17 Datentyp Liste

17.1 Beschreibung

- eine Liste ist eine Sequenz von Elementen
- die Elemente der Liste sind änderbar
- eine Liste kann unterschiedliche Datentypen enthalten
- eine Liste kann wiederum eine Liste enthalten
- eine Liste wird durch die Zuweisung von Elementen innerhalb eckiger Klammern angelegt
- mit Listen-Operatoren und Listen-Funktionen lassen sich Listen ändern und untersuchen
- es lassen sich ein- und mehrdimensionale Felder (Arrays) abbilden
- gleiche Werte können mehrfach vorkommen

- Beispiel:

```
# Liste anlegen
zahlen = [1, 5, 7, 9, 22, 35, 1]
# Liste ausgeben
print(zahlen)
```

```
[1, 5, 7, 9, 22, 35, 1]
```

```
# 1. Element der Liste ausgeben (index fängt bei 0 an)
print(zahlen[0])
```

```
1
```

```
# 2. bis 4. Element der Liste ausgeben (index = 1:4)
print(zahlen[1:4])
```

```
[5, 7, 9]
```

17.2 Listen-Operatoren

- Verketten von Listen mit **+**
- Vervielfachen von Listen mit *****
- Zugriff auf ein Element per Index (Zählweise beginnt bei 0) mit **[i]**
- Zugriff auf einen Bereich von Elementen per Index (Zählweise beginnt bei 0) mit **[a:b]**
- Prüfen, ob ein Element in der Liste enthalten ist mit **in**
- Prüfen, ob ein Element in der Liste NICHT enthalten ist mit **not in**
- Element an einem angegebenen Index löschen mit **del liste[i]**
- Bereich von Elementen per Index löschen mit **del liste[a:b]**

- Beispiele:

```
# Liste verketten
list_t1 = ["Das", "ist"]
list_t2 = ["eine", "Addition"]
list_gesamt = list_t1 + list_t2
print(list_gesamt))
```

```
['Das', 'ist', 'eine', 'Addition']
```

```
# Element in einer Liste verändern
list_2 = ["das", "ist", "eine", "Liste"]
print(list_2)
```

```
['das', 'ist', 'eine', 'Liste']
```

```
list_2[0] = "Das"
print(list_2)
```

```
['Das', 'ist', 'eine', 'Liste']
```

```
# Zugriff auf einen Bereich von Elementen in einer Liste (mit Index)
print(list_2[2:4])
```

```
['eine', 'Liste']
```

```
# Vervielfachen von Listen
list_3 = ["Python", "ist", "eine", "Programmiersprache"]
print(list_3)
```

```
['Python', 'ist', 'eine', 'Programmiersprache']
```

```
print(2*list_3)
```

```
['Python', 'ist', 'eine', 'Programmiersprache', 'Python', 'ist', 'eine',
'Programmiersprache']
```

17.3 Länge einer Liste (Anzahl der Elemente)

- Länge einer Liste (Anzahl der Elemente) **len(liste)**

- Beispiele:

```
# Länge einer Liste ausgeben
zahlen = [1, 2, 3, 5, 7, 9, 12, 67, 78]
print("Länge der Liste 'zahlen': ", len(zahlen))

print(zahlen[1:4])

for i in range(0, len(zahlen)):
    print("Index", i, "Wert", zahlen[i])
```

```
[Länge der Liste 'zahlen':  9
[2, 3, 5]
Index 0 Wert 1
Index 1 Wert 2
Index 2 Wert 3
Index 3 Wert 5
Index 4 Wert 7
Index 5 Wert 9
Index 6 Wert 12
Index 7 Wert 67
Index 8 Wert 78]
```

17.4 Listen-Funktionen (Methoden)

- Vorkommen eines Elementes in der Liste zählen **.count("xy")**
- Index des ersten Vorkommens eines Elements in der Liste ermitteln **.index("xy")**
- Element an Liste anhängen **.append("abc")**
- Liste an eine Liste anhängen **.extend(["def","ghi"])**
- neues Element an einer bestimmten Stelle einfügen **.insert(2, "jkl")**
- Element an einer bestimmten Stelle entfernen **.pop(3)**
- entfernt das erste Element aus der Liste, das identisch ist mit **.remove("def")**
- dreht die Reihenfolge der Elemente um **.reverse()**
- sortiert die Elemente der Liste aufsteigend **.sort()**
- sortiert die Elemente der Liste absteigend **.sort(reverse=True)**
- sortiert die Elemente der Liste nach einem spezifischen Kriterium, z.B. nach der Länge der Elemente **.sort(key = len)**

