# Übungen 3

**Aufgabenstellung:**

Wir definieren eine neue, aus **`dict`** abgeleitete Klasse namens **`Lexicon`**, die als Grundlage für Wörterbücher im linguistischen Sinne dienen soll, bei denen eine Abbildung von Wortformen auf lexikalische Kategorien vorliegt. Für diesen Zweck werden wir Methoden definieren, die die Arbeit mit derartigen Wörterbüchern vereinfachen.

**Vorarbeit**

Legen Sie in Python einen Ordner namens *Lexikon* an und darin eine Datei namens *lexicon.py*. Alle der folgenden Aufgaben sind in dieser Datei auszuführen.

**Aufgabe 1 (optional): `pairlist()`**

Definieren Sie eine ungebundene Funktion **`pairlist()`**, die zwei Zeichenketten als Argumente nimmt und wie folgt eine Liste von Tupeln zurückgibt:

```
>>> tupel = pairlist('a b c d','e f g h')
>>> tupel
[('a', 'e'), ('b', 'f'), ('c', 'g'), ('d', 'h')]
```

Verwenden Sie hierfür die **`zip`**-Funktion. Denken Sie an den Einsatz von **`.split()`**, um zu verhindern, dass Leerzeichen als Tupel übergeben werden.

**Aufgabe 2:** die Klasse **`Lexicon`**

Definieren Sie eine neue Klasse **`Lexicon`**, die aus **`dict`** abgeleitet ist derart, dass die folgenden Anweisungen bzw. Zuweisungen möglich sind:

```
>>> issubclass(Lexicon,dict)
True
>>> wb = Lexicon()
>>> wb
{}
>>> wb = Lexicon({'a':'det','my':'det','some':'det'})
>>> wb
{'a': 'det', 'my': 'det', 'some': 'det'}
>>> isinstance(wb,dict)
True
>>> isinstance(wb,Lexicon)
True
>>> wb = Lexicon(pairlist('boy girl dog','noun noun noun'))
>>> wb
{'boy': 'noun', 'girl': 'noun', 'dog': 'noun'}
```

**Aufgabe 3: `merge()`**

Definieren Sie eine Methode **`merge()`**, die als Argument das neue Lexikon nimmt und keine Einträge im bestehenden Lexikon überschreibt:

```
>>> wb
{'heavy': 'adj', 'light': 'adj'}
>>> wb1
{'light': 'noun', 'candle': 'noun'}
>>> wb.merge(wb1)
>>> wb
{'heavy': 'adj', 'light': 'adj', 'candle': 'noun'}
```

**Aufgabe 4: `from_wordlist()`**

Definieren Sie eine Methode **`from_wordlist()`**, die zwei Argumente hat: eine Liste von Wortformen derselben Kategorie und, als 2. Argument, die Kategorie. **`from_wordlist()`** macht daraus wie im folgenden Screenshot Einträge für das Lexikon:

```
>>> wb
{'boy': 'noun', 'girl': 'noun', 'dog': 'noun'}
>>> wb.from_wordlist(['the','my','a'],'det')
>>> wb
{'a': 'det', 'boy': 'noun', 'dog': 'noun', 'girl': 'noun', 'my': 'det', 'the': 'det'}
```

Benutzen Sie hierbei die **`dict`**-Methode **`fromkeys()`**.

**Aufgabe 5: `get_cat()`**

Definieren Sie eine Methode **`get_cat()`**, die als Argument eine Wortform hat und dazu die Kategorie liefert. Falls noch kein Eintrag im Lexikon vorhanden ist, wird auf dem Wege der Benutzerintervention ein neuer Eintrag gemacht.

```
>>> wb
{'a': 'det', 'the': 'det', 'his': 'det', 'my': 'det'}
>>> wb.get_cat('a')
'det'
>>> wb.get_cat('her')
Das Wort 'her' ist nicht im Lexikon.
Wie lautet die Kategorie?
det
'det'
>>> wb
{'a': 'det', 'the': 'det', 'his': 'det', 'my': 'det', 'her': 'det'}
```

