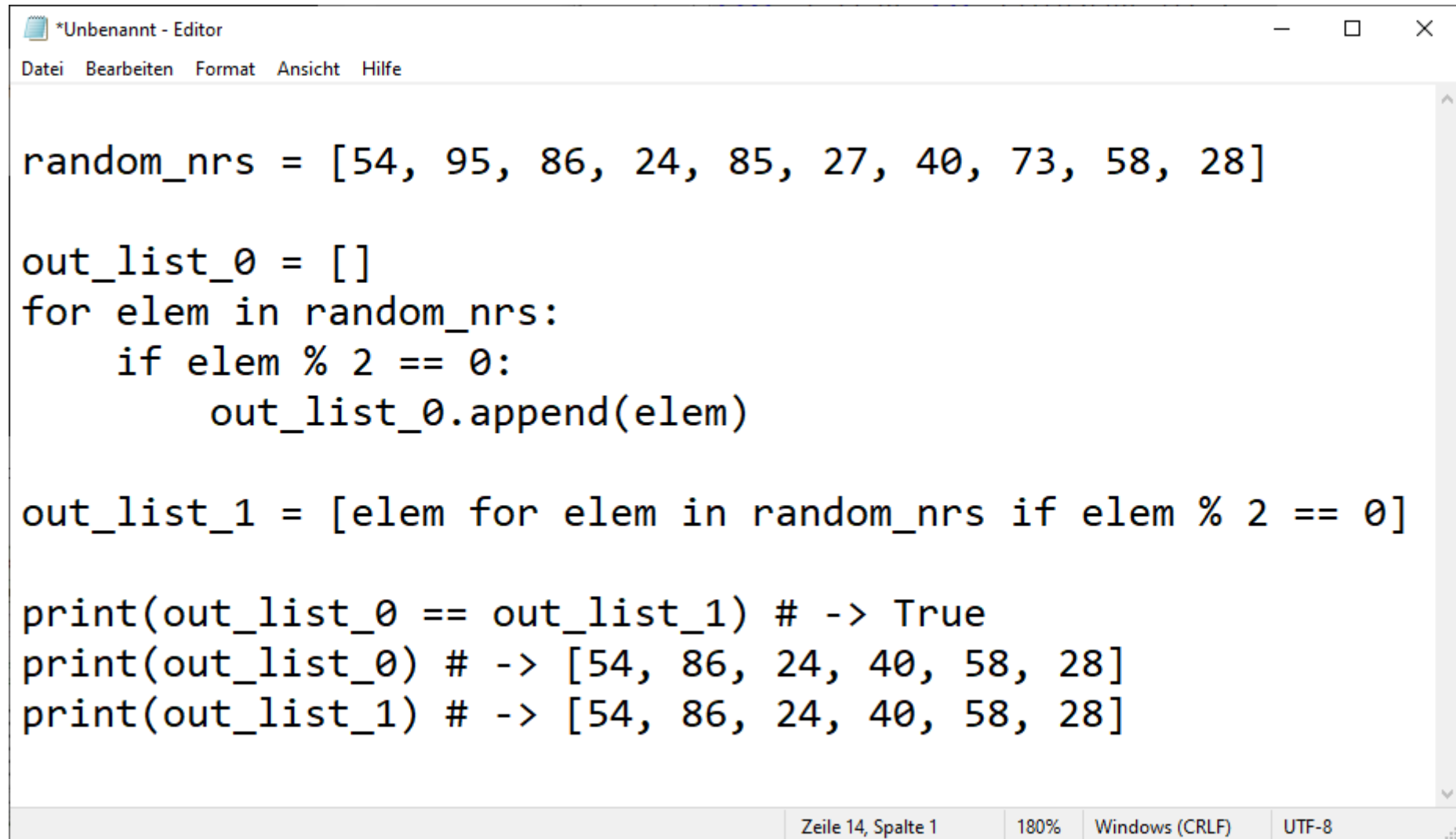


Comprehension

List Comprehension



```
*Unbenannt - Editor
Datei Bearbeiten Format Ansicht Hilfe

random_nrs = [54, 95, 86, 24, 85, 27, 40, 73, 58, 28]

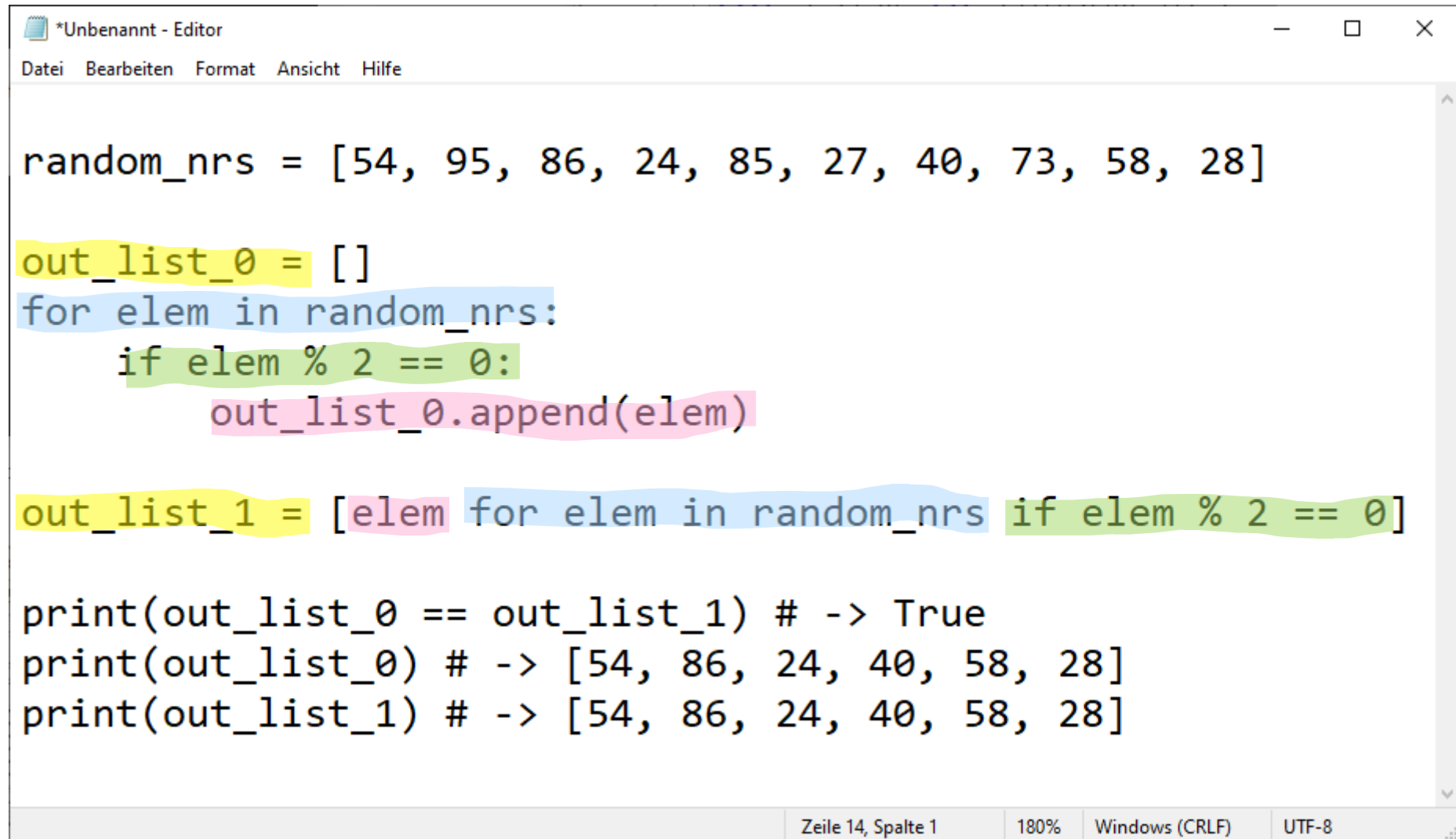
out_list_0 = []
for elem in random_nrs:
    if elem % 2 == 0:
        out_list_0.append(elem)

out_list_1 = [elem for elem in random_nrs if elem % 2 == 0]

print(out_list_0 == out_list_1) # -> True
print(out_list_0) # -> [54, 86, 24, 40, 58, 28]
print(out_list_1) # -> [54, 86, 24, 40, 58, 28]
```

Zeile 14, Spalte 1 180% Windows (CRLF) UTF-8

List Comprehension

A screenshot of a Python IDE window titled '*Unbenannt - Editor'. The window contains a Python script. The first line is 'random_nrs = [54, 95, 86, 24, 85, 27, 40, 73, 58, 28]'. The second line is 'out_list_0 = []'. The third line is 'for elem in random_nrs:'. The fourth line is 'if elem % 2 == 0:'. The fifth line is 'out_list_0.append(elem)'. The sixth line is 'out_list_1 = [elem for elem in random_nrs if elem % 2 == 0]'. The seventh line is 'print(out_list_0 == out_list_1) # -> True'. The eighth line is 'print(out_list_0) # -> [54, 86, 24, 40, 58, 28]'. The ninth line is 'print(out_list_1) # -> [54, 86, 24, 40, 58, 28]'. The IDE has a menu bar with 'Datei', 'Bearbeiten', 'Format', 'Ansicht', and 'Hilfe'. The status bar at the bottom shows 'Zeile 14, Spalte 1', '180%', 'Windows (CRLF)', and 'UTF-8'.

```
*Unbenannt - Editor
Datei Bearbeiten Format Ansicht Hilfe

random_nrs = [54, 95, 86, 24, 85, 27, 40, 73, 58, 28]

out_list_0 = []
for elem in random_nrs:
    if elem % 2 == 0:
        out_list_0.append(elem)

out_list_1 = [elem for elem in random_nrs if elem % 2 == 0]

print(out_list_0 == out_list_1) # -> True
print(out_list_0) # -> [54, 86, 24, 40, 58, 28]
print(out_list_1) # -> [54, 86, 24, 40, 58, 28]
```

