

Grundlagen der Programmierung in Python

Unterrichtsdokumentation

Modulname:	Grundlagen der Programmierung in Python in der Sek 2
------------	--

Stand: 2. Jun 2020



© **BS-Wangen**

Inhaltsverzeichnis

1 Einführung in Python.....	3
1.1 Ausgabe von Zeichenketten.....	3
1.2 Variablen.....	4
1.3 Rechenoperationen.....	7
1.4 Eingabe mittels input - Kreis.....	11
1.5 Eingabe mittels input() - Rechtecke.....	13
1.6 Eingabe mittels input() - Höhenmeter Berechnung.....	16
1.7 Funktionen/Methoden – Urlaubsfahrt.....	20
1.8 Einseitige Verzweigung – Abiball.....	27
1.9 Zweiseitige Verzweigung – Preisvorschlag.....	32
1.10 Geschachtelte Verzweigung – Mietzuschuss.....	36
1.11 Verzweigung mit logischen Operatoren Eintritt Tierpark.....	41
1.12 Verzweigung mit logischen Operatoren Autovermietung.....	45
1.13 Wiederholungen mit der For-Schleife Taschengeld.....	48
1.14 Wiederholungen mit der While-Schleife Fischteich.....	52

1 Einführung in Python

1.1 Ausgabe von Zeichenketten

(I) Problemstellung

Schreiben Sie ein Python-Programm, das Ihre persönlichen Daten auf dem Bildschirm ausgibt. Speichern Sie dieses Programm als L2_1_Textausgabe.py ab.

(II) Problemanalyse

(1) So soll die Bildschirmausgabe des Programms aussehen:
Christine Janischek, Geb.20.02.1974, Hochgratblick 7, 88260 Argenbühl

(III) Struktogramm

Ausgabe: Christine Janischek, Geb.20.02.1974, Hochgratblick 7, 88260 Argenbühl

I

(IV) Programmcode (Python-Code)

```
L2_1_Textausgabe.py ×  
1 print("Christine Janischek, Geb. 20.02.1974, Hochgratblick 7, 88260 Argenbühl")
```

1.2 Variablen

(I) Problemstellung

Schreiben Sie ein Programm, das die Daten eines Schülers in mehreren Variablen speichert und hinterher ausgibt. Zu speichern sind: Name, Vorname, Alter, Notendurchschnitt, zweite Fremdsprache (True/False) und Musik (True/False). Diese Daten sollen dann tabellarisch im Browser ausgegeben werden.

(II) Problemanalyse

1. Welche Ausgabedaten will man erhalten?

```
Name: Janischek
Vorname: Christine
Alter: 46
Notendurchschnitt: 1.5
Zweite Fremdsprache: True
Musik: False
```

2. Welche Eingabedaten werden zur Bearbeitung benötigt?

name, vorname, alter, notendurchschnitt, zweite_fremdsprache, musik

3. Welche Eigenschaften haben Ausgabedaten? (Variablenliste)

Bedeutung	Inhalt	Variablenname
Textvariablen	"Janischek"	name
Numerische Variablen	1.5	notendurchschnitt
Wahrheitswerte	True	zweite_fremdsprache

(IV) Programmcode (Python-Code)

1. Teilergebnis

```
L2_1_Textausgabe.py × L2_2_Variablen_Kurswahl.py ×
1 # Deklaration und Initialisierung der Textvariablen
2 name="Janischek"
3 vorname="Christine"
4
5 # Deklaration und Initialisierung numerischer Werte
6 alter=46
7 notendurchschnitt=1.5
8
9 # Deklaration und Initialisierung der Wahrheitswerte
10 zweite_fremdsprache=True
11 musik=False
12
13
14 #Gewünschte Ausgabe
15 # Name: Janischek
16 # Vorname: Christine
17 # Alter: 46
18 # Notendurchschnitt: 1.5
19 # Zweite Fremdsprache: True
20 # Musik: False
21
22 print("Name: ", name)
23 print("Vorname: ", vorname)
24 print("Alter: ", alter)
25 print("Notendurchschnitt: ", notendurchschnitt)
26 print("Zweite Fremdsprache: ", zweite_fremdsprache)
27 print("Musik: ", musik)
```

2. Teilergebnis

```
L2_1_Textausgabe.py × L2_2_Variablen_Kurswahl.py ×
1 # Deklaration und Initialisierung der Textvariablen
2 nachname="Mustermann"
3 vorname="Max"
4
5 # Deklaration und Initialisierung numerischer Werte
6 alter=16
7 schnitt=1.8
8
9 # Deklaration und Initialisierung der Wahrheitswerte
10 zweit_fremd=True
11 musik=False
12
13
14 #Gewünschte Ausgabe
15 # Name: Janischek
16 # Vorname: Christine
17 # Alter: 46
18 # Notendurchschnitt: 1.5
19 # Zweite Fremdsprache: True
20 # Musik: False
21
22 print("nachname als zeichenkette: ", nachname)
23 print("vorname als zeichenkette: ", vorname)
24 print("alter als Ganzzahl: ", alter)
25 print("schnitt als Dezimalzahl: ", schnitt)
26 print("zweit_fremd als Wahrheitswert: ", zweit_fremd)
27 print("musik als Wahrheitswert: ", musik)
```

Ausgabe:

```
nachname als zeichenkette: Mustermann
vorname als zeichenkette: Max
alter als Ganzzahl: 16
schnitt als Dezimalzahl: 1.8
zweit_fremd als Wahrheitswert: True
musik als Wahrheitswert: False
```

1.3 Rechenoperationen

(I) Problemstellung

Entwerfen Sie ein Programm, das die Rechenoperationen plus, minus, mal, geteilt und hoch durchführt. Definieren Sie dazu zwei Variablen, die Sie für die Berechnungen nutzen und geben Sie das Ergebnis am Bildschirm aus.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

Für für die Addition:

Wert für Zahl 1: 12

Wert für Zahl 2: 3

Ergebnis: 15

(2) Welche Daten werden zur Bearbeitung benötigt?

Benötigte Variablen:

zahl1, zahl2, ergebnis

(3) Welche Eigenschaften haben Ausgabedaten? (Variablenliste)

Bedeutung	Inhalt	Variablenname
Eingabewert für Zahl 1	Numerischer Wert (Ganzzahl)	zahl1
Eingabewert für Zahl 2	Numerischer Wert (Ganzzahl)	zahl2
Ausgabewert für das Ergebnis	Numerischer Wert	ergebnis

(4) So soll die Bildschirmausgabe des Programms (erstes Teilergebnis) aussehen:

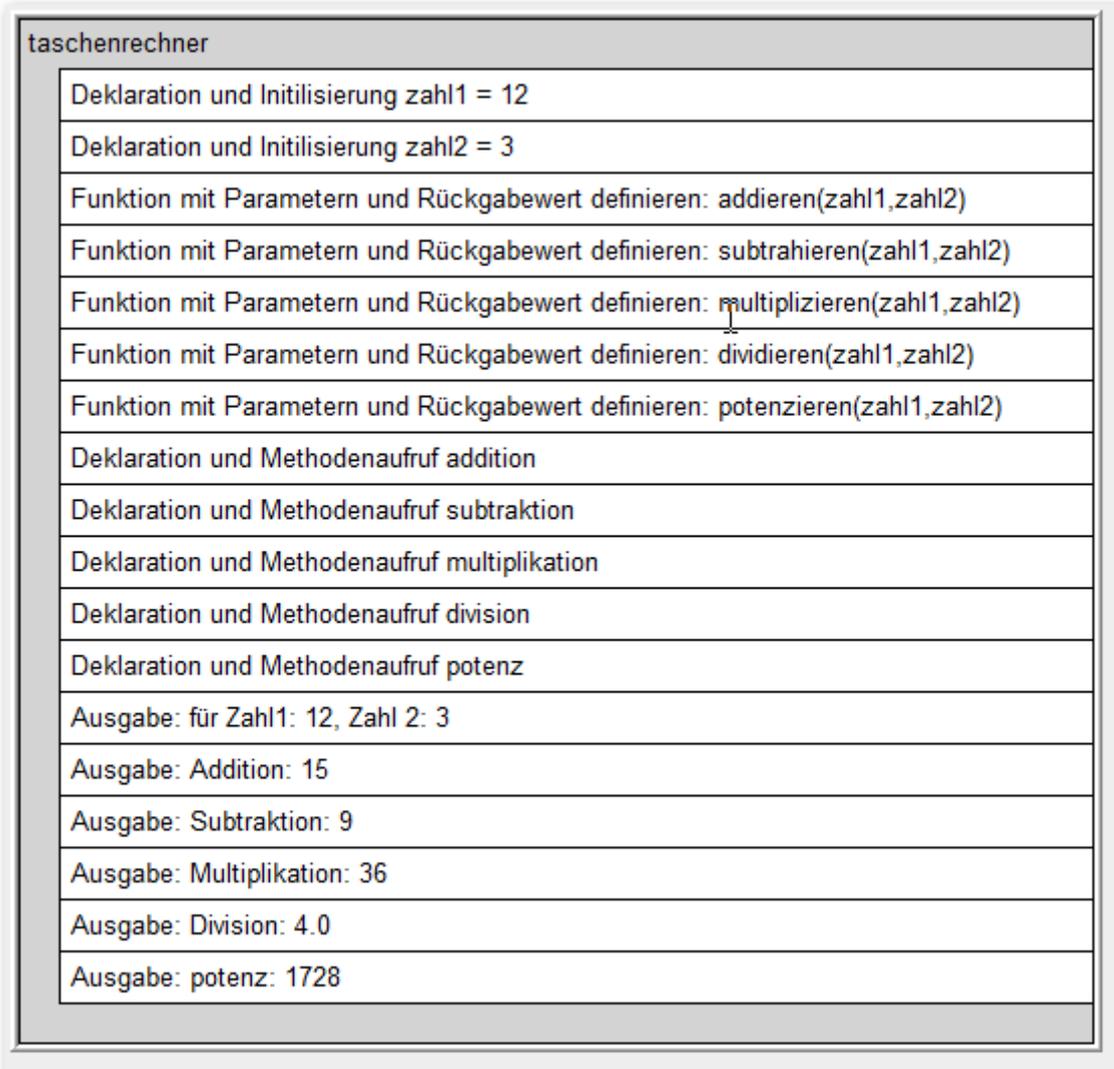
```
>>> %Run Rechenoperationen.py
Wert für Zahl 1: 12
Wert für Zahl 2: 3
Ergebnis: 15

>>> %Run Rechenoperationen.py
Wert für Zahl 1: 12
Wert für Zahl 2: 3
Ergebnis: 36
```

