Uge 4

Tirsdag

Objects

```
creamCan.spray()

content = “cream”
w = s.upper()
```

Objects

```
glueGun.applyGlue(fingers)

glue = "_"
fingers = ["thumb", "index finger"]
glue.join(fingers)

"".join(fingers)
```

Typer i Python

integers: 1321

floats: 3.2324

booleans: True, False

strings: “hello world”

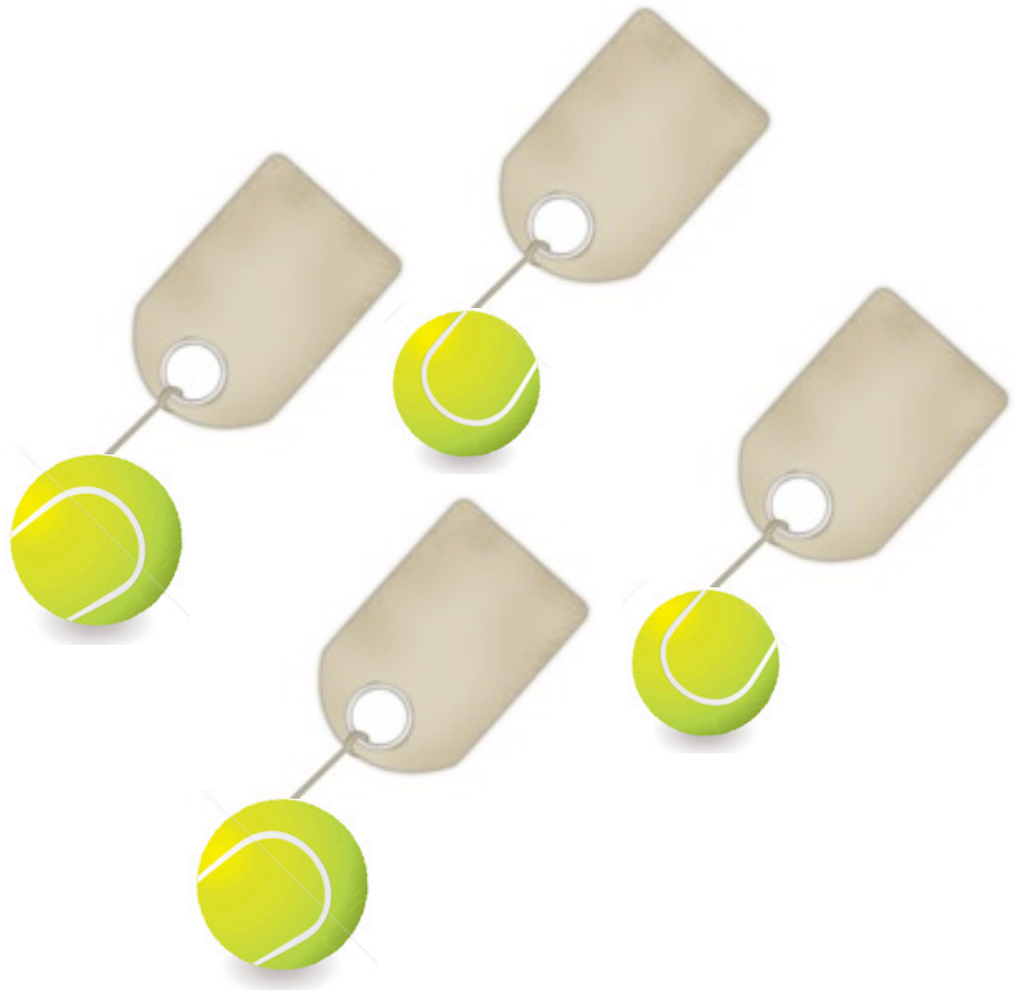
lists: [value, value, value]

