5. Fortgeschrittene Programmierkonzepte Teil 3

5.1. zip()-Funktion

- die Funktion **zip()** verbindet die Elemente von iterierbaren Objekten (Liste oder Dictionary) miteinander zu einem neuen iterierbaren Objekt:

```
# Definition einer Liste mit Hauptstädten
hauptstadt = ["Paris", "Rom", "Berlin"]
# Definition einer Liste mit Ländern
land = ["Frankreich", "Italien", "Deutschland"]

# die zip()-Funktion verbindet die Elemente beider Listen zu einer neuen
Liste
# Elemente werden in der gegebenen Reihenfolge verbunden
land_hauptstadt = zip(land, hauptstadt)

# Elemente der neuen ausgeben
for element in land_hauptstadt:
    print (x)
```

5.2. Generatoren

- Folgen von Werten lassen sich damit komfortabel erzeugen, z.B. 0, 2, 4, 6, …
- diese Werte liegen jedoch nicht als Liste im Speicher vor
- die Werte werden nacheinander mithilfe einer Generatorfunktion berechnet
- in der Generatorfunktion wird der Wert mit **yield** zurückgegeben (statt mit return)
- die Generatorwerte lassen sich mit einer for-Schleife abrufen oder
- mit **next()** wird der nächste Generatorwert bereitgestellt
- normale Funktionen geben mit der return-Anweisung die Kontrolle an die rufende Stelle zurück – beim erneuten Aufruf wird die Funktion wieder von Beginn an ausgeführt – wieder bis zur return-Anweisung
- Generatorfunktionen speichern beim Erreichen der yield-Anweisung die aktuelle Position – beim nächsten Aufruf der Generatorfunktion wird die Ausführung an dieser Stelle fortgesetzt -bis zur

```
# Definition der Generatorfunktion zum Erzeugen der ersten n
# Quadratzahlen
def quadrieren(n):
    i = 1
    while i <= n:
        q = i * i
        yield q
        i += 1

# Abrufen der n Quadratzahlen
for x in quadrieren(10):
    print(x, end=" ")
```

Aufgaben

1. Palindrom
   es eine Funktion palindrom(s) zu erstellen, die eine Zeichenkette (string) als Argument erhält und als Rückgabewert genau dann True liefert, wenn es ein Palindrom ist
   Palindrome sind Texte, die von vorne wie von hinten gelesen gleich sind, z.B. OTTO
   Für die zu prüfenden Zeichenketten ist eine Generatorfunktion zu definieren.

2. Perfekte Zahl
   ein Programm soll eine Zahl n prüfen, ob n eine perfekte Zahl ist.
   Perfekte Zahlen sind solche, die genauso groß sind, wie die Summe ihrer Teiler (außer sich selbst). So ist z. B. 6 eine perfekte Zahl, weil 1 + 2 + 3 = 6
   Die zu prüfenden Zahlen n sind mit einer Generatorfunktion zu erzeugen.