(5) Variieren Sie den Quellcode (zweites Teilergebnis) so, dass die Ausgabe wie folgt aussieht:

```
Zahl 1: 12 Zahl 2: 3
-----
Addition: 15
Subtraktion: 9
Multiplikation: 36
Division: 4.0
Potenz: 1728
```

Struktogramm:



Quellcode: zweites Teilergebnis

```
L2_1_Textausgabe.py × L2_2_Variablen_Kurswahl.py × Rechenoperationen_erstes Teilergebnis.py × Rechenoperationen.py * ×  
1 #Deklaration und Initialisierung von Eingabewerten (EINGABE)  
2 zahl1 = 12  
3 zahl2 = 3  
4 ergebnis = 0  
5  
6 #Funktionen für die einzelnen Rechenoperationen (VERARBEITUNG)  
7 def addieren(zahl1,zahl2):  
8     ergebnis = zahl1 + zahl2  
9     return ergebnis  
10  
11 def subtrahieren(zahl1,zahl2):  
12     ergebnis = zahl1 - zahl2  
13     return ergebnis  
14  
15 def multiplizieren(zahl1,zahl2):  
16     ergebnis = zahl1 * zahl2  
17     return ergebnis  
18  
19 def dividieren(zahl1,zahl2):  
20     ergebnis = zahl1 / zahl2  
21     return ergebnis  
22  
23 def potenzieren(zahl1,zahl2):  
24     ergebnis = zahl1 ** zahl2  
25     return ergebnis  
26  
27 # Rechenoperation durchführen: Methoden oder Funktionsaufruf  
28 addition = addieren(zahl1,zahl2)  
29 subtraktion = subtrahieren(zahl1,zahl2)  
30 multiplikation = multiplizieren(zahl1,zahl2)  
31 division = dividieren(zahl1,zahl2)  
32 potenz = potenzieren(zahl1,zahl2)  
33  
34 #Ausgabe der Daten (AUSGABE)  
35 print("Zahl 1:",zahl1, "Zahl 2:",zahl2)  
36 print("-----")  
37 print("Addition:",addition)  
38 print("Subtraktion:",subtraktion)  
39 print("Multiplikation:",multiplikation)  
40 print("Division:",division)  
41 print("Potenz:",potenz)
```



1.4 Eingabe mittels input - Kreis

(I) Problemstellung

Bei den bisherigen Programmen waren alle Werte fest vom Programmcode vorgegeben. Es bestand keine Möglichkeit für die Programmbeutzer, selber Werte in das Programm einzugeben. Diese Option wird jetzt gezeigt. Schreiben Sie dazu ein Programm, das den Inhalt einer Kreisfläche berechnet. Der Benutzer soll den Radius im Programm auf Aufforderung eingeben. Das Programm errechnet die passende Kreisfläche und gibt das Ergebnis aus.

Formel: Kreisfläche = Radius*Radius*pi

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

Radius: 4

Pi: 3.141....

Kreisfläche: 50.265...

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

radius, pi, kreisflaeche

(3) Welche Eigenschaften haben Eingabedaten und Ausgabedaten? (Variablenliste)

Bedeutung	Datentyp	Variable
Radius	Dezimalzahl (einfach)	radius
Pi	Dezimalzahl (einfach)	pi

(4) So soll die Bildschirmausgabe des Programms aussehen:

```
Bitte geben Sie den Raduis ein: 4
Radius: 4.0
Pi: 3.141592653589793
Kreisfläche: 50.26548245743669
```

(5) Verarbeitung

```
#VERARBEITUNG: Deklaration der Funktion/Methode
def berechne_kreisflaeche(radius, pi):
    kreisflaeche = radius*radius*pi
    return kreisflaeche

#Methodenaufruf
ergebnis = berechne_kreisflaeche(radius, pi)
```

(III) Struktogramm

Deklaration und Eingabe: radius als Dezimalzahl
Deklaration und Zuweisung: pi als Dezimalzahl = 3.14
Deklaration und Zuweisung: flaeche = pi * radius * radius
Ausgabe: "Der Inhalt der Kreisfläche beträgt:", flaeche

(IV) Programmcode (Python-Code)

```
L2_3_Rechenoperationen.py × L2_2_4_1_Input_Kreis.py ×
1 import math
2
3 # EINGABE: Deklaration der Eingabevariablen mit Eingabeaufforde
4 radius = float(input("Bitte geben Sie den Radius ein: "))
5 pi = math.pi
6
7 #VERARBEITUNG: Deklaration der Funktion/Methode
8 def berechne_kreisflaeche(radius, pi):
9     kreisflaeche = radius*radius*pi
10    return kreisflaeche
11
12 #Methodenaufruf
13 ergebnis = berechne_kreisflaeche(radius, pi)
14
15 #gewünschte AUSGABE
16 print("Radius: ", radius)
17 print("Pi: ", pi)
18 print("Kreisfläche: ", ergebnis)
19
```

1.5 Eingabe mittels input() - Rechtecke

(I) Problemstellung

Schreiben Sie ein kleines Programm, das den Umfang und den Flächeninhalt eines Rechtecks berechnet. Der Benutzer gibt dazu die Länge und die Breite des Rechtecks ein. Das Programm berechnet die Ergebnisse und gibt diese aus.

Formeln:

$$\text{Umfang} = 2 * (\text{länge} * \text{breite})$$

$$\text{Flächeninhalt} = \text{länge} * \text{breite}$$

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

```
Bitte geben Sie die Länge ein: 2
Bitte geben Sie die Breite ein: 4
Länge: 2.0
Breite: 4.0
Umfang: 16.0 cm
Flächeninhalt: 8.0 cm^2
```

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

laenge, breite

(3) Welche Eigenschaften haben Eingabedaten und Ausgabedaten? (Variablenliste)

Bedeutung	Typ	Variable
Länge	Dezimalzahl	laenge
Breite	Dezimalzahl	breite
Umfang	Dezimalzahl	umfang
Flächeninhalt	Dezimalzahl	flaecheninhalt

(4) So soll die Bildschirmausgabe des Programms aussehen:

```
Bitte geben Sie die Länge ein: 2
Bitte geben Sie die Breite ein: 4
Länge: 2.0
Breite: 4.0
Umfang: 16.0 cm
Flächeninhalt: 8.0 cm^2
```

(5) Verarbeitung

```
#VERARBEITUNG:
#Deklaration der Funktion/Methode
def berechne_umfang(laenge,breite):
    umfang = 2*(laenge*breite)
    return umfang

def berechne_flaecheninhalt(laenge,breite):
    flaecheninhalt = laenge*breite
    return flaecheninhalt

#Methodenaufruf
umfang = berechne_umfang(laenge,breite)
flaecheninhalt = berechne_flaecheninhalt(laenge,breite)
```

(III) Struktogramm

Deklaration und Eingabe: laenge als Dezimalzahl
Deklaration und Eingabe: breite als Dezimalzahl
Deklaration und Zuweisung: umfang = 2 * (laenge + breite)
Deklaration und Zuweisung: flaeche = laenge * breite
Ausgabe: Der Umfang des Rechtecks beträgt: ", umfang, "cm"
Ausgabe: "Die Fläche des Rechtecks beträgt: ", flaeche, "cm^2"

(IV) Programmcode (Python-Code)

```
L2_2_4_2_input_Rechteck.py x
1
2 # EINGABE:
3 #Deklaration der Eingabevariablen mit Eingabeaufforderung
4 laenge = float(input("Bitte geben Sie die Länge ein: "))
5 breite = float(input("Bitte geben Sie die Breite ein: "))
6
7 #VERARBEITUNG:
8 #Deklaration der Funktion/Methode
9 def berechne_umfang(laenge,breite):
10     umfang = 2*(laenge*breite)
11     return umfang
12
13 def berechne_flaecheninhalt(laenge,breite):
14     flaecheninhalt = laenge*breite
15     return flaecheninhalt
16
17 #Methodenaufruf
18 umfang = berechne_umfang(laenge,breite)
19 flaecheninhalt = berechne_flaecheninhalt(laenge,breite)
20
21 #gewünschte AUSGABE
22 print("Länge: ", laenge)
23 print("Breite: ", breite)
24 print("Umfang: ", umfang, "cm")
25 print("Flächeninhalt: ", flaecheninhalt,"cm^2")
26
27
```

1.6 Eingabe mittels input() - Höhenmeter Berechnung

(I) Problemstellung

Die Siedetemperatur von Wasser beträgt auf Höhe des Meeresspiegels **100°** Celsius. Bei zunehmender Höhe verringert sich die Siedetemperatur um **1°** Celsius pro **300** Höhenmeter.

