1. Fortgeschrittene Programmierkonzepte Teil 1

1.1. Variable Anzahl an Parametern (*args und *kwargs)

- Funktionsdefinition für das Aufsummieren von zwei Zahlen:
    - Anzahl der Parameter muss bei der Definition und dem Aufruf übereinstimmen

```
def teilen(a, b):
    q = a / b
    return q

z1 = 5
z2 = 3
s = teilen(z1,z2)
print(f"Die Quotient der Zahlen {z1} und {z2} ist {s}")
```

    - benannte Parameter
        - Reihenfolge der Parameter muss beim Aufruf nicht eingehalten werden

```
def volumen(breite, laenge, tiefe):
    vol = breite * laenge * tiefe
    return vol

v = volumen(tiefe=6,laenge=2,breite=4)
print(v)
```

    - optionale Parameter
        - diese Parameter müssen einen Vorgabewert besitzen und am Ende der Parameterliste stehen

```
def volumen(breite, laenge, tiefe=2):
    vol = breite * laenge * tiefe
    return vol

v = volumen(laenge=2,breite=4)
print(v)
```

- Variable Anzahl von Parametern
  - diese Parameter werden mit einem Stern markiert und stehen am Ende der Parameterliste
  - dieser Parameter enthält ein Tupel
  - dieses Tupel kann mit einer Schleife abgearbeitet werden

```
def mittelwert(parm1, *parms):
    mw = parm1
    for p in parms:
        mw += p
    return mw / (1.0 + len(parms))

z1 = 5
z2 = 3
m = mittelwert(z1,z2,6)
print(f"Der Mittelwert ist {m}")
```

- Dictionary zur Parameterübergabe
  - diese Parameter werden mit einem Doppelstern markiert und stehen am Ende der Parameterliste
  - dieser Parameter enthält ein Assoziatives Array (Dictionary)
  - dieses Dictionary kann mit einer Schleife abgearbeitet werden

```
def ausgeben_kwargs(**kwargs):
    for k, v in kwargs.items():
        print(f"{k} = {v}")


ausgeben_kwargs(Helligkeit=5,
                Bildschirm="Hochformat",
                Bildschirmauflösung="Full HD")
```

1.2. Anonyme Funktionen mit dem lambda-Operator

- Anonyme Funktionen sind Funktionen ohne Namen
- wie Funktionen mit Namen führen anonyme Funktionen Berechnungen oder Ähnliches durchzuführen
- eine besondere Form der anonymen Funktion ist ein Lambda-Ausdruck
- der Lambda-Ausdruck berechnet über die Parameter ein Resultat
- Aufbau eines Lambda-Ausdrucks:
    - **lambda** parameter : ausdruck

    - Definition einer Lambda-Funktion mit einem Parameter:
      Verdopplung eines Wertes
    - die Berechnung erfolgt unmittelbar an der Stelle
    - im print-Befehl ist der Lambda-Ausdruck in Klammer zu schreiben – die Parameterübergabe folgt unmittelbar dahinter, ebenfalls in Klammern

```
print( (lambda x: x * 2) (5) )
```

    - Definition einer Lambda-Funktion mit einem Parameter:
      Verdopplung eines Wertes
    - dem Lambda-Ausdruck wird ein Name gegeben
    - im print-Befehl ist der Name des Lambda-Ausdruck anzugeben, in Klammer folgt der Parameter

```
f = lambda x: x * 2
print(f(3))
```

    - Definition einer Lambda-Funktion mit mehreren Parametern:
      „irgendeine beliebige Berechnung“
    - dem Lambda-Ausdruck wird ein Name gegeben
    - im print-Befehl ist der Name des Lambda-Ausdruck anzugeben, in Klammer folgen die Parameter

```
f = lambda x, y, z: (x - y) * z
print(f(1,2,3))
```

Aufgaben

1. Literzahl umwandeln
   Es soll ein Programm implementiert werden, dass für beliebige Volumenangaben als Gleitpunktzahl in Liter den entsprechenden Wert im ml, cl oder l ausgibt:

Eingabe	Ausgabe
1.0 und größer	l
0.1 und größer	cl
0.001 und größer	ml
kleiner als 0.001	Wert zu klein

   Beispiel

Eingabe	Ausgabe:
1.0	1.0 l
0.42	42.0 cl
0.023	23.0 ml
0.0001	Wert zu klein

2. LKW-Maut
   Ein Programm soll für erfasste LKW die entsprechende LKW-Mautgebühr nach der folgenden Tabelle berechnen.:

Schadstoffklasse	A [ct/km]	B [ct/km]	C [ct/km]	D [ct/km]	E [ct/km]	F [ct/km]
Bis 3 Achsen	12,50	14,60	15,70	18,80	19,80	20,80
Ab 4 Achsen	13,10	15,20	16,30	19,40	20,40	21,40

   Beispiel

1. Schadstoffklasse A, 2 Achsen, 13 km	162,5 Eurocent
2. Schadstoffklasse D, 5 Achsen, 13 km	252,2 Eurocent