```
# ------------------------------------------------------------------------
# ---- Skriptsprachenorientierte Programmiertechnik
# ---- Übungsaufgabe #4 - v1.2 - Prof. Endejan
# ---- Übungsaufgabe #4 - v1.2.a V. 2018-01 Prof. Engels
# ----
# ---- Beispiellösungen zu Übungsaufgaben; die Lösungen orientieren sich
# ---- an bereits besprochenen Sprachkonstrukten; andere (effizientere)
# ---- Lösungen sind ggf. möglich.
# ---- Fragen und Anmerkungen bitte direkt an mich
# ---- volker.engels@fom.de
# ----
# ---- Die doppelten Kommentarzeichen können über eine Selektion des Bereiches
# ---- und Alt + 4 entfernt werden (erneute Einkommentierung mit Alt + 3)
# ------------------------------------------------------------------------

import copy  # für Aufgabe 17 (deepcopy), s. u.


# ---- Hilfsfunktion für Aufgabenüberschrift
# ---- (nicht Teil der Übung; dient lediglich der Übersicht)
def aufgabe(text=''):
    global aufgaben_nr
    aufgaben_nr += 1
    print('\n'+5*'='+' Aufgabe '+str(aufgaben_nr), end='')
    if text:
        print(' ('+str(text)+')', end='')
    print(' '+5*'=')


aufgaben_nr = 0
# ---- Aufgabe 1 -----
# Erstellen Sie zwei Listen mit der Bezeichnung 'gerade' bzw. 'ungerade',
# in der alle geraden bzw. ungeraden Zahlen im Intervall [1;20] enthalten sind.
aufgabe()

gerade = list(range(2, 21, 2))
ungerade = list(range(1, 21, 2))

print("gerade  ", gerade)
print("ungerade", ungerade)


# ---- Aufgabe 2 -----
# Erstellen Sie eine neue Liste 'alle', die alle Elemente von 'gerade' und
# 'ungerade' enthält und sortieren Sie die Liste nach aufsteigenden Werten.
aufgabe()

alle = gerade + ungerade
print("alle:", alle)
alle.sort()
print("alle sortiert:", alle)

# alternativ mit Funktion sorted()
alle = sorted(gerade + ungerade)
print("alle sortiert (alternativ):", alle)


# ---- Aufgabe 3 -----
# Weisen Sie den Variablen a, b, c die ersten drei Werte von gerade zu
# (a erhält Wert von gerade[0] etc.). (Tuple Unpacking)
aufgabe()
```

```
62
63 a, b, c = gerade[:3]
64
65 print("a:", a, ", b:", b, ", c:", c)
66
67
68 # ---- Aufgabe 4 -----
69 # Erstellen Sie eine Liste 'gerade_mal_zehn', deren Elemente jeweils den
70 # 10-fachen Wert der Elemente von 'gerade' haben (z[0] = 10 *g[0]).
71 aufgabe()
72
73 gerade_mal_zehn = [x*10 for x in gerade]
74
75 print("gerade * 10:", gerade_mal_zehn)
76
77
78 # ---- Aufgabe 5 -----
79 # Ersetzten Sie jedes zweite Element von 'gerade' durch jede zweite Zahl von
80 # 'gerade_mal_zehn' (jeweils angefangen bei Position 2 bzw. Index 1).
81 aufgabe()
82
83 gerade[1::2] = gerade_mal_zehn[1::2]
84
85 print("gerade nach Ersetzung", gerade)
86
87 # ---- Aufabe 6 -----
88 # Sortieren Sie die Liste gerade_mal_zehn in absteigender Reihenfolge.
89 aufgabe()
90
91 gerade_mal_zehn.sort(reverse=True)
92
93 # alternativ
94 # gerade_mal_zehn.sort()
95 # gerade_mal_zehn.reverse()
96
97 print(gerade_mal_zehn)
98
99
100 # ---- Augabe 7 -----
101 # Erstellen Sie eine neue Liste zahlen mit den ganzen Zahlen im Intervall
102 # [-4;3]. Sortieren Sie die Liste anschließend aufsteigend nach dem Betrag
103 # der Elemente (Ergebnis: [0,-1,1,-2,2,-3,3,-4]).
104 # (Python bietet die Funktion abs() zur Berechnung des Betrags einer Zahl.)
105 aufgabe()
106
107 zahlen = list(range(-4, 4))
108 print("zahlen:", zahlen)
109
110 zahlen.sort(key=abs)
111
112 print("zahlen nach Betrag sortiert", zahlen)
113
114 # ---- Aufgabe 8 -----
115 # Erstellen Sie eine Liste farben mit den Elementen
116 # "blau", "gelb", "grün", "braun", "schwarz".