Entwerfen Sie ein Programm, mit dem man mithilfe der Siedetemperatur des Wassers die aktuelle Höhe bestimmen kann.

Formel: Für die höhenmeter > 0 (NHN= Normalhöhennull) gilt

$$\text{siedetemperatur} = 100 - (1 * \text{höhenmeter} / 300)$$

$$\text{höhenmeter} = (100 - \text{siedetemperatur}) * 300$$

Die Eingabe der Daten soll am Bildschirm erfolgen.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

Höhenmeter: 300
Siedetemperatur: 99

Tab

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

höhenmeter, siedetemperatur

(3) Welche Eigenschaften haben Eingabedaten und Ausgabedaten? (Variablenliste)

Bedeutung	Typ	Variable
Siedetemperatur	Dezimalzahl	siedetemperatur
Höhenmeter	Dezimalzahl	hoehenmeter

(4) So soll das Ergebnis aussehen:

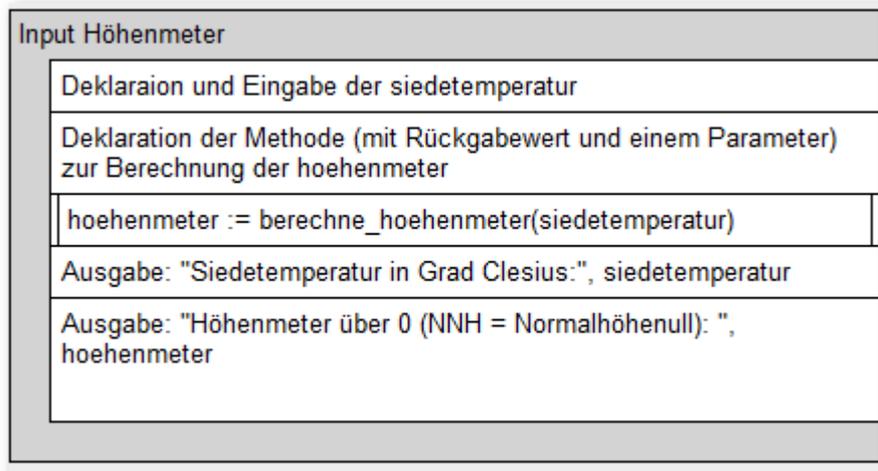
```
Bitte geben Sie die Siedetemperatur ein: 98
Siedetemperatur in Grad Celsius: 98.0
Höhenmeter über 0 (NNH = Normalhöhenull): 600.0
```

(5) Verarbeitung

```
#VERARBEITUNG:
#Deklaration der Funktion/Methode
def berechne_hoehenmeter(siedetemperatur):
    hoehenmeter = (100 - siedetemperatur) * 300
    return hoehenmeter

#Methodenaufruf
hoehemeter = berechne_hoehenmeter(siedetemperatur)
```

Struktogramm:



Programmcode/Quellcode/Lösung:

```
led> x L2_2_4_3_Input_Hoehenmeter.py x

# EINGABE:
#Deklaration der Eingabevariablen mit Eingabeaufforderung
siedetemperatur = float(input("Bitte geben Sie die Siedetemperatur ein: "))

#VERARBEITUNG:
#Deklaration der Funktion/Methode
def berechne_hoehenmeter(siedetemperatur):
    hoehenmeter = (100 - siedetemperatur) * 300
    return hoehenmeter

#Methodenaufruf
hoehemeter = berechne_hoehenmeter(siedetemperatur)

#gewünschte AUSGABE
print("Siedetemperatur in Grad Celsius: ", siedetemperatur)
print("Höhenmeter über 0 (NNH = Normalhöhenull): ", hoehemeter)
```

I

1.7 Funktionen/Methoden – Urlaubsfahrt

Allgemeines zur Möglichkeit Funktionen/Methoden zu entwickeln:

Ihr habt das bereits in der Programmierung des (bisher zumeist) des Programmcode- agert habt. ¹

```
# Funktion mit Rückgabewert, ohne Parameter
def sag_hallo1():
    return 'Hallo '+nachname+' '+vorname
```

In der Zwischenzeit ist es möglich Programmlogik systematisch in eine Funktion/Methode auszulagern.

Für diese Funktionen/Methoden gibt es unterschiedliche Ansätze/Varianten für die Umsetzung. Diese sollt Ihr nun anhand eines kurzen Beispiel kennenlernen.

In der Praxis entwickelt jeder Programmierer seine Präferenzen für zumeist zwei Variante, kennen tut er aber in der Regel alle Varianten:

Parameter sind Variablen (→ Werte) die in der Klammer stehen und explizit **übermittelt** werden, *Rückgabewerte* werden mit dem „**return**“ explizit **ermittelt** (zurückgegeben).

„**print**“ erzeugt eine „prompte“ Ausgabe (sofort), das ist nicht immer gewollt!

Variante 1:

Variante 2:

```
# Funktionen mit Rückgabewert, mit Parametern
def sag_hallo2(vorname,nachname):
    return 'Hallo '+nachname+' '+vorname
```

¹Man nennt dieses Prinzip in der Programmierung auch die **Kapselung**. Sinn und Zweck ist die **Wiederverwendbarkeit** an anderer Stelle. Das hat auch den Vorteil, dass sich der Programmcode an anderer Stelle nicht unnötiger Weise wiederholt → bessere **Wartbarkeit**. Alle diese Grundprinzipien dienen einer besseren Qualität von Software im Ergebnis. Gute Software ist deshalb i.d.R. objektorientiert programmiert!

Variante 3:

```
# Funktionen ohne Rückgabewert, ohne Parametern
def sag_hallo3():
    print('Hallo '+nachname+' '+vorname)
```

Variante 4:

```
# Funktionen ohne Rückgabewert, mit Parametern
def sag_hallo4(vorname,nachname):
    print('Hallo '+nachname+' '+vorname)
```

Kleines Testprogramm: *Tipp → Bitte selbst testen!*

```
L2_5_Funktionen.py × L3_1_Einseitige_Verzweigung_Abball.py ×
1 #EINGABE: Deklaration und Initialisierung über die Eingabeaufforderung
2 vorname = input("Bitte Vorname eingeben: ")
3 nachname = input("Bitte Nachname eingeben: ")
4
5 #VERARBEITUNG:
6 # Funktion mit Rückgabewert, ohne Parameter
7 def sag_hallo1():
8     return 'Hallo '+nachname+' '+vorname
9
10 # Funktionen mit Rückgabewert, mit Parametern
11 def sag_hallo2(vorname,nachname):
12     return 'Hallo '+nachname+' '+vorname
13
14 # Funktionen ohne Rückgabewert, ohne Parametern
15 def sag_hallo3():
16     print('Hallo '+nachname+' '+vorname)
17
18 # Funktionen ohne Rückgabewert, mit Parametern
19 def sag_hallo4(vorname,nachname):
20     print('Hallo '+nachname+' '+vorname)
21
22 # Funktionsaufruf mit Rückgabewert, ohne Parameter
23 begruessung_mR_oP = sag_hallo1()
24
25 # Funktionsaufruf mit Rückgabewert, mit Parametern
26 begruessung_mR_mP = sag_hallo2(vorname,nachname)
27
28 # Funktionsaufruf ohne Rückgabewert, ohne Parametern
29 begruessung_oR_oP = sag_hallo3()
30
31 # Funktionsaufruf ohne Rückgabewert, mit Parametern
32 begruessung_oR_mP = sag_hallo4(vorname,nachname)
33
34 # AUSGABE: Effekte
35 print("Variante 1 (mit Rückgabewert ohne Parameter): ", begruessung_mR_oP)
36 print("Variante 2 (mit Rückgabewert mit Parameter): ",begruessung_mR_mP)
37 print("Variante 3 (ohne Rückgabewert ohne Parameter): ",begruessung_oR_oP)
38 print("Variante 4 (ohne Rückgabewert mit Parameter): ",begruessung_oR_mP)
39
```

Ausgabe des Testprogramm:

```
>>> %Run L2_5_Funktionen.py

Bitte Vorname eingeben: Christine
Bitte Nachname eingeben: Janischek
Hallo Janischek Christine
Hallo Janischek Christine
Variante 1 (mit Rückgabewert ohne Parameter): Hallo Janischek Christine
Variante 2 (mit Rückgabewert mit Parameter): Hallo Janischek Christine
Variante 3 (ohne Rückgabewert ohne Parameter): None
Variante 4 (ohne Rückgabewert mit Parameter): None
```

Bemerkung:

Bei *Variante 3* und *4* erfolgt die Ausgabe mit „**print**“ prompt (sofort) und nicht zum gewünschten Zeitpunkt. Bei Aufruf der Funktion/Methode werden in diesen Varianten die Werte dementsprechend nicht mehr „erinnert“ (vergessen → der Wert ist „None“).

Es gibt Situationen wo genau das eine Rolle spielt! Manchmal mehr, manchmal weniger!

Thema:	Grundlagen der Programmierung in Python – Arbeitsauftrag Quelle: L2 5.1 Funktion Urlaubsfahrt
--------	---

Hinweis: Beachten Sie zur Bearbeitung der nachfolgenden Aufgabenstellungen das Informationsmaterial L2_5_1 Information_Funktion_ohne_Parameter.docx, L2_5_2 Information_Funktion_mit_Parameter.docx und L2_5_3 Information_Funktion_mit_Rückgabewert.docx

(I) Problemstellung

Sie möchten in den Ferien selbst mit dem Auto in den Urlaub fahren. Schreiben Sie ein Programm, das Ihnen die Benzinkosten für Ihre Urlaubsfahrt berechnet.