dictionaries: { key: value, key: value }

Dictionaries

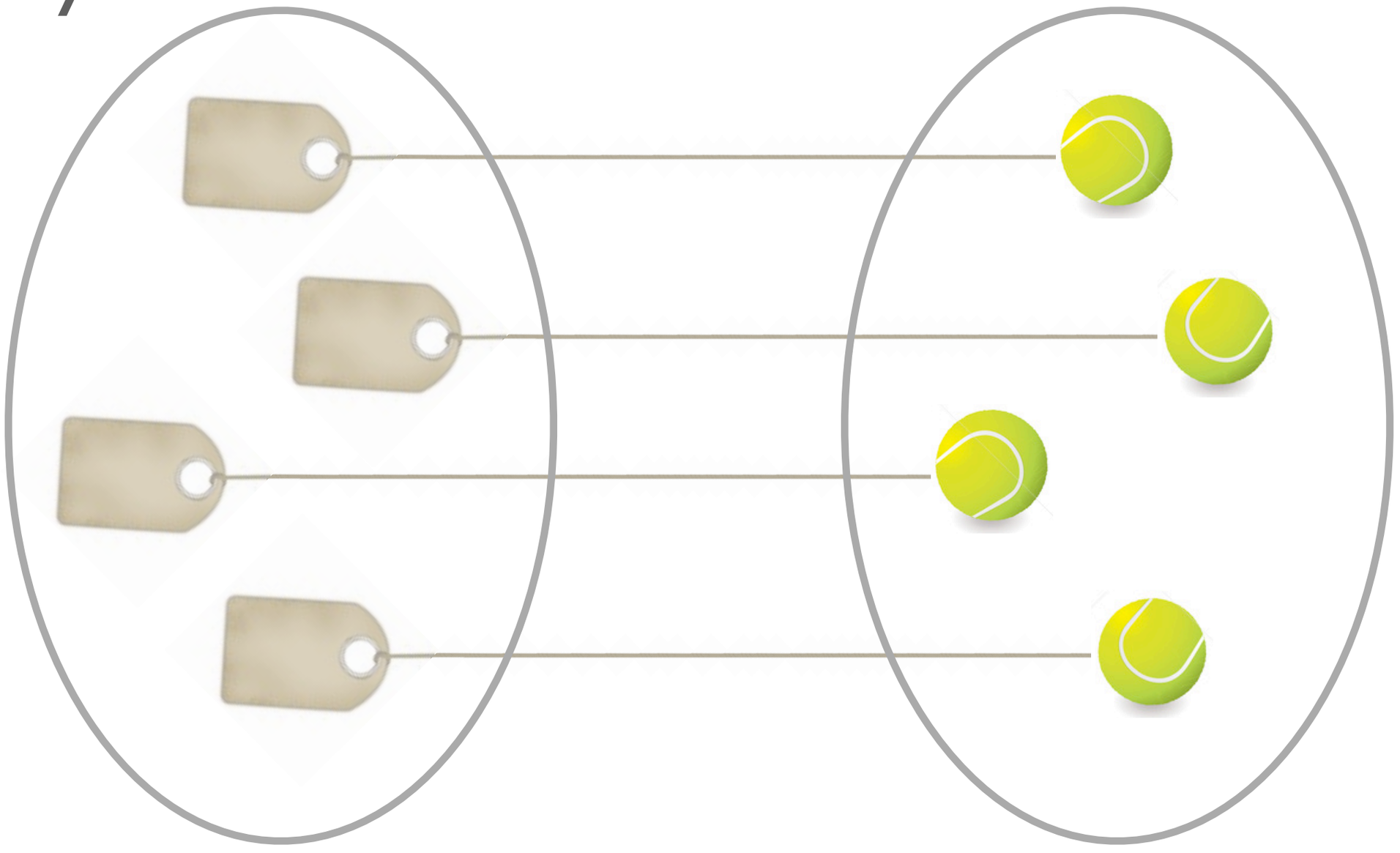
Liste

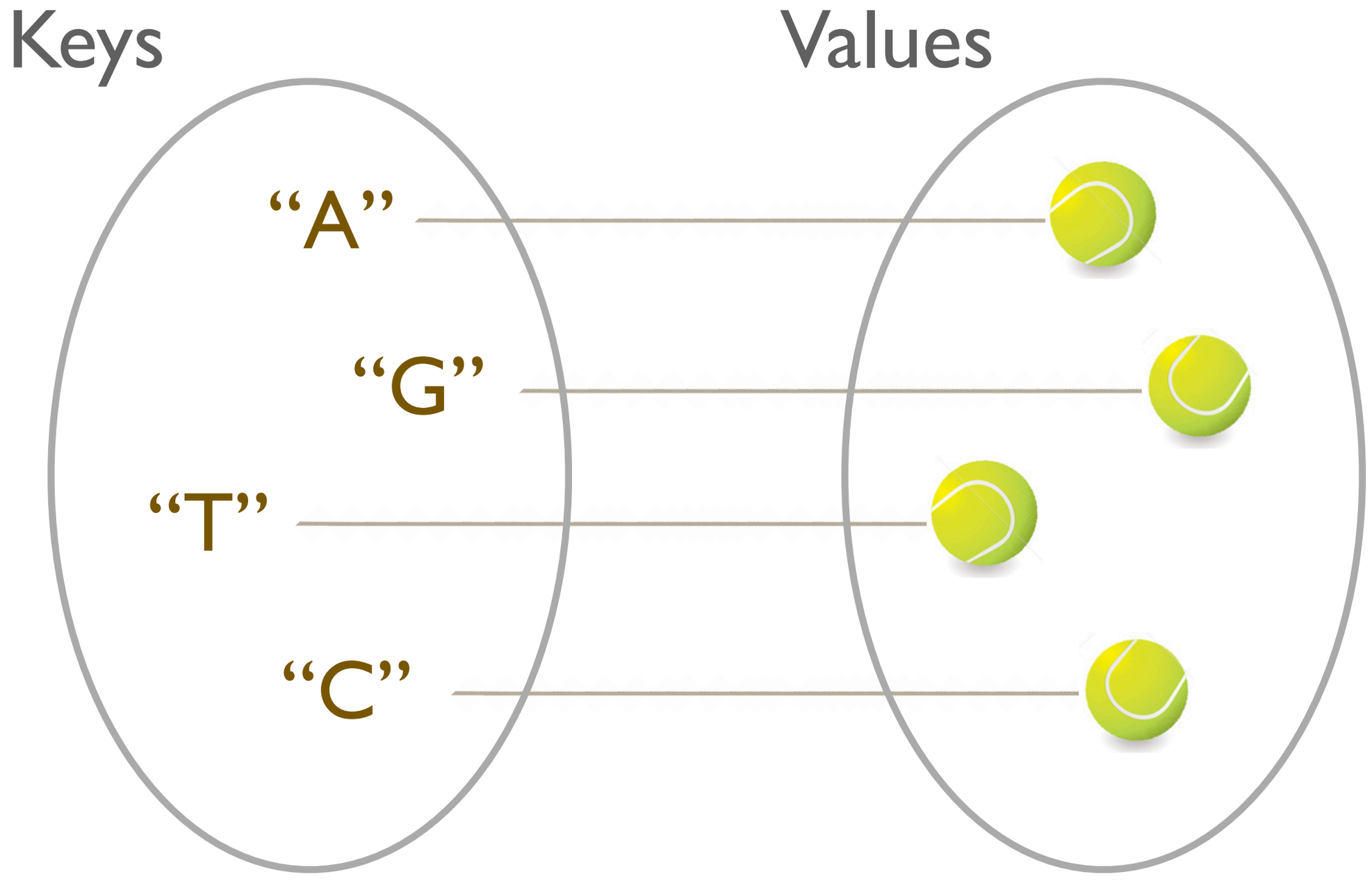
Dictionary

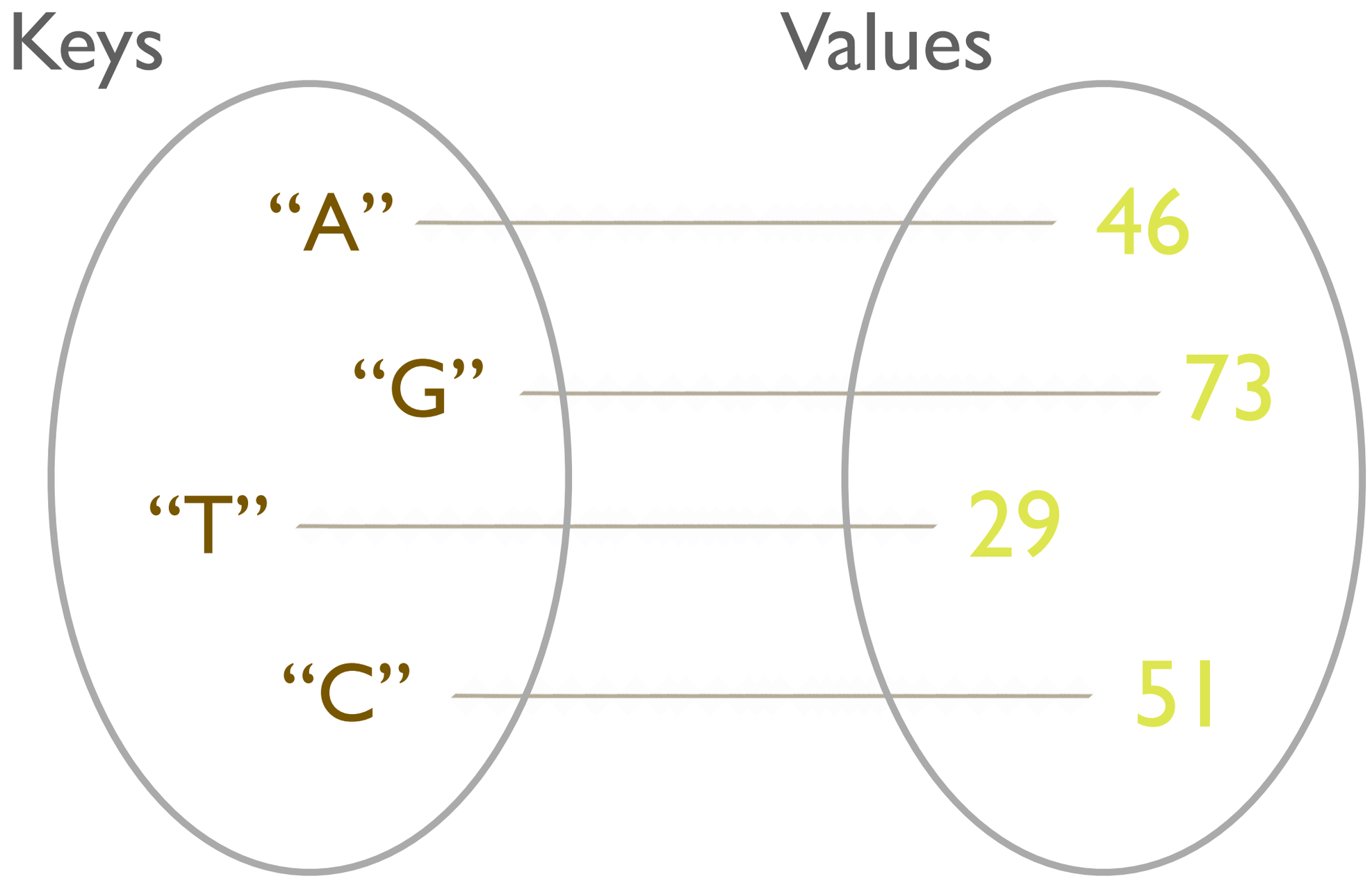


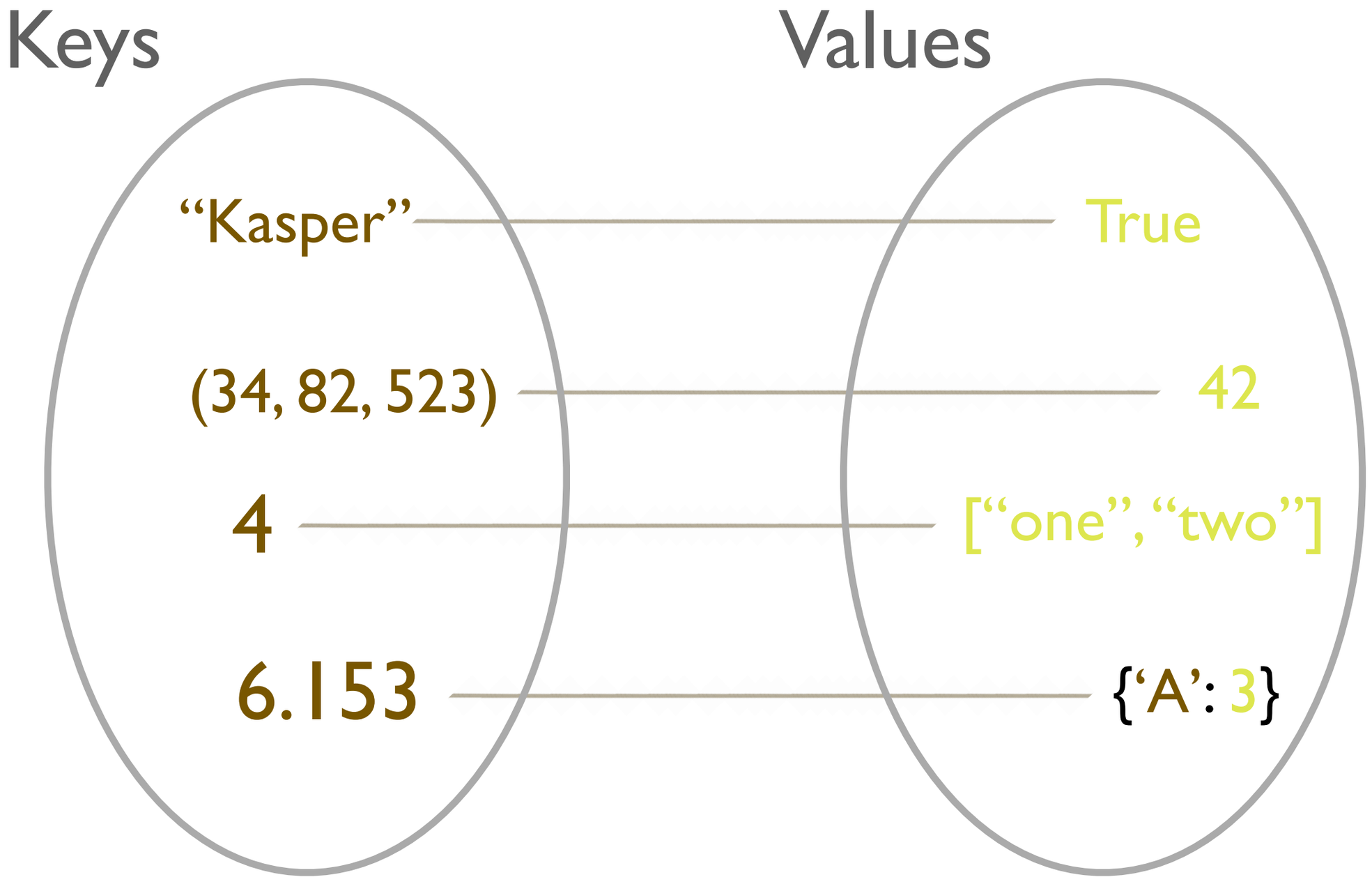
Keys

Values




Keys
Values
“A”
“G”
“T”
“C”


Keys
Values
“A”
46
“G”
73
“T”
29
“C”
51


Keys
Values
“Kasper”
True
(34, 82, 523)
42
4
[“one”, “two”]
6.153
{‘A’: 3}

