Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

FS24 CAS PML - Python

Niklaus Johner

niklausbernhard.johner@bfh.ch

FS24 CAS PML - Python

8. More about lists

Listen Initialisierung

- Literale Initialisierung

```
In [4]: l1 = []
   ...: l2 = [1, 2, 3]
   ...: print(l1, l2)
   ...:
[] [1, 2, 3]
```

- Klassen Initialisierung von einem iterierbaren Objekt

```
In [5]: l3 = list("abcd")
   ...: l4 = list(range(4))
   ...: print(l3, l4)
   ...:
['a', 'b', 'c', 'd'] [0, 1, 2, 3]
```

Neue Elementen einfügen

- *li.append(el)*:
  fügt ein Element *el* am Ende der Liste *li* ein.

```
In [6]: l1 = [1]
   ...: l1.append(2)
   ...: print(l1)
   ...:
[1, 2]
```

- *li.insert(pos, el)*:
  fügt ein Element *el* an der Position *pos* der Liste *li* ein.

```
In [7]: l1.insert(1, 3)
   ...: print(l1)
   ...:
[1, 3, 2]
```

- *li.extend(seq)*:
  fügt alle Elementen von *seq* am Ende der Liste *li* ein.

```
In [13]: l1.extend([2, 3])
    ...: print(l1)
    ...:
[1, 3, 2, 2, 3]
```

Neue Elementen einfügen

- Eine Sequenz wird mit *append* als ein Element in einer Liste eingefügt, wohingegen mit *extend* jedes Element der Sequenz in diese Liste eingefügt wird

```
In [15]: l1 = [1]
    ...: l2 = [1]
    ...: l1.append([2, 3])
    ...: l2.extend([2, 3])
    ...: print("l1=",l1)
    ...: print("l2=",l2)
    ...:
l1= [1, [2, 3]]
l2= [1, 2, 3]
```

Elemente löschen

- *li.pop()*:
  Das letzte Element wird von *li* herausgenommen und zurückgegeben.

```
In [21]: l1 = [1, 2, 3, 4]
    ...: el = l1.pop()
    ...: print("el =",el)
    ...: print("l1 =",l1)
    ...:
el = 4
l1 = [1, 2, 3]
```

- *li.pop(pos)*:
  Das Element an der Position *pos* wird von *li* herausgenommen und zurückgegeben.

```
In [22]: el = l1.pop(0)
    ...: print("el =",el)
    ...: print("l1 =",l1)
    ...:
el = 1
l1 = [2, 3]
```

- *li.remove(el)*:
  Das Element *el* wird von *li* gelöscht.

```
In [20]: l1 = [1, 2, 1, 3]
    ...: l1.remove(1)
    ...: print("l1 =",l1)
    ...:
l1 = [2, 1, 3]
```

Elemente zuweisen

- Man kann einem Element einen neuen Wert zuweisen

```
In [5]: words = ["a", "thousand", "words"]
   ...: words[1] = "few"
   ...: print(words)
['a', 'few', 'words']
```

Liste sortieren

- *li.sort()*: Die Liste *li* sortieren
  - *reverse* Parameter kontrolliert die Richtung der Sortierung

```
In [50]: l1 = [1, 2, 0, 4]
    ...: l1.sort()
    ...: print("l1=", l1)
    ...:
l1= [0, 1, 2, 4]
```

```
In [51]: l1 = [1, 2, 0, 4]
    ...: l1.sort(reverse=True)
    ...: print("l1=", l1)
    ...:
l1= [4, 2, 1, 0]
```

Liste sortieren

- *sorted(li)*: gibt eine sortierte Liste zurück
  - *reverse* Parameter kontrolliert die Richtung der Sortierung

```
In [53]: l1 = [1, 2, 0, 4]
    ...: l2 = sorted(l1)
    ...: l3 = sorted(l1, reverse=True)
    ...: print("l1=", l1,"l2=", l2,"l3=", l3)
    ...:
l1= [1, 2, 0, 4] l2= [0, 1, 2, 4] l3= [4, 2, 1, 0]
```

Nested lists

- Listen können andere Listen enthalten (nested list)
- Man kann dann mit nested for-loops über die Elemente iterieren

```
In [60]: ll = [[1, 2], ["Bla", "bla"]]
    ...: for l1 in ll:
    ...:     for el in l1:
    ...:         print(l1, el)
    ...:
[1, 2] 1
[1, 2] 2
['Bla', 'bla'] Bla
['Bla', 'bla'] bla
```