**Aufgabe 6: `get_cat_table()`**

Definieren Sie eine Methode **`get_cat_table()`**. Diese Methode nimmt kein Argument und liefert ein invertiertes Lexikon mit den Kategorien als *keys* und den Wortformen als *values* zurück. Da zu einer Kategorie mehrere Wortformen gehören können, werden letztere als Liste gespeichert, z.B. **`{'det':['the','a','an'], ...}`**

```
>>> wb
{'boy': 'noun', 'cried': 'verb', 'slept': 'verb', 'the': 'det', 'my': 'det', 'girl': 'noun'}
>>> wb.get_cat_table()
{'noun': ['boy', 'girl'], 'verb': ['cried', 'slept'], 'det': ['the', 'my']}
```

**Aufgabe 7: `print_table()`**

Definieren Sie eine Methode **`print_table()`**. Diese Methode nimmt kein Argument und druckt das Lexikon nach Wortformen sortiert alphabetisch in einer zweispaltigen Tabelle.

```
>>> wb.print_table()
a       det
boy     noun
in      prep
my      det
sun     noun
the     det
```

**Optional:**

Verwenden Sie für die Anhübschung der Ausgabe die Methode **`ljust()`**, deren Argument die Länge des längsten Wortes im Lexikon ist. Dafür müssen Sie eine ungebundene Funktion **`longest()`** definieren, die die Länge des längsten *keys* im Dictionary als **`int`** ausgibt:

```
>>> wb.keys()
['a', 'and', 'hands', 'hand', 'an']
>>> longest(wb)
5
```

Dieses wird dann bei der Verwendung von **`ljust()`** als dessen Argument übergeben

**Aufgabe 8: `print_cat_table()`**

Definieren Sie eine Methode **`print_cat_table()`**. Diese erzeugt einen als Tabelle formatierten Ausdruck eines mit der Methode **`get_tag_table()`** erstellten invertierten Lexikons.

```
>>> wb.print_cat_table()
det   ['a', 'the', 'my']
noun  ['boy', 'sun', 'cat']
prep  ['on', 'under', 'in']
```

### Aufgabe 9: `statistics()`

Definieren Sie eine Methode **`statistics()`**. Diese hat kein Argument und gibt als statistisches Datum die Worthäufigkeit nach Kategorie aus.

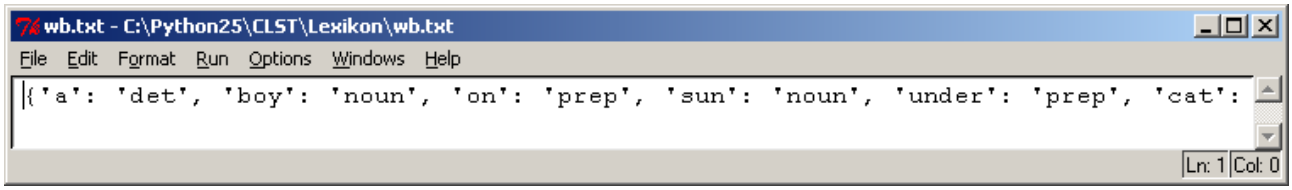
```
>>> wb.statistics()
det     => 3
noun    => 3
prep    => 3
```

### Aufgabe 10: `save()`

Definieren Sie eine Methode **`save()`**. Diese hat als Argument einen Dateinamen, unter dem das Lexikon in einer externen Datei gespeichert wird.

```
>>> wb.save('wb.txt')
```

⇓




### Aufgabe 11: `load()`

Definieren Sie eine Methode **`load()`**. Diese hat als Argument den Namen der Datei, aus der das Lexikon eingelesen werden soll.

```
>>> wbneu = Lexicon(wb.load('wb.txt'))
>>> wbneu.print_table()
a       det
boy     noun
cat     noun
on      prep
sun     noun
under   prep
```