Dictionaries versus lists

Label (key) i stedet for index

Direkte adgang til værdi vha. label - Hvad er alternativet?

Hvad er det så kan kan med dictionaries?

Definition af dictionaries

```
# new empty dictionary:
d = {}
d = dict()

# new non-empty dictionary:
d = {‘bird’: ‘flying animal’,
     ‘mouse’: ‘small animal’}
```

Basale operationer

```
# new key value pair:
d[‘bird’] = ‘flying animal’

# new value for existing key:
d[‘bird’] = ‘living dinosaurs’

# get value for a key:
definition = d[‘bird’]

# delete key-value pair:
del d[‘bird’]
```

Vigtige dictionary metoder

d.keys() returnerer en liste af keys

d.values() returnerer en liste af values

d.items() returnerer en liste af (key, value) tupler

d.has_key(“bird”) tester om d har en key “bird”

Opgave

Lav et for loop der tæller antallet af de forskellige bogstaver i en streng.

Eksempel: ‘AGTACCGATACATAGCC’

{ ’A’: 6, ‘G’: 3, ‘T’: 3, ‘C’: 5 }

Løsning

```
dna = “AGTACCGATACATAGCC”

counts = {‘A’:0, ‘G’:0, ‘T’:0, ‘C’:0}

for b in dna:
    counts[b] += 1


for b, c in counts.items():
    print b, c
```

Opgave

Lav et for loop der tæller antallet af de forskellige bogstaver i en streng.

Hvis du ikke ved noget om hvad der er i strengen!