117 aufgabe()
118
119 farben = ["blau", "gelb", "grün", "braun", "schwarz"]
120
121 print("Initialiserung:", farben)
122
```

```
123 # ---- Aufabe 9 -----
124 # Fügen Sie an die erste Position (Index 0) sowie zwischen den bereits
125 # bestehenden Elementen  jeweils ein neues Element mit dem Wert „weiß“
126 # ein (am Anfang sowie zwischen jedem der ursprünglichen Elemente von
127 # farben soll „weiß“ erscheinen).
128 aufgabe()
129
130 for i in range(len(farben)):
131     farben.insert(i*2, "weiß")
132     # farben[i*2:i*2] = "weiß",    # alternativ (Komma beachten!)
133
134 print("weiß eingefügt:", farben)
135
136 # Alternative (vom Ende der Liste her anfangen)
137 for i in range(len(farben)-1, -1, -1):
138    farben.insert(i, "weiß")
139
140 # Alternative mit Liste Comprehension
141 # (auf Nachfrage eingefügt)
142 farben = ["weiß" if i%2==0 else farben[(i-1)//2] for i in range(len(farben)*2)]
143
144 # oder mit neuer Liste...
145 farben = ["blau", "gelb", "grün", "braun", "schwarz"]
146
147 neu = len(farben) * 2 * ['weiß']  # Liste mit 10 x 'weiß' erstellen ...
148 neu[1::2] = farben  # ... und jedes 2. Element durch farben ersetzen
149 farben = neu
150 print("weiß eingefügt (2):", farben)
151
152
153 # ---- Aufgabe 10 -----
154 # Ersetzten Sie das erste Element von farben durch den Wert „rot“.
155 aufgabe()
156
157 farben[0] = "rot"
158
159 print("rot an Anfang:", farben)
160
161
162 # ---- Aufgabe 11 -----
163 # Löschen Sie, angefangen bei Position 1 (Index 0), das jeweils
164 # zweite Element von farben.
165 aufgabe()
166
167 del farben[::2]
168
169 print("jedes 2. gelöscht:", farben)
170
171
172 # ---- Aufgabe 12 -----
173 # Erstellen Sie eine neue Liste farben2 als Kopie von farben.
174 aufgabe()
175
176 farben2 = farben.copy()
177 # farben2 = farben[:]  # Alternative Schreibweise
178
179 print("farben2:", farben2)
180
181
182 # ---- Aufgabe 13 -----
183 # Drehen Sie die Reihenfolge der Elemente von farben um.
```

```
184 aufgabe()
185
186 farben.reverse()
187
188 print("reverse:", farben)
189
190
191 # ---- Aufgabe 14 -----
192 # Geben Sie die Werte von farben der Reihe nach aus (angefangen von Pos. 1 bis
193 # zum letzten Element) und entfernen Sie dabei das jeweils ausgegebene Element
194 # aus der Liste.
195 aufgabe()
196
197 # for element in range(len(farben)):
198 #     print(farben.pop(0), "entfernt; Rest:", farben)
199
200 while farben:
201     print(farben.pop(0), "entfernt; Rest:", farben)
202
203 # ---- Aufgabe 15 -----
204 # Hängen Sie den Wert „blau“ an die Liste farben2 an.
205 aufgabe()
206
207 farben2.append("blau")
208 # farben2 += "blau",
209 # farben2 += ["blau"]
210
211 print("farben2:", farben2)
212
213 # ---- Aufgabe 16 -----
214 # Entfernen Sie das erste in farben2 auftretende „blau“ aus der Liste.
215 aufgabe()
216
217 farben2.remove("blau")
218
219 # sichere Variante (falls 'blau' nicht in farben2 enthalten ist)
220 try:
221     farben2.remove("blau")
222 except ValueError:
223     pass
224
225 # Alternative 1
226 if "blau" in farben2:
227     farben2.remove("blau")
228
229 # Alternative 2 (Ausnutzung der lazy evaluation)
230 "blau" in farben2 and farben2.remove("blau")
231
232 # Alternative 3 (Nutzung einer Conditional Expression)
233 farben2.remove("blau") if "blau" in farben2 else None
234
235
236 # ---- Aufgabe 17 -----
237 # Gegeben sei folgende Liste: liste  = [0,1,[20,21,22],[30,31,32],(40,41,42)]
238 # sowie eine Kopie der Liste, die erzeugt wurde über kopie  = liste.copy()
239 # Welche Listen-Inhalte bzw. Resultate erwarten Sie beim Aufruf der folgenden
240 # Anweisungen?