Da die Berechnung der **gesamten Benzinkosten** etwas länger bzw. komplexer ist als die bisherigen Berechnungen, soll diese Berechnung in eine eigene Funktion ausgelagert werden. Wie das gemacht wird, erfahren Sie in den oben angegebenen Informationsmaterialien. Bearbeiten Sie alle drei angegebenen Materialien (mitsamt der kleinen Herausforderungen am Ende der Informationsblätter) und benutzen Sie anschließend für dieses Programm eine Funktion mit Parameter und Rückgabewert.

Folgende Parameter soll diese Funktion erhalten: **Benzinpreis, Verbrauch Auto, Länge der Strecke**.

Gesamtkosten:

```
gesamtverbrauch = strecke * verbrauch_auf_100km / 100
```

```
kosten = gesamtverbrauch * benzinpreis_je_liter
```

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

gesamtkosten (siehe Bildschirmausgabe)

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

strecke, benzinpreis_je_liter, verbrauch_auf_100km

I

(3) Welche Eigenschaften haben Eingabedaten und Ausgabedaten? (Variablenliste)

Bedeutung	Typ	Variable
Benzinpreis	Dezimalzahl	benzinpreis_je_liter
Verbrauch	Dezimalzahl	verbrauch_auf_100km
Strecke	Dezimalzahl	strecke
Gesamtkosten	Dezimalzahl	gesamtkosten

(4) So soll die Bildschirmausgabe des Programms aussehen:

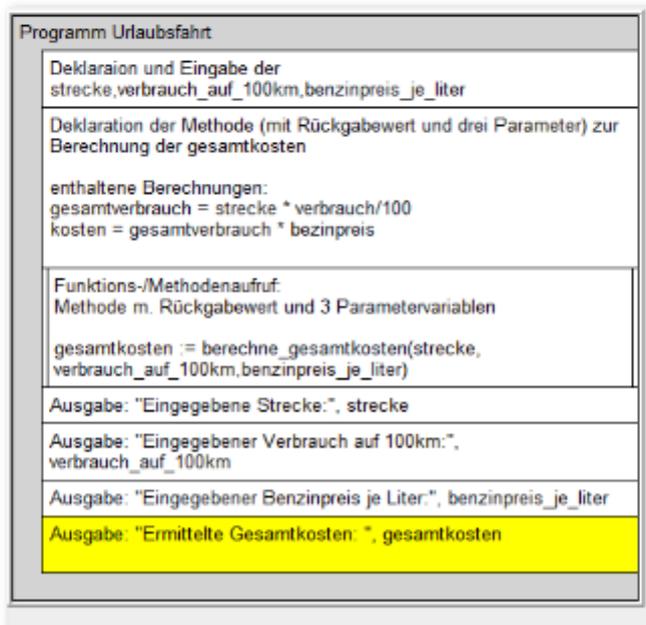
```
>>> %Run L2_2_5_1_Funktion_Urlaubsfahrt.py
Bitte geben Sie den Benzinpreis je Liter ein: 1.1
Bitte geben Sie den durchschnittlichen Verbrauch auf 100km ein: 5.6
Bitte geben Sie die Strecke ein die Sie zurücklegen möchten: 500
Eingegebene Strecke: 500.0
Eingegebener Verbrauch auf 100km: 5.6
Eingegebener Benzinpreis je Liter: 1.1
Ermittelte Gesamtkosten: 30.80 €
```

(5) Verarbeitung

```
#VERARBEITUNG:
#Deklaration der Funktion/Methode
def berechne_gesamtkosten(strecke, verbrauch_auf_100km,benzinpreis_je_liter ):
    gesamtverbrauch = strecke * verbrauch_auf_100km /100
    kosten = gesamtverbrauch * benzinpreis_je_liter
    return kosten

#Methodenaufruf
gesamtkosten = berechne_gesamtkosten(strecke, verbrauch_auf_100km,benzinpreis_je_liter )
```

(III) Struktogramm



(IV) Programmcode (Python-Code)

```
L2_5_Funktionen.py × L2_2_5_1_Funktion_Urlobsfahrt.py × Rabattrechner.py ×
1  import locale
2
3  # EINGABE:
4  #Deklaration der Eingabevariablen mit Eingabeaufforderung
5  benzinpreis_je_liter = float(input("Bitte geben Sie den Benzinpreis je Liter ein: "))
6  verbrauch_auf_100km = float(input("Bitte geben Sie den durchschnittlichen Verbrauch auf 100km ein: "))
7  strecke = float(input("Bitte geben Sie die Strecke ein die Sie zurücklegen möchten: "))
8
9
10 #VERARBEITUNG:
11 #Deklaration der Funktion/Methode
12 def berechne_gesamtkosten(strecke, verbrauch_auf_100km,benzinpreis_je_liter ):
13     gesamtverbrauch = strecke * verbrauch_auf_100km /100
14     kosten = gesamtverbrauch * benzinpreis_je_liter
15
16     #Formatierung des Wertes mit local -> siehe auch import local
17     return locale.format_string("%.2f",kosten,1)+" €"
18
19
20 #Methodenaufruf
21 gesamtkosten = berechne_gesamtkosten(strecke, verbrauch_auf_100km,benzinpreis_je_liter )
22
23 #gewünschte AUSGABE
24 print("Eingegebene Strecke: ", strecke)
25 print("Eingegebener Verbrauch auf 100km: ", verbrauch_auf_100km)
26 print("Eingegebener Benzinpreis je Liter: ", benzinpreis_je_liter)
27 print("Ermittelte Gesamtkosten: ", gesamtkosten)
28
29 |
```

1.8 Einseitige Verzweigung – Abiball

Thema:	Grundlagen der Programmierung in Python – Arbeitsauftrag Quelle: L3 1.1 Einseitige Verzweigung - Abiball
--------	--

Hinweis: Beachten Sie zur Bearbeitung der nachfolgenden Aufgabenstellungen das Informationsmaterial L3_1_1_Information_Verzweigungen_einseitig.docx

(I) Problemstellung

In diesem Projekt soll überprüft werden, welcher Betrag zu zahlen ist, wenn eine Karte für den Abiball **30,00 Euro** kostet und für Bestellungen von **drei** oder mehr Karten ein **Rabatt von 20%** nachgelassen wird.

Berechnungen: Eingabe preis und rabatt

$\text{rabattbetrag} = \text{preis} * (1 - \text{rabatt}/100)$

$\text{kosten_der_bestellung} = \text{menge} * \text{preis} - \text{rabattbetrag}$

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

gesamtkosten_je_bestellung

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

preis_je_karte, menge, rabatt

(3) Welche Eigenschaften haben Eingabedaten und Ausgabedaten? (Variablenliste)

Bedeutung	Typ	Variable
Preis je Karte	Dezimalzahl	preis_je_karte
Menge an Karten	Ganze Zahl (Integer)	menge
Rabatt in %	Dezimalzahl	rabatt
Kosten der Bestellung	Dezimalzahl formatiert	kosten_der_bestellung

(4) Bildschirmausgabe des Programms:

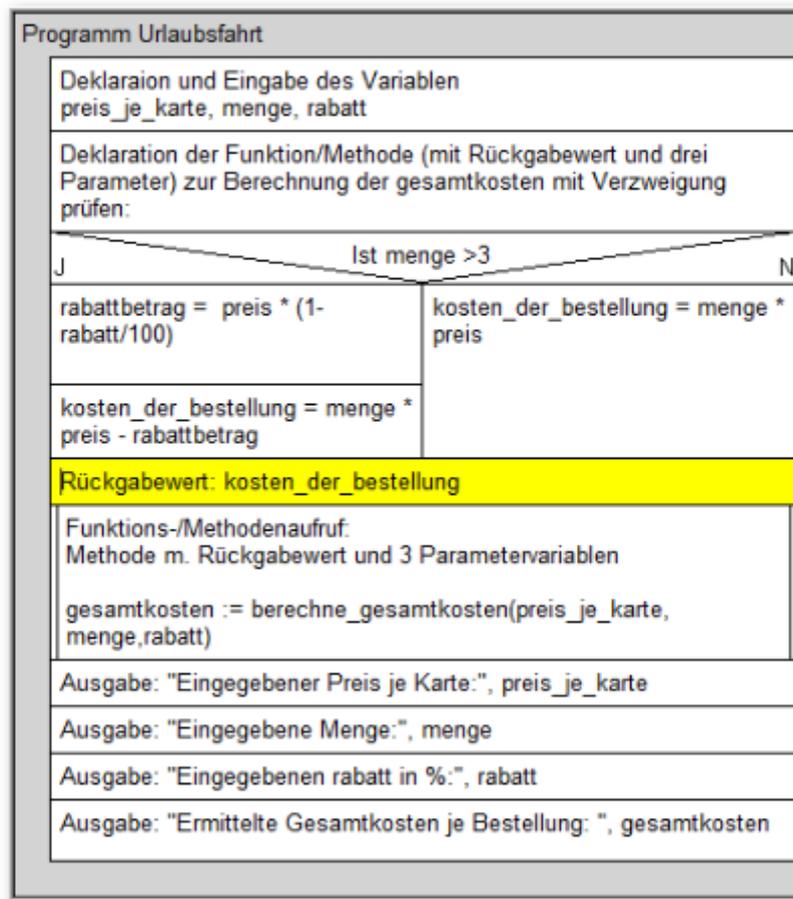
```
>>> %Run L3_1_Einseitige_verzweigung_Abiball.py
Bitte geben Sie den Preis für die Karte ein: 30
Bitte geben Sie die Menge an Karten ein: 5
Bitte geben Sie den Rabatt in % an: 20
Eingegebener Preis je Karte: 30.0 €
Eingegebene Menge: 5
Eingegebenen rabatt in %: 20.0
Ermittelte Gesamtkosten je Bestellung: 126.00 €
```