Eksempel: ‘AGTACCGATACATAGCC’

{ ’A’: 6, ‘G’: 3, ‘T’: 3, ‘C’: 5 }

Optællinger af ukendte kategorier

```
dna = “AGTACCGATACATAGCC”

counts = {}
for b in dna:
    if not counts.has_key(b):
        counts[b] = 0
    counts[b] += 1


print counts
for b, c in counts.items():
    print b, c
```

“in” operatoren (like .has_key())

```
d = {“Stan”: “Getz”,
     “Britney”: “Spears”}

firstName = “Stan”

if firstName in d:
    print d[firstName]

for k in d:
    print x, d[x]

for v in k.values():
    print v
```

Opgave

Hvordan kan man bruge dictionaries til at finde antallet af forskellige elementer i en liste?

F.eks.:

```
[3,2,3,8,4,8,4,3]
```

Løsning på opgave

```
lst = [4,3,3,4,6,3,6]

uniqueNumbers = {}
for number in lst:
    uniqueNumbers[number] = None
print uniqueNumbers.keys()
```

Gennemløb af keys - sorteret

```
d = {‘A’:1, ‘B’:2, ‘C’:3, ‘D’:4}

print d.keys()

keyList = d.keys()
keyList.sort()
print keyList
```

Filer

Åbne en fil - fil objektet

```
f = open('workfile.txt', 'r')

f = open('workfile.txt', 'w')
```

Skrive til en fil

```
f = open('workfile.txt', 'w')

f.write(“First line\n”)
f.write(“Second line\n”)

# eller
f.write(“First line\nSecond line\n”)

f.close()
```

Læse fra en fil

```
f = open('workfile.txt', 'r')

print f.read()

print f.read()

f.close()
```

```
print f.readline()

print f.readline()
```

```
print f.readlines()
['First line\n', 'Second line\n']
```

Læse og skrive en fil linie for linie

```
inputFile = open(“workfile”, ‘r’)
outputFile = open(“outputfile”, ‘w’)

for line in inputFile:
    line = line.upper()
    outputFile.write(line)
```

Læse fra en lukket fil :-(

```
f = open(“workfile”, ‘r’)
f.close()
f.read()
```

Uge 5

Torsdag

Dagens tekst

Næste uges aflevering

For loops

Precedence - “rangfølge”

Referencer

Næste uges aflevering - Fasta parsing

```
>sequenceOne some description
AGTACACCAGTAATGACAGATATTTGCCGTAAGCATGACCAGACGTT
ATGACAGATATTTGCCGTAAGCATGACCAGATGACAGATATTTGCCG
TAAGCATGACCAGATGACAGATATTTGCCGTAAGCATGACCAGATGA
CAGATATTTGCCGTAAGCATGACCAGATGACAGATATTTGCCGTAAG
CATGACCAGATGACAGATATTTGCCGTAAGCATGACCAGAGTACCCA
TGAATGCGG
>sequenceTwo some description
AGTACACCAGTAATGACAGATATTTGCCGTAAGCATGACCAGACGTT
ATGACAGATATTTGCCGTAAGCATGACCAGATGACAGATATTTGCCG
TAAGCATGACCAGATGACAGATATTTGCCGTAAGCATGACCAGATGA
CAGATATTTGCCGTAAGCATGACCAGATGACAGATATTTGCCGTAAG
CATGACCAGATGACAGATATTTGCCGTAAGCATGACCAGAGTACCCA
TGAATGCGG
```

Variable i for loops

```
lst = [4, 2, 5]

for x in lst:
    x = 0
print lst

for i in range(len(lst)):
    lst[i] = 0
print lst
```

Precedence

`**`
`+x, -x`
`*, /, //, %`
`+, -`
`in, not in, is, is not, <, <=,!=, ==`
`not x`
`and`
`or`

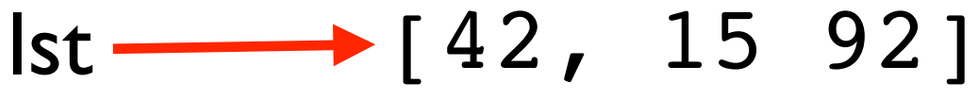
Precedence - invisible parentheses

```
-x * y**2 + z == p
((((-x) * (y**2)) + z) == p)

True and False or True
(True and False) or True

not True and False or True
(True and False) or True
```