Zeile 14, Spalte 1 180% Windows (CRLF) UTF-8

List Comprehension

- Allgemein:

```
[elem for elem in random_nrs if elem % 2 == 0]
```

Mache das für dieses Iterable in dieser Situation

- elem und elem müssen gleich heißen!
- Bedingung ist optional
- List Comprehension ist nie die einzige Lösung!
 - Abwägen zwischen Speicherbedarf und Performance

Übung

- Szenario:
Du bist Entwickler in einem
Online Shop

- Feature Request:
Erzeuge eine Liste mit
allen Kopfhörern,
die die Farbe haben,
die der Nutzer auswählt.

Farbe

Suchen...

☒ Blau (45)

☒ Gelb (3)

☐ Beige (10)

☐ Braun (8)

☐ Creme (2)

Mehr anzeigen (12)

★★★★★ (138)

JBL Tune 230NC True Wireless, In-ear Kopfhörer Black



Signalübertragung

True wireless, kabellos

Maximale Betriebsdauer

Maximale Musikwiedergabezeit mit ANC aus 10 ...

Gewicht (laut Hersteller)

57.5 g

Lieferumfang

1x JBL Tune 230NC TWS Kopfhörer, 1x Ladestati...

● Lieferung 31.05.2022 - 02.06.2022

● Marktabholung 30 min nach Bestellbestätigung
Sulzbach (Main-Taunus-Zentrum) (Markt ändern)

UVP 99,99 **84,99**
inkl. MwSt. versandkostenfrei



★★★★★ (0)

SENNHEISER SPORT True Wireless, In-ear Kopfhörer Bluetooth Black



Signalübertragung

True wireless, kabellos

Maximale Betriebsdauer

9 Std. + bis zu 27 Std. über die Ladebox

Gewicht (laut Hersteller)

55 g

Lieferumfang

Paar Ohrhörer, Ladebox mit Bandanhänger, USB...

● Lieferung 31.05.2022 - 02.06.2022

● Marktabholung 30 min nach Bestellbestätigung
Sulzbach (Main-Taunus-Zentrum) (Markt ändern)

129,90
inkl. MwSt. versandkostenfrei
0% Finanzierung: In 12 Raten à 10,83 €**



★★★★★ (5)

XIAOMI Redmi Buds 3 Lite, In-ear Kopfhörer Bluetooth Schwarz



Signalübertragung

kabellos, True wireless

Maximale Betriebsdauer

5 Std. Earbuds, 18 Std. Ladecase

● Lieferung 31.05.2022 - 02.06.2022

● Marktabholung 30 min nach Bestellbestätigung
Sulzbach (Main-Taunus-Zentrum) (Markt ändern)

LC mit nested loops

```
chars = ["a", "b", "c"]
nrs = [10, 20, 30, 40, 50]

chars_nrs = []
for c in chars:
    for i in nrs:
        chars_nrs.append((c, i))

print(chars_nrs)
# [('a', 10), ('a', 20), ('a', 30), ('a', 40), ('a', 50),
#  ('b', 10), ('b', 20), ('b', 30), ('b', 40), ('b', 50),
#  ('c', 10), ('c', 20), ('c', 30), ('c', 40), ('c', 50)]

chars_nrs = [(c, i) for c in chars for i in nrs]

print(chars_nrs)
# [('a', 10), ('a', 20), ('a', 30), ('a', 40), ('a', 50),
#  ('b', 10), ('b', 20), ('b', 30), ('b', 40), ('b', 50),
#  ('c', 10), ('c', 20), ('c', 30), ('c', 40), ('c', 50)]
```

Comprehension mit anderen Datentypen

- **Tupel** gehen leer aus:

```
61 print((i for i in range(10)))
```

```
<generator object <genexpr> at 0x0000023677763F40>
```

(natürlich kann man `tuple(i for i in range(10))` nutzen

- **Sets** nutzen die gleiche Syntax wie Listen (`{...}` statt `[...]`)
- **Dictionaries** nutzen diese Struktur:

`{key: value for elem in iterable}`

- key und value müssen natürlich existierende Namen sein!

Fallstricke

- LC sollte nicht genutzt werden, wenn:
 - Es *zu viele* Bedingungen gibt, nach denen gefiltert werden soll
 - Wenn verschiedene Listen erzeugt werden sollen als Ergebnis der Iteration
 - Besser 1x iterieren und mehrere Listen erzeugen
 - Generell: Wenn der Befehl zu lang wird
- Fazit: Mit Bedacht verwenden

Hausaufgabe

- Siehe 01_Hausi.py

Weitere Infos

- <https://docs.python.org/3/tutorial/datastructures.html#list-comprehensions>
- Sehr lesenswert:
<https://docs.python.org/3/tutorial/datastructures.html#nested-list-comprehensions>