241 aufgabe()
242
243 liste = [0, 1, [20, 21, 22], [30, 31, 32], (40, 41, 42)]
244 kopie = liste.copy()  # Alternative: kopie = liste[:]
```

```
245
246 print("liste:", liste)
247 print("kopie:", kopie)
248
249
250 # ---- Inhalte von liste und kopie nach den folgenden Operationen
251 def ausfuehren(anweisung):
252     """Hilfsfunktion zur Anzeige und Ausführung einer anweisung."""
253     print("Anweisung:", anweisung)
254     try:
255         exec(anweisung)
256     except TypeError as fehler:
257         print("--> Fehler:", fehler)
258         return
259     print("--> Resultat")
260     print("--> liste:", liste)
261     print("--> kopie:", kopie)
262
263 ausfuehren("liste[0] = 'null'")
264 ausfuehren("kopie[1] = 'EINS'")
265 ausfuehren("kopie[2] = 'ZWEI'")
266 ausfuehren("kopie[3][0] = 'DREI-NULL'")
267 ausfuehren("kopie[4][0] = 'VIER-NULL'")
268
269 # ------------------------------------------------------------------------
270 # liste: [0, 1, [20, 21, 22], [30, 31, 32], (40, 41, 42)]
271 # kopie: [0, 1, [20, 21, 22], [30, 31, 32], (40, 41, 42)]
272 # Anweisung: liste[0] = 'null'
273 # --> Resultat
274 # --> liste: ['null', 1, [20, 21, 22], [30, 31, 32], (40, 41, 42)]
275 # --> kopie: [0, 1, [20, 21, 22], [30, 31, 32], (40, 41, 42)]
276 # Anweisung: kopie[1] = 'EINS'
277 # --> Resultat
278 # --> liste: ['null', 1, [20, 21, 22], [30, 31, 32], (40, 41, 42)]
279 # --> kopie: [0, 'EINS', [20, 21, 22], [30, 31, 32], (40, 41, 42)]
280 # Anweisung: kopie[2] = 'ZWEI'
281 # --> Resultat
282 # --> liste: ['null', 1, [20, 21, 22], [30, 31, 32], (40, 41, 42)]
283 # --> kopie: [0, 'EINS', 'ZWEI', [30, 31, 32], (40, 41, 42)]
284 # Anweisung: kopie[3][0] = 'DREI-NULL'
285 # --> Resultat
286 # --> liste: ['null', 1, [20, 21, 22], ['DREI-NULL', 31, 32], (40, 41, 42)]
287 # --> kopie: [0, 'EINS', 'ZWEI', ['DREI-NULL', 31, 32], (40, 41, 42)]
288 # Anweisung: kopie[4][0] = 'VIER-NULL'
289 # --> Fehler: 'tuple' object does not support item assignment
290 # ------------------------------------------------------------------------
291
292
293 # ---- Lösung: Nutzung einer tiefen Kopie
294 aufgaben_nr -= 1
295 aufgabe('Alternative mit deepcopy()')
296
297 # import copy (s.o.)
298 liste = [0, 1, [20, 21], [30, 31], (40, 41)]
299 kopie = copy.deepcopy(liste)
300
301 ausfuehren("liste[0] = 'null'")
302 ausfuehren("kopie[1] = 'EINS'")
303 ausfuehren("kopie[2] = 'ZWEI'")
304 ausfuehren("kopie[3][0] = 'DREI-NULL'")
305 ausfuehren("kopie[4][0] = 'VIER-NULL'")
```

```
306
307 # ------------------------------------------------------------------------
308 # Anweisung: liste[0] = 'null'
309 # --> Resultat
310 # --> liste: ['null', 1, [20, 21], [30, 31], (40, 41)]
311 # --> kopie: [0, 1, [20, 21], [30, 31], (40, 41)]
312 # Anweisung: kopie[1] = 'EINS'
313 # --> Resultat
314 # --> liste: ['null', 1, [20, 21], [30, 31], (40, 41)]
315 # --> kopie: [0, 'EINS', [20, 21], [30, 31], (40, 41)]
316 # Anweisung: kopie[2] = 'ZWEI'
317 # --> Resultat
318 # --> liste: ['null', 1, [20, 21], [30, 31], (40, 41)]
319 # --> kopie: [0, 'EINS', 'ZWEI', [30, 31], (40, 41)]
320 # Anweisung: kopie[3][0] = 'DREI-NULL'
321 # --> Resultat
322 # --> liste: ['null', 1, [20, 21], [30, 31], (40, 41)]
323 # --> kopie: [0, 'EINS', 'ZWEI', ['DREI-NULL', 31], (40, 41)]
324 # Anweisung: kopie[4][0] = 'VIER-NULL'
325 # --> Fehler: 'tuple' object does not support item assignment
326 # ------------------------------------------------------------------------
327
```