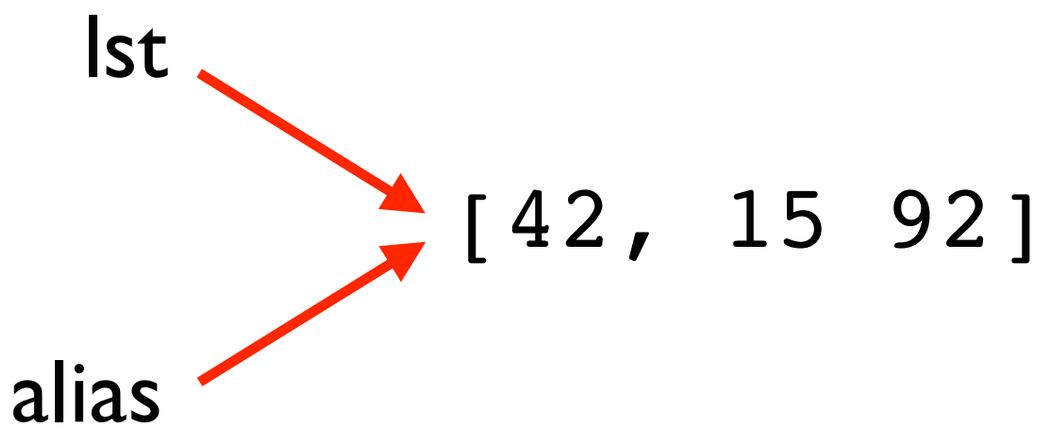
Referencer

En variabel kan kun indeholde en enkelt værdi og for sammensatte datatyper er den værdi en reference.

- tænk på det som en pil til de parenteser der indeholder listen



Kopierer man referencen har man to variabler der refererer til den samme ting - eller et alias.

