FACHOBERSCHULE BERUFSOBERSCHULE BAMBERG	Informatik		Datum:
	Name:	Klasse:	

Lektion 7a: Listen

Was Sie in Python nun unter der Bezeichnung **Liste** kennenlernen, würde in den meisten anderen Programmiersprachen als **array** bezeichnet werden. Eine Liste ist wohl der wichtigste Datentyp zur Speicherung einer variablen Anzahl an Elementen. Die Ordnung der Elemente bleibt erhalten, es können jederzeit und an jeder beliebigen Stelle Elemente hinzugefügt oder gelöscht werden.

Listen erstellen

Die einfachste Möglichkeit besteht darin, Werte eines beliebigen Typs (Zahlen, Wörter,..) in **eckige Klammern** zu schreiben. Die Werte in einer Liste bezeichnet man dann als **Elemente**.
Setzen wir das doch gleich mal um. In unserer Klasse haben wir aktuell Tobias, Mara und Emre. Schreiben wir sie in eine Liste:

```
schueler=["Tobias","Mara","Emre"]
```

Elemente

Indizierung und Operationen auf Listen

Natürlich möchten wir auch manchmal nachsehen können, welche Elemente sich in unserer Liste befinden. Wenn der Lehrer einen Überblick über alle Elemente (schueler) haben möchte, geht das mit Hilfe des print-Befehls.

```
print(schueler)
```

Es kann aber auch nützlich sein, auf ein einzelnes Element in der Liste zugreifen zu können. Die Syntax für den Zugriff auf die Elemente einer Liste ist: der Klammer-Operator. Den Ausdruck innerhalb der Klammern nennt man **Index**.
print(schueler[**1**]) = Mara

Der **Slice-Operator** lässt sich noch vielfältiger benutzen:

```
schueler=["Tobias","Mara","Emre"]
            0        1       2
```

`print(schueler[1:2])`	→	Mara (von 1 bis 2, aber zweiter Index nicht inkludiert)
`print(schueler[:3])`	→	Tobias, Mara, Emre (entspricht [0:3])
`print(schueler[1:])`	→	Mara, Emre (alle Listenelemente ab 1 anzeigen)
`print(schueler[:])`	→	Tobias, Mara, Emre (alle Listenelemente anzeigen)

Listen verändern

Listenelemente überschreiben
Beispiel: `schueler[2] = "Milo"`
Folge: Emre war vorher auf Position 2 in der Liste - Position 2 („Emre“) wurde durch Milo ersetzt

Listenelemente sortieren: *sort()-Methode.*
Beispiel: `schueler.sort()`
Diese Methode sortiert die Elemente einer Liste in aufsteigender Reihenfolge.

Neue Listenelemente aufnehmen: *append()-Methode*
Beispiel: `schueler.append("Lilly")`

Listenelemente löschen:

1. Wenn Sie den Index des gewünschten Elements kennen, können Sie die Methode ***pop()*** verwenden:
   schueler.pop(3)
2. Kennen Sie das Element, das entfernt werden soll, aber nicht seinen Index, empfiehlt sich ***remove()***
   Beispiel: schueler.remove("Lilly")
3. Möchten Sie mehrere Elemente löschen, können Sie **del()-Methode mit einem Slice-Index** verwenden.
   Beispiel: del(schueler[1:3])
   Folge: Löscht die Schüler mit dem Index 1 und 2

Operationen auf Listen

Die Länge einer Liste erhält man mit dem Befehl "**len(liste)**". Die Länge ist immer um eins größer als der Index des letzten Elements der Liste.

```
# initialisiere eine vorgefüllte Liste
haustiere = ["Hund", "Katze", "Hamster"]

# und gebe die Länge der Liste aus
länge_der_liste = len(haustiere)
print(länge_der_liste)

```

```
3
```

Man kann auch mit dem **in-Operator** den Wahrheitswert erhalten, ob ein Element in einer Liste vorhanden ist.

```
# initialisiere eine vorgefüllte Liste
zahlen = [1, 2, 3, 4, 5, 6]

# wenn das Element '10' in der Liste der Zahlen ist, gebe einen Text aus
# die '10' ist hier aber nicht in der Liste dabei
if 10 in zahlen:
    print("Die 10 ist dabei!")
else:
    print("Die 10 ist nicht dabei!")

