LU07b - While-Else-Anweisung in Python

In Python kann die `while`-Anweisung eine optionale `else`-Klausel enthalten:

```
while condition:
    # code block to run
else:
    # else clause code block
```

In dieser Syntax wird die `condition` zu Beginn jeder Iteration geprüft. Der Codeblock innerhalb der `while`-Anweisung wird so lange ausgeführt, wie die `condition` `True` ist.

Wenn die `condition` `False` wird und die Schleife normal läuft, wird der `else`-Block ausgeführt. Wird die Schleife jedoch vorzeitig durch eine `break`- oder `return`-Anweisung beendet, wird der `else`-Block nicht ausgeführt.

Das folgende Flussdiagramm veranschaulicht die `while…else`-Schleife:

Wenn Sie mit anderen Programmiersprachen wie JavaScript, Java oder C# vertraut sind, werden Sie feststellen, dass der `else`-Block im Zusammenhang mit einer Schleife recht merkwürdig ist.

Allerdings erweist sich der `while else`-Block in einigen Fällen als sehr nützlich. Werfen wir einen Blick auf ein Beispiel für die Verwendung der `while else`-Anweisung.

Einfaches Beispiel

```
# list of fruits
my_list =
['papaya','banana','pineapple','mango','grape
s']

size = len(my_list) #length/size of the list
i=0

# iterating through the fruit list
while i<size:
   if my_list[i] == 'mango':
       print('mango found!')
       break
   i+=1
else:
   print('mango not found!')
```

Komplexeres Beispiel

Angenommen, wir haben die folgende Liste von Früchten, in der jede Frucht ein `Dictionary` ist, das aus den Schlüsseln `fruit name` und `qty` besteht:

```
basket = [
    {'fruit': 'apple', 'qty': 20},
    {'fruit': 'banana', 'qty': 30},
    {'fruit': 'orange', 'qty': 10}
]
```

Wir wollen ein Programm erstellen, das es den Benutzern erlaubt, einen Obstnamen einzugeben. Anhand des eingegebenen Namens suchen wir die Frucht in der `basket`-Liste und zeigen die Menge an, wenn die Frucht in der Liste enthalten ist.

Falls die Frucht nicht gefunden wird, können die Benutzer die Menge für diese Frucht eingeben und sie zur Liste hinzufügen.

Das folgende Programm ist der erste Versuch:

```
basket = [
    {'fruit': 'apple', 'qty': 20},
    {'fruit': 'banana', 'qty': 30},
    {'fruit': 'orange', 'qty': 10}
]

fruit = input('Enter a fruit:')

index = 0
found_it = False

while index < len(basket):
    item = basket[index]
    # check the fruit name
    if item['fruit'] == fruit:
        found_it = True
        print(f"The basket has {item['qty']}
{item['fruit']}(s)")
        break

    index += 1

if not found_it:
    qty = int(input(f'Enter the qty for
{fruit}:'))
```

```
    basket.append({'fruit': fruit, 'qty':
qty})
    print(basket)
```

Beachten Sie, dass es eine bessere Möglichkeit gibt, dieses Programm zu entwickeln. Das Programm in diesem Beispiel dient lediglich der Demonstration.

Wie es funktioniert:

- Zuerst wird mit der Funktion `input()` eine Benutzereingabe abgefragt.
- Zweitens: Initialisieren Sie den `index` auf Null und das `found_it`-Flag auf False. Der `index` wird für den Zugriff auf die Liste nach Index verwendet. Und das `found_it`-Flag wird auf `True` gesetzt, wenn der Name der Frucht gefunden wird.
- Drittens: Iterieren Sie die Liste und prüfen Sie, ob der Name der Frucht mit dem eingegebenen Namen übereinstimmt. Wenn ja, setzen Sie das `found_it`-Flag auf `True`, zeigen Sie die Menge der Frucht an und verlassen Sie die Schleife mit der `break`-Anweisung.
- Prüfen Sie schließlich das Flag `found_it` nach der Schleife und fügen Sie die neue Frucht zur Liste hinzu, wenn `found_it` auf False steht.

Im Folgenden wird das Programm ausgeführt, wenn `apple` die Eingabe ist:

```
Enter a fruit:apple
The basket has 20 apple(s)
```

Und das folgende führt das Programm aus, wenn `lemon` die Eingabe ist:

```
Enter a fruit:lemon
Enter the qty for lemon:15
[{'fruit': 'apple', 'qty': 20}, {'fruit':
'banana', 'qty': 30}, {'fruit': 'orange',
'qty': 10}, {'fruit': 'lemon', 'qty': 15}]
```

Das Programm funktioniert wie erwartet.

Es wird jedoch übersichtlicher, wenn Sie stattdessen die `while else`-Anweisung verwenden.

Im Folgenden sehen Sie die neue Version des Programms, die die `while else`-Anweisung verwendet:

```
basket = [
    {'fruit': 'apple', 'qty': 20},
    {'fruit': 'banana', 'qty': 30},
    {'fruit': 'orange', 'qty': 10}
```

```
]

fruit = input('Enter a fruit:')

index = 0

while index < len(basket):
    item = basket[index]
    # check the fruit name
    if item['fruit'] == fruit:
        print(f"The basket has {item['qty']}
{item['fruit']}(s)")
        break

    index += 1
else:
    qty = int(input(f'Enter the qty for
{fruit}:'))
    basket.append({'fruit': fruit, 'qty':
qty})
    print(basket)
```

In diesem Programm ersetzt der `else`-Block die Notwendigkeit des `found_it`-Flags und der `if`-Anweisung nach der Schleife.

Wenn die Frucht nicht gefunden wird, wird die `while`-Schleife normal beendet und der `else`-Block wird ausgeführt, um eine neue Frucht zur Liste hinzuzufügen.

Wenn die Frucht jedoch gefunden wird, wird die `while`-Schleife auf die `break`-Anweisung stossen und vorzeitig beendet werden. In diesem Fall wird die `else`-Klausel nicht ausgeführt.

Der `else`-Block in der `while else`-Anweisung wird ausgeführt, wenn die Bedingung der `while`-Schleife `False` ist und die Schleife normal läuft, ohne auf die `break`- oder `return`-Anweisung zu stoßen.

Probieren Sie die Python `while else`-Anweisung immer dann aus, wenn Sie ein Flag in einer `while`-Schleife benötigen.

M319-LU07, M319-E1G, M319-E1F