- Beispiele:

```
# Vorkommen eines Elementes in der Liste zählen .count("xy")
l_1 = [16, 37, 31, 10, 27]
l_2 = [45, 76, 53, 28, 37]
l_gesamt = l_1 + l_2
print(l_1)
print(l_2)
print("37 kommt", l_1.count(37), "-mal in l_1 vor")
print("37 kommt", l_2.count(37), "-mal in l_2 vor")
print("37 kommt", l_gesamt.count(37), "-mal in l_gesamt vor")
```

```
[16, 37, 31, 10, 27]
[45, 76, 53, 28, 37]
37 kommt 1 -mal in l_1 vor
37 kommt 1 -mal in l_2 vor
37 kommt 2 -mal in l_gesamt vor
```

```
# Index des ersten Vorkommens eines Elements in der Liste ermitteln
# index("xy")
print("37 hat in l_gesamt den Index", l_gesamt.index(37))
```

```
37 hat in l_gesamt den Index 1
```

```
# Element an Liste anhängen    .append("abc")
l_gesamt.append("Ende")
print(l_gesamt)
```

```
[16, 37, 31, 10, 27, 45, 76, 53, 28, 37, 'Ende']
```

```
# Liste an eine Liste anhängen .extend(["def","ghi"])
l_gesamt.extend([9, 45, "Python"])
print("Länge von l_gesamt: ", len(l_gesamt))
print(l_gesamt)
```

```
Länge von l_gesamt:  14
[16, 37, 31, 10, 27, 45, 76, 53, 28, 37, 'Ende', 9, 45, 'Python']
```

```
# neues Element an einer bestimmten Stelle einfügen
# insert(2, "jkl")
l_gesamt.insert(10, "kein")
print(l_gesamt)
```

```
[16, 37, 31, 10, 27, 45, 76, 53, 28, 37, 'kein', 'Ende', 9, 45,
'Python']
```

```
# Element an einer bestimmten Stelle entfernen    .pop(3)
l_gesamt.pop(11)
print(l_gesamt)
l_gesamt.pop(10)
print(l_gesamt)
```

```
[16, 37, 31, 10, 27, 45, 76, 53, 28, 37, 'kein', 9, 45, 'Python']
[16, 37, 31, 10, 27, 45, 76, 53, 28, 37, 9, 45, 'Python']
```

```
# entfernt das erste Element aus der Liste, das identisch ist mit
#.remove("def")
l_gesamt.remove(37)
print(l_gesamt)
```

```
[16, 31, 10, 27, 45, 76, 53, 28, 37, 9, 45, 'Python']
```

```
# dreht die Reihenfolge der Elemente um       .reverse()
l_gesamt.reverse()
print(l_gesamt)
```

```
['Python', 45, 9, 37, 28, 53, 76, 45, 27, 10, 31, 16]
```

```
# sortiert die Elemente der Liste aufsteigend     .sort()
l_gesamt.remove("Python")
l_gesamt.sort()
print(l_gesamt)
```

```
[9, 10, 16, 27, 28, 31, 37, 45, 45, 53, 76]
```

```
# sortiert die Elemente der Liste absteigend      .sort(reverse=True)
l_gesamt.sort(reverse=True)
print(l_gesamt)
```

```
[76, 53, 45, 45, 37, 31, 28, 27, 16, 10, 9]
```

```
# sortiert die Elemente der Liste nach einem spezifischen Kriterium,
z.B. nach der Länge der Elemente  sort(key = len)
l_3 = ["Anfang", "12", "Python", "+4711", "3.14", "Ende"]
print(l_3)
l_3.sort(key=len)
print(l_3)
```

```
['Anfang', '12', 'Python', '+4711', '3.14', 'Ende']
['12', '3.14', 'Ende', '+4711', 'Anfang', 'Python']
```

Aufgabe

- Füge zwei vorher festgelegte Listen zu einer zusammen und gib diese aus
  Beispiel: [3,8,9,2] zusammengefügt mit [4,6] ergibt [3,8,9,2,4,6]

- Trenne eine Liste an einem Index in zwei Teillisten.
  Hinweis: Der Index beginnt bei 0. Getrennt wird vor dem Eintrag an der Indexstelle. Soll eine Liste an Index 0 getrennt werden, dann ist die erste Liste leer.
  Beispiel: [1,3,5,8,9,3] getrennt an Index 3 ergibt [1,3,5] [8,9,3]

- Von der Konsole sind beliebig viele Elemente einzulesen und einer Liste hinzuzufügen
  Mit der Eingabe q soll die Eingabe beendet werden.
  Die Liste ist zur Kontrolle auszugeben; anschließend ist die Liste sortiert auszugeben

- Erstelle eine while-Schleife über die Werte von der Konsole eingelesen und in eine Liste eingefügt werden sollen. Bei der Eingabe von q soll die Eingabeschleife beendet werden. Die so erstellte Liste ist auf der Konsole auszugeben.

- Erstelle eine Liste mit den Namen von fünf Großstädten. Diese Liste ist sortiert auszugeben. Über die Konsole sollen anschließend zwei weitere Städte zur Eingabe angefordert werden und zur Liste hinzugefügt werden. Diese Liste, nun mit 7 Städtenamen, ist wieder zu sortieren und anschließend in umgekehrter Reihenfolge auszugeben. Im nächsten Schritt werden die erste und die letzte Stadt aus der Liste entfernt und die reduzierte Liste ausgegeben.

- Entferne ein bestimmtes Element aus einer festgelegten Liste.
  Hinweis: Kommt ein Element mehr als einmal vor, sollte es überall entfernt werden.
  Beispiel: [3,8,9,5,1,3,6,4] ohne 3 ergibt [8,9,5,1,6,4]