```

```
Die 10 ist nicht dabei!
```

Man kann auch den kleinsten **min(liste)** oder größten Wert mit **max(liste)** aus einer Liste von Zahlen herausholen.

```
# initialisiere eine vorgefüllte Liste
zahlen = [1, 2, 3, 4, 5, 6]

# gebe die kleinste Zahl in der Liste aus
kleinste_zahl = min(zahlen)
print(kleinste_zahl)

# gebe die größte Zahl in der Liste aus
größte_zahl = max(zahlen)
print(größte_zahl)

```

```
1
6
```

Verschachtelte Listen

Man kann auch Listen als Elemente in eine Liste einfügen. Dann hat man eine Liste von Listen, was einer Tabelle entspricht.

Tier	Weißer Hai	Hund	Stubenfliege
Größe	groß	mittel	klein
Farbe	grau-weiß	braun	schwarz
Art	Fisch	Säugetier	Insekt
Lebensraum	Wasser	Erde	Luft

```
# initialisiere die Liste von Listen, das ist nichts anderes als
# eine Tabelle
liste_von_listen = []
tiere = ["Weißer Hai", "Hund", "Stubenfliege"]
größe = ["groß", "mittel", "klein"]
farbe = ["grau-weiß", "braun", "schwarz"]
art = ["Fisch", "Säugetier", "Insekt"]
lebensraum = ["Wasser", "Erde", "Luft"]

# und füge die Elemente in die Liste von Listen ein
liste_von_listen.append(tiere)
liste_von_listen.append(größe)
liste_von_listen.append(farbe)
liste_von_listen.append(art)
liste_von_listen.append(lebensraum)

# gebe die Tabelle aus
print(liste_von_listen)

```

```
[['Weißer Hai', 'Hund', 'Stubenfliege'], ['groß', 'mittel', 'klein'], ['grau-weiß', 'braun',
'schwarz'], ['Fisch', 'Säugetier', 'Insekt'], ['Wasser', 'Erde', 'Luft']]
```

List Comprehension

Aufgabe: Erstellen Sie eine Liste xs in der die Zahlen 1 – 8 abgespeichert sind. Erstellen Sie zudem eine leere Liste ys. Schreiben Sie ein Programm, dass die Quadratzahlen zu den Zahlen aus xs in ys schreibt.

```
xs=[1,2,3,4,5,6,7,8]
ys=[ ]
for i in xs:
   ys.append(i*i)
print(ys)
```

```
#Kompaktere Schreibweise:
xs=[1,2,3,4,5,6,7,8]
ys=[i*i for i in xs]
print(ys)
```

List Comprehension ist eine kompaktere Schreibweise, diese dienen dazu Code zu sparen und sind bei langen Programmen übersichtlicher

FACHOBERSCHULE BERUFSOBERSCHULE BAMBERG	Informatik		Datum:
	Name:	Klasse:	

Lektion 7b: Dictionaries

Dictionaries (auf deutsch "Wörterbücher") sind den Listen ähnlich. Jedoch ist der Index nicht eine Reihe von Ganzzahlen, die bei 0 beginnt und die Elemente der Liste durchzählt. Bei Dictionaries ist der Index eine beliebige Zeichenkette oder ein beliebiger Zahlenwert, z. B. die Kommazahl 4.9 oder die Zeichenkette "telefonnummer". Auf diese Weise kann man jedes Element im Dictionary mit einem eindeutigen Namen ansprechen. Dieser Name heißt "Schlüssel" und das dazugehörige Element heißt "Wert", zusammen bilden sie ein **"Schlüssel-Wert-Paar"**. Im Englischen spricht man auch von **"key-value-pairs"**.

Definition eines Dictionaries

```
d= {„München“ : „MUC“, „Budapest“ : „BUD“, „Helsinki“ : „HEL“}
```

Die Schlüssel (keys) sind hier München, Budapest und Helsinki.
Die zugeordneten Werte (values) sind „MUC“, „BUD“ und „HEL“.

Operationen auf Dictionaries

Wertzuordnungen können gespeichert werden, z.B. Nachname und Nummer in einem Telefonbuch (dies ist in einer Liste nicht möglich!)

```
d= {“München“ : “MUC“, “Budapest“ : “BUD“, “Helsinki“ : “HEL“}

print (d)                   # Dictionary ausgeben

print (d[“München“])        # zugeordneten Wert ausgeben, hier also MUC

d[“Saigon“] = “SGN“         # Dictionary wird nachträglich ein Eintrag (Key und Value) hinzugefügt

del d[“Budapest“]           # Eintrag wird aus Dictionary gelöscht

for key in d:               #mit for-Schleife auf Schlüssel bzw. Werte zugreifen
      value = d[key]
      print(key)                  #Alle Schlüssel werden ausgegeben
      print(value)                #Alle Werte werden ausgegeben
```

alternativ mit Methode .items()

```
for key, value in d.items():
    print(key, value)
```