(5) Verarbeitung

```
#VERARBEITUNG:
#Deklaration der Funktion/Methode
def berechne_gesamtkosten(preis_je_karte, menge, rabatt ):
    if (menge > 3):
        rabattbetrag = preis_je_karte * (1-rabatt/100)
        kosten_der_bestellung = menge * preis_je_karte - rabattbetrag
    else:
        kosten_der_bestellung = menge * preis_je_karte

    #Formatierung des Wertes mit locale -> siehe auch import locale
    return locale.format_string("%.2f",kosten_der_bestellung,1)+" €"
```

(III) Struktogramm



(IV) Programmcode (Python-Code)

```
L2_5_Funktionen.py × L3_1_Einseitige_Verzweigung_Abiball.py ×
1  import locale
2
3  # EINGABE:
4  #Deklaration der Eingabevariablen mit Eingabeaufforderung
5  preis_je_karte = float(input("Bitte geben Sie den Preis für die Karte ein: "))
6  menge = int(input("Bitte geben Sie die Menge an Karten ein: "))
7  rabatt = float(input("Bitte geben Sie den Rabatt in % an: "))
8
9
10 #VERARBEITUNG:
11 #Deklaration der Funktion/Methode
12 def berechne_gesamtkosten(preis_je_karte, menge,rabatt ):
13     if (menge > 3):
14         rabattbetrag = preis_je_karte * (1-rabatt/100)
15         kosten_der_bestellung = menge * preis_je_karte - rabattbetrag
16     else:
17         kosten_der_bestellung = menge * preis_je_karte
18
19     #Formatierung des Wertes mit local -> siehe auch import local
20     return locale.format_string("%.2f",kosten_der_bestellung,1)+" €"
21
22
23 #Methodenaufruf
24 gesamtkosten = berechne_gesamtkosten(preis_je_karte, menge,rabatt )
25
26 #gewünschte AUSGABE
27 print("Eingegebener Preis je Karte: ", preis_je_karte, " €")
28 print("Eingegebene Menge: ", menge)
29 print("Eingegebenen rabatt in %: ", rabatt)
30 print("Ermittelte Gesamtkosten je Bestellung: ", gesamtkosten)
31
32
```

Alternative Lösung mit einer Einseitigen Verzweigung:

```
L3_2_Zweiseitige_Verzweigung_Preisvorschlag.py × L3_1_3 Lösung if geschachtelt Mietzuschuss.py × L3_1_Einseitige_Verzweigung_Abiball_lsg2.py ×
1 import locale
2
3 # EINGABE:
4 #Deklaration der Eingabevariablen mit Eingabeaufforderung
5 preis_je_karte = 30
6 menge = int(input("Bitte geben Sie die Menge an Karten ein: "))
7 rabatt = 20
8
9
10 #VERARBEITUNG:
11 #Deklaration der Funktion/Methode
12 def berechne_gesamtkosten(preis_je_karte, menge,rabatt ):
13     kosten_der_bestellung = menge * preis_je_karte
14     if (menge > 3):
15         rabattbetrag = preis_je_karte * (1-rabatt/100)
16         kosten_der_bestellung = menge * preis_je_karte - rabattbetrag
17
18
19     #Formatierung des Wertes mit local -> siehe auch import local
20     return locale.format_string("%.2f",kosten_der_bestellung,1)+" €"
21
22
23 #Methodenaufruf
24 gesamtkosten = berechne_gesamtkosten(preis_je_karte, menge,rabatt )
25
26 #gewünschte AUSGABE
27 print("Eingegebener Preis je Karte: ", preis_je_karte, " €")
28 print("Eingegebene Menge: ", menge)
29 print("Eingegebenen rabatt in %: ", rabatt)
30 print("Ermittelte Gesamtkosten je Bestellung: ", gesamtkosten)
31
32
```

1.9 Zweiseitige Verzweigung – Preisvorschlag

(I) Problemstellung

In diesem Projekt können Gebote für einen zum Kauf angebotenen Artikel abgegeben werden. In dem Programmcode ist der **Mindestpreis** (hier 24.99) hinterlegt. Ist der von dem Anwender vorgeschlagene **Kaufpreis** geringer, wird der **Text** „Ihr Angebotspreis liegt unterhalb des Mindestpreises!“ ausgegeben. Andernfalls erscheint der Text „Herzlichen Glückwunsch, Sie haben den Artikel erworben!“.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

```
Python 3.7.5 (bundled)
>>> %Run L3_2_Zweiseitige_Verzweigung_Preisvorschlag.py

Bitte geben Ihr Gebot für den Kaufpreis ein: 25
Ihr Gebot lautet: 25.00 €
Mindestpreis: 24.99 €
Meldung: Herzlichen Glückwunsch, Sie haben den Artikel erworben!

>>> %Run L3_2_Zweiseitige_Verzweigung_Preisvorschlag.py

Bitte geben Ihr Gebot für den Kaufpreis ein: 15
Ihr Gebot lautet: 15.00 €
Mindestpreis: 24.99 €
Meldung: Ihr Angebotspreis liegt unterhalb des Mindestpreises!
```

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

Kaufpreis, fixer Wert für den mindestpreis

(3) Welche Eigenschaften haben Eingabedaten und Ausgabedaten? (Variablenliste)

Bedeutung	Typ	Variable
Kaufpreis	Dezimalzahl	kaufpreis
Mindestpreis	Dezimalzahl	mindestpreis

(4) Bildschirmausgabe des Programms (falls Angebotspreis niedriger als 24.99 ist):

```
>>> %Run L3_2_Zweiseitige_Verzweigung_Preisvorschlag.py
Bitte geben Ihr Gebot für den Kaufpreis ein: 15
Ihr Gebot lautet: 15.00 €
Mindestpreis: 24.99 €
Meldung: Ihr Angebotspreis liegt unterhalb des Mindestpreises!
```

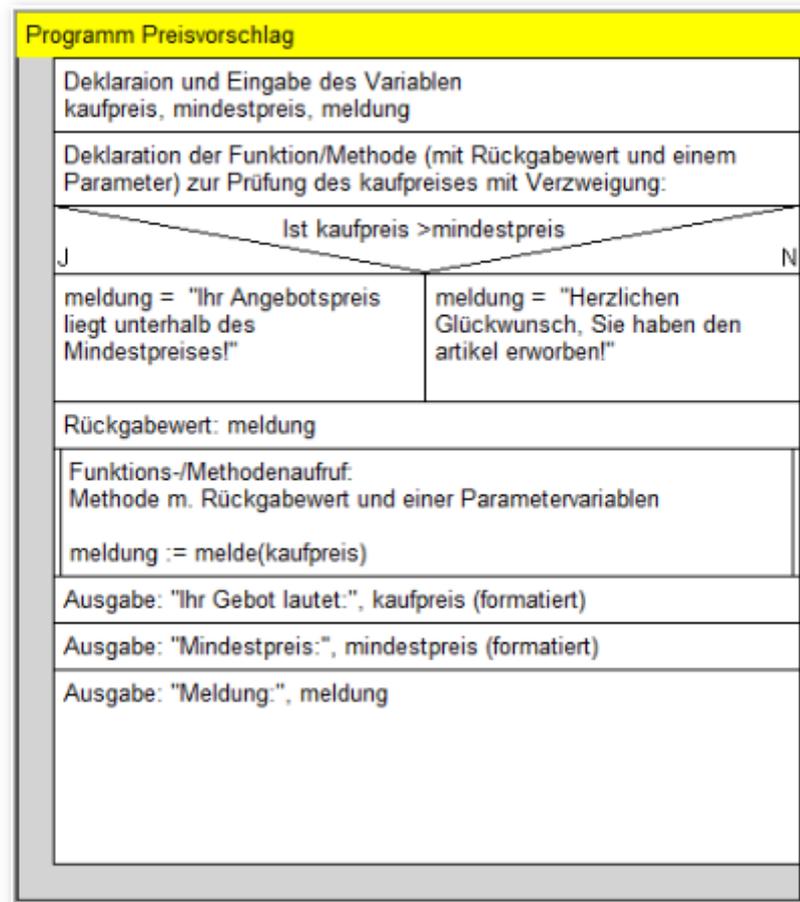
(5) Verarbeitung

```
#VERARBEITUNG:
#Deklaration der Funktion/Methode
def melde(kaufpreis):

    if (kaufpreis < mindestpreis):
        meldung = "Ihr Angebotspreis liegt unterhalb des Mindestpreises!"
    else:
        meldung = "Herzlichen Glückwunsch, Sie haben den Artikel erworben!"
    return meldung

#Methodenaufruf
meldung = melde(kaufpreis)
```

(III) Struktogramm



(IV) Programmcode (Python-Code)

```
L3_2_Zweiseitige_Verzweigung_Preisvorschlag.py × L3_1_3 Lösung if geschachtelt Mietzuschuss.py ×
1  import locale
2
3  # EINGABE:
4  #Deklaration der Eingabevariablen mit Eingabeaufforderung, Fixer Wert und Stri
5  kaufpreis = float(input("Bitte geben Ihr Gebot für den Kaufpreis ein: "))
6  mindestpreis = 24.99
7  meldung = ""
8
9
10 #VERARBEITUNG:
11 #Deklaration der Funktion/Methode
12 def melde(kaufpreis):
13
14     if (kaufpreis < mindestpreis):
15         meldung = "Ihr Angebotspreis liegt unterhalb des Mindestpreises!"
16     else:
17         meldung = "Herzlichen Glückwunsch, Sie haben den Artikel erworben!"
18     return meldung
19
20
21 #Methodenaufruf
22 meldung = melde(kaufpreis)
23
24 #gewünschte AUSGABE
25 #Formatierung der Beträge mit local -> siehe auch import local
26 print("Ihr Gebot lautet: ", locale.format_string("%.2f",kaufpreis,1), " €")
27 print("Mindestpreis: ", locale.format_string("%.2f",mindestpreis,1), " €")
28 print("Meldung: ", meldung)
29
```

1.10 Geschachtelte Verzweigung – Mietzuschuss

(I) Problemstellung

Schreiben Sie ein Programm, das den Mietzuschuss in Abhängigkeit vom Mietpreis berechnet.

