1. Bildung von Listen mit `L = [...]`
2. Listensyntax `L[0]`, `L[1]`, ...
3. Bedeutung negativer Listenindizes wie `L[-1]`
4. Mehrfachindizes bei verschachtelten Listen wie `L[3][2]`
5. `IndexError` bei einem Index, dem kein Element entspricht
6. Slices: `L[a:b]`, `L[a:]`, `L[:b]`, `L[a:]`, `L[a:b:step]`

Beispiele: `L = [4, 7, 8, 1, 6, 9, 2, 3]`

- `L[3:7] ⇒ [1, 6, 9, 2]`
- `L[3:] ⇒ [1, 6, 9, 2, 3]`
- `L[5:5] ⇒ []`
- `L[1:5:2] ⇒ [7, 1]`
- `L[5:1:-2] ⇒ [9, 1]` (zählt rückwärts, Start und Ende vertauscht)
- `L[::-3] ⇒ [3, 6, 7]`

7. Listenfunktionen: `len(L)`, `sorted(L)`, `max(L)`, `min(L)` `sum(L)`
8. Listenoperatoren `+` und `*`
9. Listenmethoden: `L.append(element)`, `L.pop()`, `L.pop(index)`, `L.insert(i, element)`, `L.index(element)`, `L.remove(element)`, `L.reverse()`, `L.sort()`, `L.extend(list)`
L steht für eine Liste; *element* und *index* stehen jeweils für ein Element bzw. einen Index.
10. Du weißt, dass das Kopieren einer Listenvariablen nur das Kopieren einer Listenreferenz bedeutet und dass eine Änderungen an der Kopie eine Änderung am Original bewirkt.
11. Du weißt, dass eine Slice jeweils eine „echte“ Kopie einer (Teil-)Liste im Speicher erzeugt.
12. Listen mit `for` iterieren (durchlaufen): `for <var> in <liste>:`
`<block>`

13. Du kannst einfache List-Comprehensions (d.h. ohne `if` und `if-else`) auswerten.

Beispiele:

- `[x**2 for x in [3, 5, 2]] ⇒ [9, 25, 4]`
- `[2*i for i in range(9, 3, -2)] ⇒ [18, 14, 10]`

14. Du kannst Funktionen definieren, welche
 - das Maximum (oder das Minimum) der Elemente einer Liste berechnen und als Wert zurückgeben,
 - die Summe (oder das Produkt) der Elemente einer Liste berechnen und als Wert zurückgeben.