Python - Zusammenfassung

Variablen

Variablen speichern Informationen, die im Programm verwendet werden. Sie haben einen Namen und können verschiedene Datentypen enthalten (Zahlen, Texte, Listen, Wahr/Falsch Aussagen usw.).

```
Meine_Variable = 42 # Beispiel für eine Variable, die eine Zahl speichert

text = "Hello"  # Beispiel für eine Variable, die einen Text speichert

running = True # Beispiel für eine Variable, die eine Wahr/Falsch Aussaage speichert

text_new = text #Variable, die den Wert der Variable text speichert
```

Listen

Listen sind geordnete Sammlungen von Werten. Sie können Zahlen, Texte, andere Listen und viele weitere verschiedene Datentypen enthalten. Jedes Element einer Liste steht an einer bestimmten Stelle und kann einzeln abgerufen oder geändert werden. Mit len(liste) erhält man die Länge einer Liste.

```
Shopping_List = ["Bread", "Milk", "Eggs"]  # Eine Liste mit Einkaufsartikeln

Shopping_List.append("Butter")  # Fügt "Butter" am Ende der Liste hinzu

First_Item = Shopping_List[0]
# Zugriff auf das erste Element der Liste (bei Elementen einer Liste beginnt die Zählung bei 0)

New_List = [1, True, "Hello", Shopping_List]  # Eine neue Liste mit verschiedenen Datentypen

print(len(New_List))  # Gibt die Länge der Liste New_List aus
```

Mathematik / Operatoren

Operatoren führen Rechen- oder Vergleichsoperationen durch. Es gibt arithmetische (+, –, *, /), logische (and, or, not) und Vergleichsoperatoren (==, !=, <, >).

```
x = 1

y = x * 9  # Multipliziert x mit 2 und speichert das Ergebnis in y
z = y / 3  # Teilt y durch 3 und speichert das Ergebnis in z
z = z + 1  # Erhöht z um 1

a = (y + z) * (x + 1)
```

Beispiel arithmetische Operatoren

```
x = 7
y = 3

print(y == x) # gibt False zurück, da y und x nicht gleich sind
print(y > 2 and x == 7) # gibt True zurück, da y größer als 2 und x = 7 ist
print(y > 2 and x == 60)  # gibt False zurück, da y zwar größer als 2, x aber nicht 60 ist
```

Beispiel logische / Vergleichsoperatoren

print() und input()

print() und input() sind in Python „eingebaute“ Funktionen.
print() gibt das, was zwischen den Klammern steht, im Terminal aus.
input() tut genau dasselbe, und wartet danach, bis der Nutzer im Terminal eine Antwort eingegeben hat.

```
text = "Hello World"  #Variable, die einen Text speichert

print(text)  #Text ausgeben

answer = input("Wie heißt du? ")  #Eingabeaufforderung für den Benutzer

print("Hallo " + answer + "!")  #Begrüßung des Benutzers mit der eingegebenen Antwort
```

If / Else

Bedingungen steuern den Ablauf des Programms. Je nach Wahrheitswert (`True` oder `False`) wird ein bestimmter Codeblock ausgeführt.

X == 5 ist im Beispiel unten beispielsweise erfüllt (True), weswegen der unter dem „if“ eingerückte Code ausgeführt wird. Wäre dies nicht der Fall würde der Code nicht ausgeführt werden.

```
x = 5
y = 5

if x == 5:  # Überprüft, ob x gleich 5 ist
    print("x ist gleich 5")  # Gibt eine Nachricht aus, wenn die Bedingung wahr ist

if x <= 10 and x > 0:  # Überprüft, ob x kleiner oder gleich 10 und größer als 0 ist
    print("x ist zwischen 0 und 10")  # Gibt eine Nachricht aus, wenn die Bedingung wahr ist
else: # falls nicht x kleiner oder gleich 10 und größer als 0 ist:
    if x > 3 or y > 3:
        print("x ist größer als 3 oder y ist größer als 3")
```

Wiederholungen

Schleifen wiederholen Anweisungen. Mit for oder while kann man Listen durchgehen oder Aktionen mehrfach ausführen. Die Beispiele unten sind eine 10-fache Wiederholung (oben), und eine Endlose Wiederholung, die aber trotzdem mithilfe von „break“ beendet werden kann (unten).

```
x = 0
while x < 10:  # Solange x kleiner als 10 ist
    print(x)  # Gibt den aktuellen Wert von x aus
    x += 1  # Erhöht x um 1

while True:  # Endlosschleife, da Bedingung immer erfüllt
    user_input = input("Gib etwas ein (oder 'x' zum Beenden)")
    if user_input.lower() == 'x':  # Überprüft, ob die Eingabe 'exit' ist
        print("Beende die Schleife.")
        break # Beendet die Schleife
```

Funktionen

Funktionen bündeln Code in Blöcke, die man mehrfach verwenden kann. Sie können Parameter (=Daten) annehmen und Werte zurückgeben (wie z.B add_numbers).

```
def greet(name): # Funktion, die einen Namen als Parameter nimmt
    print("Hallo " + name + "!")  # Gibt eine Begrüßung mit dem eingegebenen Namen aus

def add_numbers(a, b):  # Funktion, die zwei Zahlen addiert
    return a + b  # Gibt die Summe der beiden Zahlen zurück

greet("Valentin")  # Aufruf der Funktion greet mit dem Namen "Valentin"
result = add_numbers(5, 3)  # Aufruf der Funktion add_numbers mit
print("Das Ergebnis der Addition ist: " , result)  # Ausgabe des Ergebnisses der Addition
```

Hier ein etwas komplexeres Programm: Zuerst wir der Nutzer zum Eingeben einer beliebigen Zahl aufgefordert. Diese wird in der Variable input_number gespeichert. Danach wird diese Zahl 10 mal mit x multipliziert und das Ergebnis in der Konsole ausgegeben. X wird bei jeder Wiederholung um 1 erhöht. Gibt man bspw. 2 als input_number ein, ist der Output 2,4,6,8,10,12,14,16,18

```
def multiplyNumber(number, factor): #Funktion, die eine eingegebene Zahl quadriert
    print(f"Das Ergebnis von {number} mal {factor} ist: {number * factor}")  # Ausgabe des Ergebnisses
    return number * factor

input_number = int(input("Gib eine Zahl zum multiplizieren ein: "))

x = 1

while x < 10:  # Solange x kleiner als 10 ist
    result = multiplyNumber(input_number, x)  # Aufruf der Funktion
    x += 1  # Erhöht x um 1
```