Bei einer Miete von **weniger als 500,00 Euro** beträgt der **Zuschuss 2%**. Von 500,00 Euro bis **unter 1000,00 Euro** beträgt der **Zuschuss 5%** und **ab 1000,00 Euro** erhält man einen **Zuschuss von 7%**.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

Zuschussrate, mietpreis, zuschussbetrag

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

mietpreis

(3) Welche Eigenschaften haben Eingabedaten und Ausgabedaten? (Variablenliste)

Bedeutung	Typ	Variable
Mietpreis	Dezimalzahl	mietpreis
Zuschussbetrag	Dezimalzahl	zuschussbetrag
Zuschussrate	Ganze Zahl	zuschussrate

(4) Bildschirmausgabe des Programms:

Vollständiger Test:

```
>>> %Run L3_1_2_Zweiseitige_Verzweigung_Preisvorschlag.py
Bitte geben Sie den Mietpreis an: 499
Ihr Zuschussrate: 2 %
Ihr Mietpreis: 499.00 €
Zuschussbetrag: 9.98 €

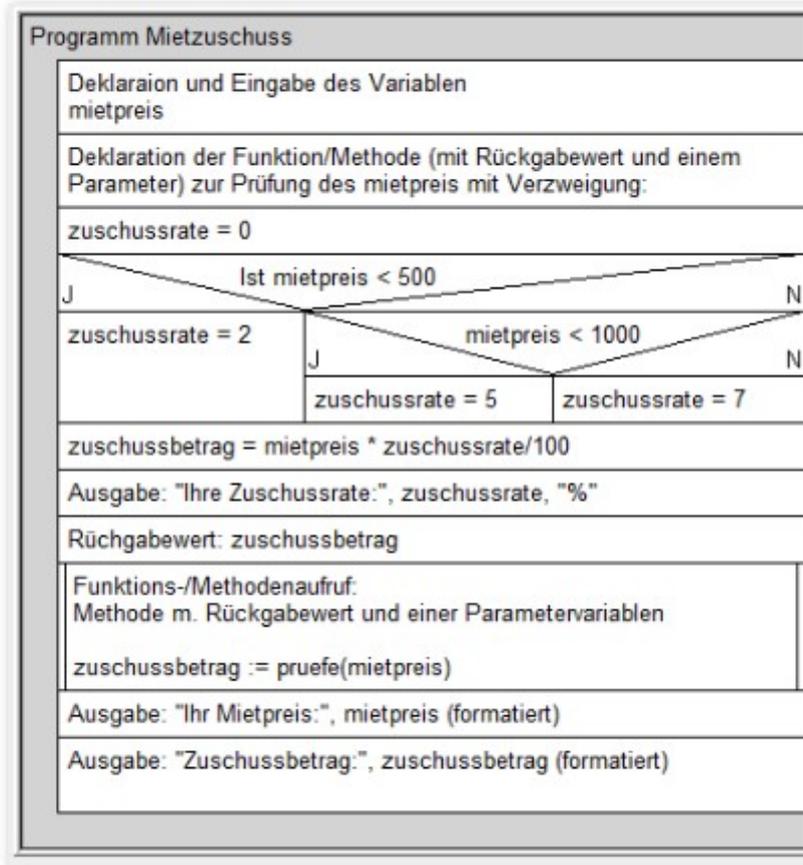
>>> %Run L3_1_2_Zweiseitige_Verzweigung_Preisvorschlag.py
Bitte geben Sie den Mietpreis an: 500
Ihr Zuschussrate: 5 %
Ihr Mietpreis: 500.00 €
Zuschussbetrag: 25.00 €

>>> %Run L3_1_2_Zweiseitige_Verzweigung_Preisvorschlag.py
Bitte geben Sie den Mietpreis an: 1001
Ihr Zuschussrate: 7 %
Ihr Mietpreis: 1001.00 €
Zuschussbetrag: 70.07 €
```

(5) Verarbeitung

```
3 #VERARBEITUNG:
9 #Deklaration der Funktion/Methode
3 def pruefe(mietpreis):
1     zuschussrate = 0
2
3     if (mietpreis < 500):
4         zuschussrate = 2
5     else:
6         if(mietpreis <1000):
7             zuschussrate = 5
8         else:
9             zuschussrate = 7
10    zuschussbetrag = mietpreis * zuschussrate/100
11    print("Ihr Zuschussrate: ", zuschussrate, " %")
12    return zuschussbetrag
13
14
15 #Methodenaufruf
16 zuschussbetrag = pruefe(mietpreis)
```

(III) Struktogramm



(IV) Programmcode (Python-Code)

```
L3_1_Einseitige_Verzweigung_Abilball.py x L3_1_3_Geschachtelte_Verzweigung_Mietzuschuss.py x
1  import locale
2
3  # EINGABE:
4  #Deklaration der Eingabevariablen mit Eingabeaufforderung, Fixer Wert und String ohne Inhalt
5  mietpreis = float(input("Bitte geben Sie den Mietpreis an: "))
6
7
8  #VERARBEITUNG:
9  #Deklaration der Funktion/Methode
10 def pruefe(mietpreis):
11     zuschussrate = 0
12
13     if (mietpreis < 500):
14         zuschussrate = 2
15     else:
16         if(mietpreis <1000):
17             zuschussrate = 5
18         else:
19             zuschussrate = 7
20     zuschussbetrag = mietpreis * zuschussrate/100
21     print("Ihr Zuschussrate: ", zuschussrate, " %")
22     return zuschussbetrag
23
24
25 #Methodenaufruf
26 zuschussbetrag = pruefe(mietpreis)
27
28 #gewünschte AUSGABE
29 #Formatierung der Beträge mit local -> siehe auch import local
30 print("Ihr Mietpreis: ", locale.format_string("%.2f",mietpreis,2), " €")
31 print("Zuschussbetrag: ", locale.format_string("%.2f",zuschussbetrag,2), " €")
32
```

1.11 Verzweigung mit logischen Operatoren Eintritt Tierpark

(I) Problemstellung

In diesem Projekt soll überprüft werden, ob ein Besucher eines Tierparks ein **Kind** (bis **12 Jahre**) oder ein **Senior** (ab **60 Jahre**) ist. Kinder und Senioren zahlen **8,00 Euro** Eintritt, für alle anderen beträgt der Eintrittspreis **12,00 Euro**.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

alter, eintrittspreis

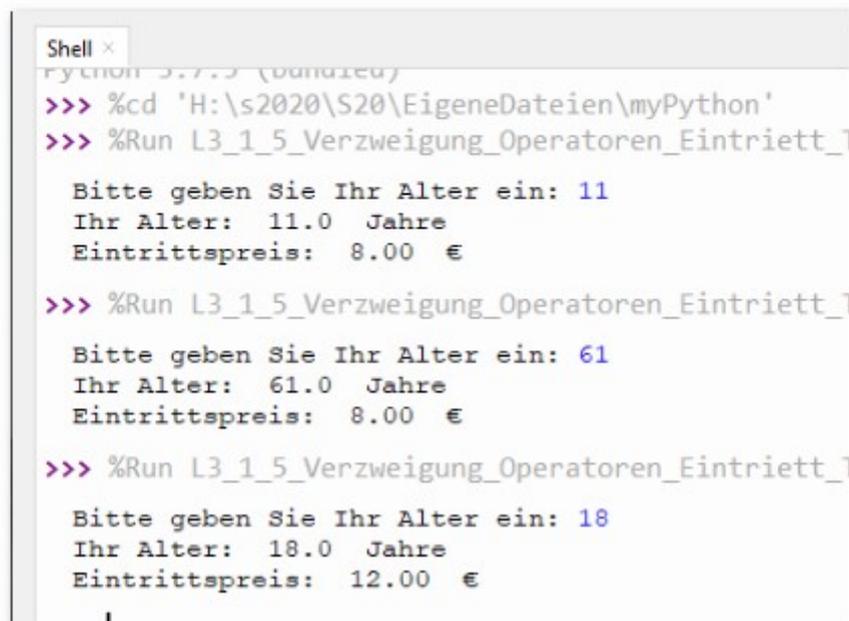
(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

alter

(3) Welche Eigenschaften haben Eingabedaten und Ausgabedaten? (Variablenliste)

Bedeutung	Typ	Variable
Alter	Ganze Zahl	alter
Eintrittspreis	Dezimalzahl	eintrittspreis

