweisung in nur einer Zeile dargestellt werden.

A if-Bedingung else B

➔ Das bedeutet: Mache A wenn die Bedingung true ist, ansonsten mache B

Zum Beispiel:

```
bill = 20

tip = float(input("Rechnung: " + str(bill) + " Euro, wie viel Trinkgeld möchten Sie geben: "))

if tip < bill * 0.1:
   print("Vielen Dank")
else:
   print("Wow vielen lieben Dank, ich wünsche Ihnen einen wunderschönen Tag!")
```

Kann vereinfacht werden zu einer Zeile:

print("Vielen Dank") if tip < bill * 0.1 else print("Wow vielen lieben Dank!!!")

WICHTIG

Bei der bedingten Anweisung muss die Bedingung ein logischer Ausdruck sein, sodass der Ausgabewert True oder False sein kann. Zudem wird immer zuerst die Bedingung ausgewertet und je nachdem ob sie true oder false ist wird dann A oder B ausgewertet.

DIE WHILE SCHLEIFE

Eine Schleife besteht immer aus einem Schleifenkopf, in welchem die Bedingung für die Schleife definiert ist. Am Anfang steht das Schlüsselwort while, gefolgt von der Bedingung und einem Doppelpunkt. Aus der Bedingung muss wieder ein Wahrheitswert also True oder False zurückgegeben werden.
Nach dem Schleifenkopf folgt der Schleifenkörper, also alles was nach dem Doppelpunkt eingerückt ist. Die dortigen Anweisungen, können nun mehrmals hintereinander ausgeführt werden.

Zum Beispiel:

```
while 2 < 10:
  print („Das ist eine Ausgabe…“)
```

Zuerst wird geprüft ob die Bedingung der Schleife wahr ist. Falls sich der Wahrheitswert True ergibt, wird der Schleifenkörper nochmal ausgeführt. Das läuft so lange weiter, bis der Wahrheitswert False entsteht, dann wird der Schleifenkörper nicht nochmal ausgeführt. Die Schleife ist also beendet und der Code läuft normal weiter.
Im obigen Beispiel wurde eine Endlosschleife erstellt, da 2 immer kleiner ist als 10.

SCHLÜSSELWÖRTER BREAK UND CONTINUE

Diese Schlüsselwörter können in Kombination mit einer while- oder for- Schleife eingesetzt werden.
Das Schlüssewort „break“ ist beispielsweise sinnvoll, wenn wir die Eingabe eines Nutzers limitieren wollen. Kombiniert man eine if-Schleife mit „break“ kann man eine Schleife sofort beenden.

Das Schlüsselwort „continue” funktioniert ähnlich wie „break“. Hier wird die Schleife allerdings nicht sofort beendet, sondern der aktuelle Schleifendurchlauf des Schleifenkörpers. Der Rest des Schleifenkörpers wird einfach übersprungen und die Schleifenbedingung wird erneut überprüft.

In Python gibt es die Besonderheit, dass while Schleifen mit else kombiniert werden können. Else wird dabei nicht ausgeführt, wenn eine Schleife durch „break“ vorzeitig beendet wurde.

DIE FOR SCHLEIFE

Die for Schleife kann auf zwei unterschiedliche Arten verwendet werden.

1. Man kann mit ihrer Hilfe über ein iterierbares Objekt laufen
2. Die for Schleife kann in Kombination mit der Funktion range() als Zählerschleife eingesetzt werden

Die for-Schleife für ein iterierbares Objekt

Ein iterierbares Objekt ist etwas, das man durchlaufen kann. In Python ist das alles, was unter den Datentyp str oder list fällt.

Zum Beispiel:

```
for element in [2, 4, 6, 8]:
    print(element)
```

Der Schleifenkopf beginnt wieder mit dem Schlüsselwort „for“, worauf der Bezeichner folgt. Danach kommt das Schlüsselwort „in“ und ein iterierbares Objekt. Der Schleifenkopf endet wieder mir einem „:“, worauf der eingerückte Schleifenkörper folgt. Dieser enthält die Anweisungen, die bei jedem Schleifendurchlauf ausgeführt werden sollen.

Mit einer for-Schleife wird jedes Element eines iterierbaren Objekts mit einem Schleifendurchlauf durchlaufen. Durch den Bezeichner, kann man im jeweiligen Schleifendurchlauf auf das aktuell durchlaufene Element zugreifen.
Im Fall einer Liste, greift man also auf jedes Objekt einer Liste zu. Im Falle eines Strings, auf jeden einzelnen Buchstaben.

Die for-Schleife als Zählerschleife

Der Aufbau ist exakt der gleiche. Allerdings schreibt man anstatt des iterierbaren Objekts die Funktion range() hin, mit der man Parameter übergeben kann.

Zum Beispiel:

```
for element in range(10):
    print(element)
```

Mit diesem Programm kann man sich die Zahlen 0 bis 0 ausgeben lassen. Der Endwert, also 10, wird nicht ausgeben da er exkludiert wird. Der Startwert wird immer übergeben.

Mit einem zusätzlichen dritten Wert in der Klammer, kann man die Größe des Zählerschritts bestimmen. Standardmäßig ist das 1. Mit range (3, 10, 2) gibt die Schleife 3, 5, 7, 9 aus.
Das Ganze funktioniert auch mit negativen Schritten für die Rückwärtszählung.

Es ist zudem möglich, bei einer for-Schleife „break“, „continue“ oder auch „else“ zu verwenden.