(4) Bildschirmausgabe des Programms:



```
Shell x
Python 3.7.5 (tags/bundled)
>>> %cd 'H:\s2020\S20\EigeneDateien\myPython'
>>> %Run L3_1_5_Verzweigung_Operatoren_Eintriett_1.py

Bitte geben Sie Ihr Alter ein: 11
Ihr Alter: 11.0 Jahre
Eintrittspreis: 8.00 €

>>> %Run L3_1_5_Verzweigung_Operatoren_Eintriett_1.py

Bitte geben Sie Ihr Alter ein: 61
Ihr Alter: 61.0 Jahre
Eintrittspreis: 8.00 €

>>> %Run L3_1_5_Verzweigung_Operatoren_Eintriett_1.py

Bitte geben Sie Ihr Alter ein: 18
Ihr Alter: 18.0 Jahre
Eintrittspreis: 12.00 €
```

(5) Verarbeitung

```
def pruefeKategorie(alter):
    if(alter < 12 or alter >= 60):
        eintrittspreis = 8.00
    else:
        eintrittspreis = 12.00
    return eintrittspreis

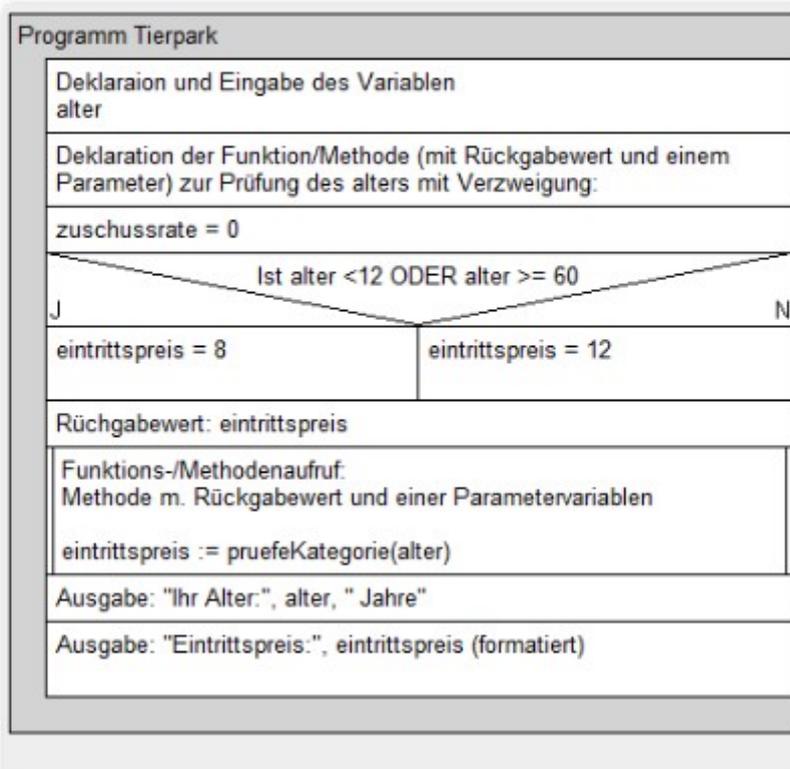
eintrittspreis = pruefeKategorie(alter)
```

Operatoren in Python: <https://www.python-kurs.eu/operatoren.php>

Operator	Bezeichnung	Beispiel
+, -	Addition, Subtraktion	10 - 3
*, /, %	Multiplikation, Division, Rest	27 % 7 Ergebnis: 6
+x, -x	Vorzeichen	-3
~x	Bitweises Not	~3 - 4 Ergebnis: -8
**	Exponentiation	10 ** 3 Ergebnis: 1000
or, and, not	Boolesches Oder, Boolesches Und, Boolesches Nicht	(a or b) and c
in	"Element von"	1 in [3, 2, 1]
<, <=, >, >=, !=, ==	Die üblichen Vergleichsoperatoren	2 <= 3
~, &, ^	Bitweises Oder, Bitweises Und, Bitweises XOR	6 ^ 3
<<, >>	Shiftoperatoren	6 << 3

(III) Struktogramm

I



(IV) Programmcode (Python-Code)

```
L3_1_5_Verzweigung_Operatoren_Eintritt_Tierpark.py ×
1  import locale
2
3  # EINGABE:
4  #Deklaration der Eingabevariablen mit Eingabeaufforderung, Fixer Wert und String ohne
5  alter = float(input("Bitte geben Sie Ihr Alter ein: "))
6
7
8  #VERARBEITUNG:
9  #Deklaration der Funktion/Methode
10 def pruefeKategorie(alter):
11     if(alter < 12 or alter >= 60):
12         eintrittspreis = 8.00
13     else:
14         eintrittspreis = 12.00
15     return eintrittspreis
16
17 #Methodenaufruf
18 eintrittspreis = pruefeKategorie(alter)
19
20
21
22 #gewünschte AUSGABE
23 #Formatierung der Beträge mit local -> siehe auch import local
24 print("Ihr Alter: ", alter, " Jahre")
25 print("Eintrittspreis: ", locale.format_string("%.2f", eintrittspreis, 2), " €")
26
```

1.12 Verzweigung mit logischen Operatoren Autovermietung

(I) Problemstellung

Die Firma Rent A Car in Stuttgart vermietet Oldtimer für 50,00 € pro Tag. Alle Kunden, die nicht in Stuttgart wohnen, erhalten einen Sonderrabatt von 10%.

Das Projekt soll nach Eingabe des Wohnorts und der Mietdauer die Mietkosten ermitteln und anzeigen.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

wohnort, mietdauer, mietkosten, rabatt, preis

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

wohnort, mietdauer(tage)

(3) Welche Eigenschaften haben die Eingabedaten und die Ausgabedaten? (Variablenliste)

Bedeutung	Typ	Variable
Wohnort	Zeichenkette	wohnort
Mietdauer	Ganze Zahl	mietdauer
Mietkosten	Dezimalzahl	mietkosten
Rabatt	Dezimalzahl	rabatt
Preis	Dezimalzahl	preis

(4) Bildschirmausgabe des Programms:

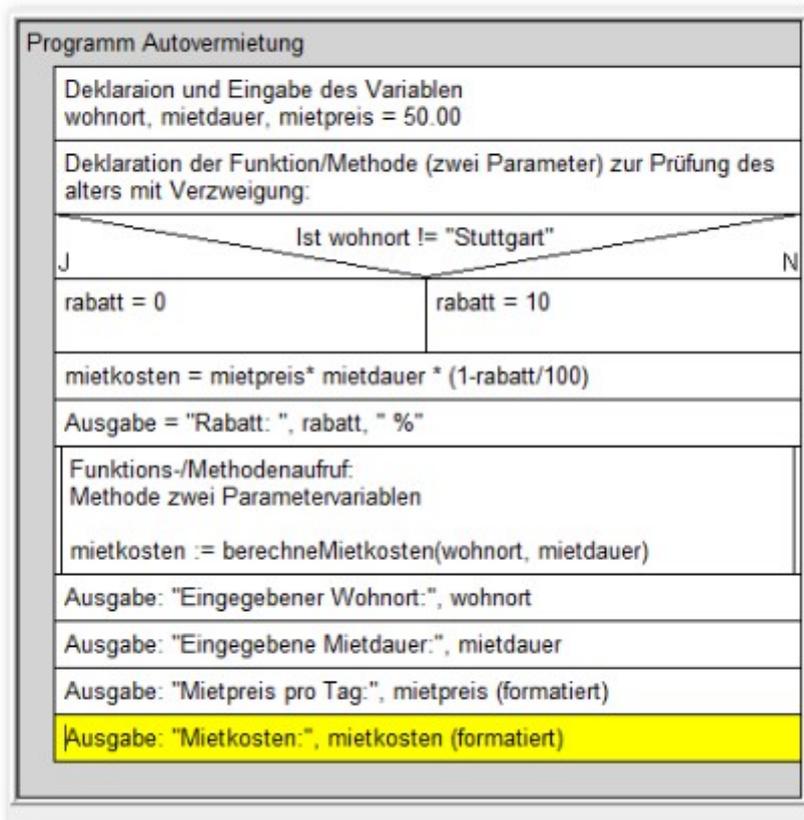
```
Eingegeneber Wohnort: Stuttgart
Eingegebene Mietdauer: 14 Tage
Preis pro Tag: 50,00 €
Rabatt: 10 %
Mietkosten: 630 €
```

(5) Verarbeitung

```
def berechneMietkosten(wohnort, mietdauer):
    if(wohnort != `Stuttgart`):
        rabatt = 0
    else:
        rabatt = 10
    mietkosten = mietpreis * mietdauer * (1- rabatt/100)
    return mietkosten

mietkosten = berechneMietkosten(wohnort, mietdauer)
```

(III) Struktogramm



(IV) Programmcode (Python-Code)

```
L3_1_6_Verzweigung_Operatoren_Autovermietung.py x
1  import locale
2
3  # EINGABE:
4  #Deklaration der Eingabevariablen mit Eingabeaufforderung, Fixer Wert und String ohne Inhalt
5  wohnort = input("Bitte geben Sie Ihren Wohnort ein: ")
6  mietdauer = float(input("Bitte geben Sie Ihre gewünschte Mietdauer in Tagen ein: "))
7  mietpreis = 50.00
8
9
10 #VERARBEITUNG:
11 #Deklaration der Funktion/Methode
12 def berechneMietkosten(wohnort, mietdauer):
13     if(wohnort != "Stuttgart"):
14         rabatt = 0
15     else:
16         rabatt = 10
17     mietkosten = mietpreis * mietdauer * (1- rabatt/100)
18     print("Rabatt: ", rabatt, " %")
19     return mietkosten
20
21 #Methodenaufruf
22 mietkosten = berechneMietkosten(wohnort, mietdauer)
23
24 #gewünschte AUSGABE
25 #Formatierung der Beträge mit local -> siehe auch import local
26 print("Eingegebener Wohnort: ", wohnort)
27 print("Eingegebene Mietdauer: ", 14, " Tage")
28 print("Mietpreis pro Tag: ", locale.format_string("%.2f",mietpreis,2), " €")
29
30 print("Mietkosten: ", locale.format_string("%.2f",mietkosten,2), " €")
31
32
```

1.13 Wiederholungen mit der For-Schleife Taschengeld

(I) Problemstellung

Schreiben Sie ein Programm, das den Betrag des Taschengelds in Abhängigkeit vom Alter ausgibt. Das Taschengeld wird vom 6. bis zum 21. Lebensjahr ausgezahlt. Dabei sind der Anfangsbetrag und die jährliche Erhöhung vom Benutzer einzugeben.

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

Betrag, alter

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

anfangsbetrag, erhoehung

(3) Welche Eigenschaften haben die Eingabedaten und die Ausgabedaten? (Variablenliste)

Bedeutung	Typ	Variable
Betrag,Anfangsbetrag,Erhoehung	Dezimalzahl	betrag, anfangsbetrag, erhoehung
Alter,Lebensjahr	Ganze Zahl	alter, lebensjahr

(4) Bildschirmausgabe des Programms:

Tabellenzeile

```
Anfangsbetrag: 384 €
Erhöhung: 2%
```

```
-----
```

```
Alter      | Betrag   |
6. Lebensjahr    384 €
7. Lebensjahr    391,68 €
8. Lebensjahr    384 €
9. Lebensjahr    384 €
...

```

(5) Verarbeitung

```
#VERARBEITUNG:
#Deklaration der Funktion/Methode
def berechneBetragJeLebensJahr(betrag,erhoehung):
    #Ausgabe einmalig
    print("Anfangsbetrag:", betrag)
    print("Erhöhung:", erhoehung)
    print("-----")
    print("Alter          | Betrag          |")
    for i in range(6,21):
        betrag = betrag * (1+ erhoehung/100)
        #Ausgabe zeilenweise
        print(i , "Lebensjahr:  ", locale.format_string("%.2f",betrag,2), " €")

#Methodenaufruf
berechneBetragJeLebensJahr(betrag,erhoehung)
```

Python Funktion → xrange

```
xrange(begin,end)
```

Python → For Schleife

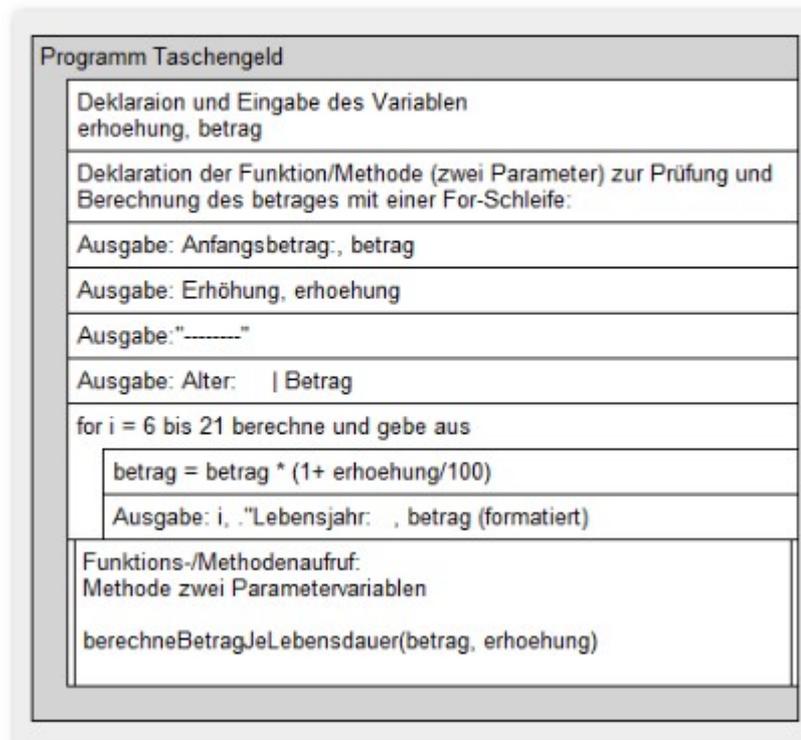
```
for i in range(6,21)
    print(i)
```

Python → While Schleife

```
i = 6
while i < 21:
    print(i)
    i += 1
```

I

(III) Struktogramm



(IV) Programmcode (Python-Code)

```
L3_2_1_Wiederholungen_For_Schleife_Taschengeld.py x
1  import locale
2
3  # EINGABE:
4  #Deklaration der Eingabevariablen mit Eingabeaufforderung, Fixer Wert und
5  erhoehung = float(input("Bitte geben Sie die Erhöhung ein: "))
6  betrag = float(input("Bitte geben Sie den Anfangsbetrag: "))
7
8
9  #VERARBEITUNG:
10 #Deklaration der Funktion/Methode
11 def berechneBetragJeLebensJahr(betrag,erhoehung):
12     #Ausgabe einmalig
13     print("Anfangsbetrag:", betrag)
14     print("Erhöhung:", erhoehung)
15     print("-----")
16     print("Alter      | Betrag      |")
17     for i in range(6,21):
18         betrag = betrag * (1+ erhoehung/100)
19         #Ausgabe zeilenweise
20         print(i , "Lebensjahr:  ", locale.format_string("%.2f",betrag,2))
21
22 #Methodenaufruf
23 berechneBetragJeLebensJahr(betrag,erhoehung)
24
```

1.14 Wiederholungen mit der While-Schleife Fischteich

(II) Problemanalyse

(1) Welche Ausgabedaten will man erhalten?

jahr, fischbestand, endbestand, faktor

(2) Welche Eingabedaten werden zur Bearbeitung benötigt?

endbestand, faktor

(3) Welche Eigenschaften haben die Eingabedaten und die Ausgabedaten? (Variablenliste)

Bedeutung	Typ	Variable
Fischbestand, jahr	Ganze Zahl	
faktor	Dezimalzahl	

(4) Bildschirmausgabe des Programms:

Eingegebener, zu erreichender Fischbestand: 384 €

Faktor: 2

Jahr | Bestand |

1. Jahr 3

2. Jahr 6

3. Jahr 12

4. Jahr 24

...

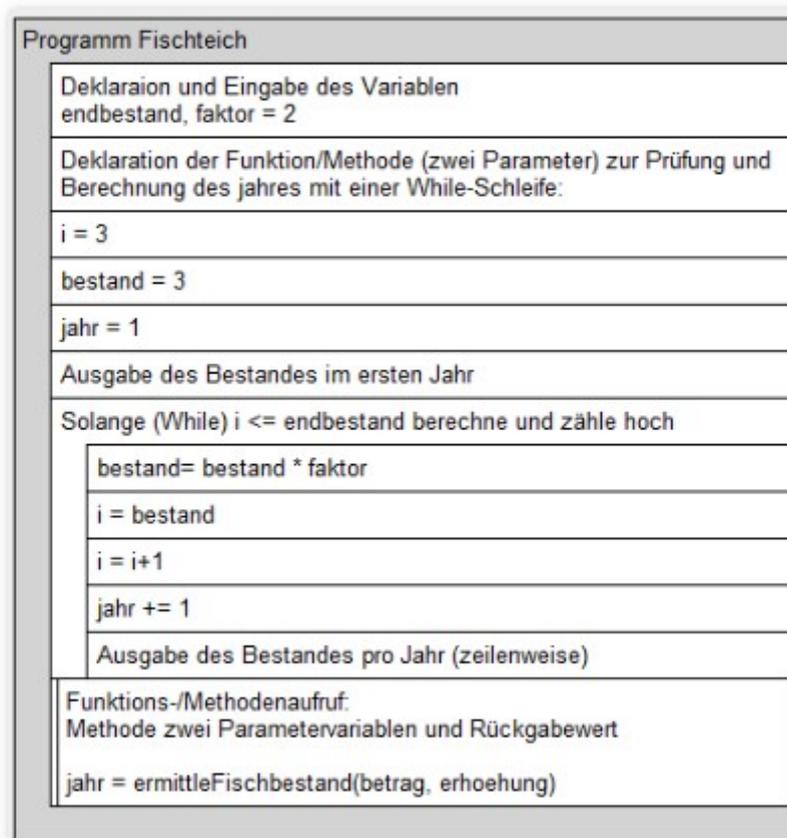
(5) Verarbeitung

```
faktor = 2

def ermittleFischbestand(endbestand, faktor):
    i = 3
    jahr = 1
    while i <= endbestand:
        i = i * faktor
        i = i+1
        jahr += 1
    return jahr

#Methodenaufruf
jahr = ermittleFischbestand(endbestand, faktor)
```

(III) Struktogramm



(IV) Programmcode (Python-Code)

```
L3_2_2_Wiederholungen_While_Schleife_Fischteich.py ×
5  endbestand = int(input("Bitte geben Sie den Endbestand ein: "))
6  faktor = 2
7
8
9  #VERARBEITUNG:
10 #Deklaration der Funktion/Methode
11 def ermittleFischbestand(endbestand, faktor):
12     i = 3
13     bestand = 3
14     jahr = 1
15     print("Bestand im ", jahr, ". Jahr: ",bestand)
16     while i <= endbestand:
17         bestand = bestand * faktor
18         i = bestand
19         i = i+1
20         jahr += 1
21         print("Bestand im ", jahr, ". Jahr: ",bestand)
22     return jahr
23
24 #Methodenaufruf
25 jahr = ermittleFischbestand(endbestand, faktor)
26
27 #Ausgabe
28 print("Jahr in dem der Endbestand erreicht ist:", jahr, " Jahr")
29
```