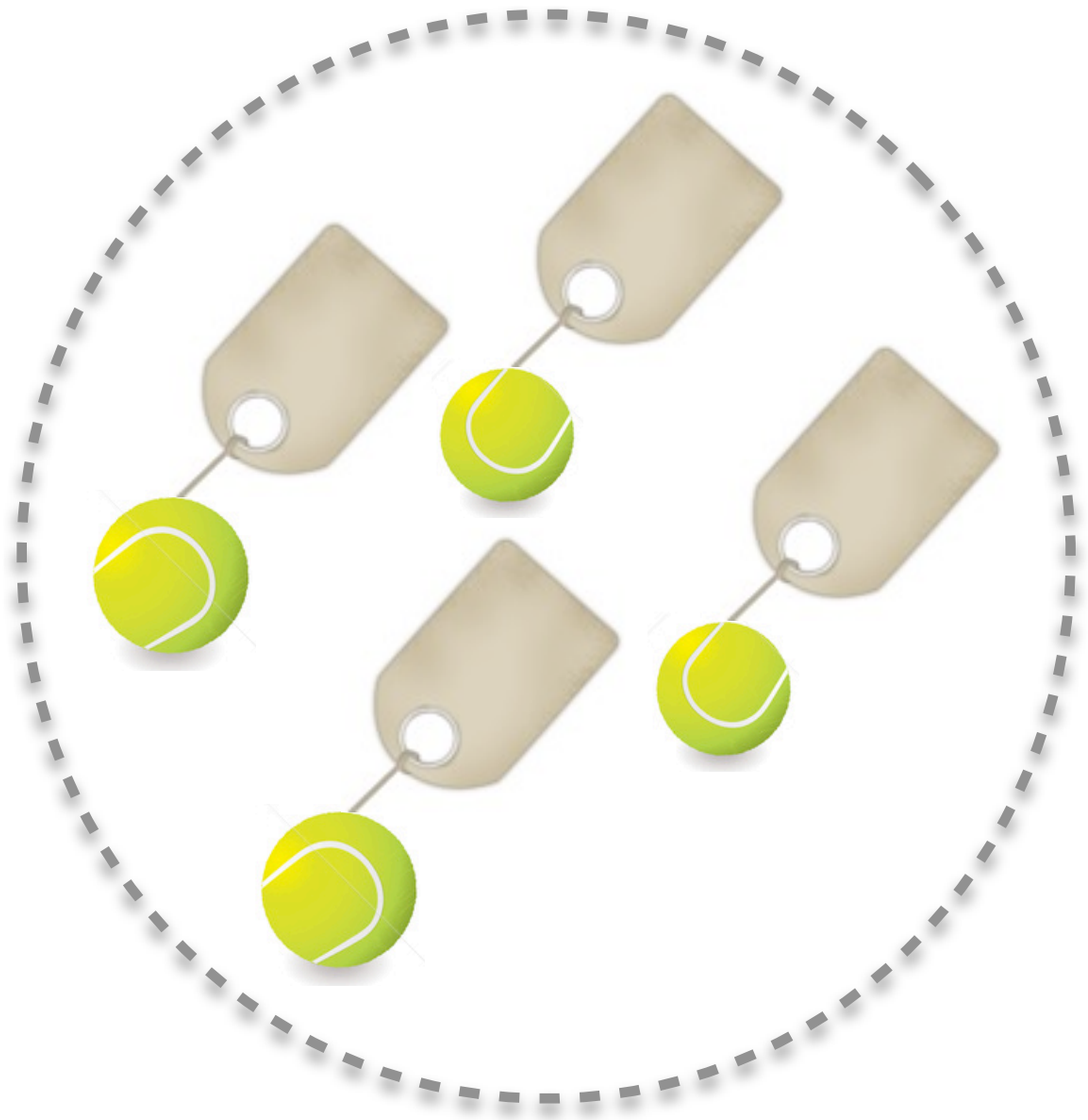


Liste variabel

```
lst = [“first”, “second”, “third”]

otherLst = lst

otherLst[1] = “tenth”

print lst
```

Dictionary variabel

```
D = {“one”:1, “two”:2}

otherD = D

otherD[“two”] = 3

print D[“two”]
```



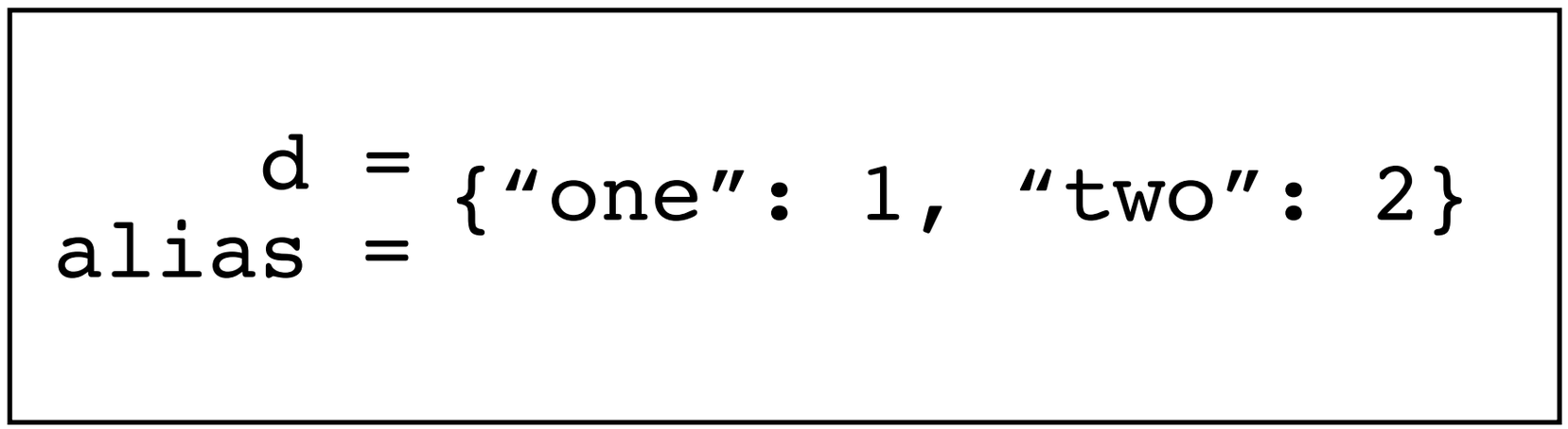
Aliasing

```
d = [42, 15 92]

alias = d
```

```
d =
[42, 15 92]
alias =
```

Aliasing

```
d = {“one”: 1, “two”: 2}

alias = d
```



Kloning af lister og dictionaries

```
lst = [1,2,3]
lstCopy = lst[0:len(lst)]
lstCopy = lst[:]
```

```
d = {“one”:1, “two”:2}
dCopy = d.copy()
```

Lister som parametre til funktioner

```
def modifyList(l):
    l[0] = “changed”
    return l


L = [“one”, “two”, “three”]

modifiedL = modifyList(L)

print L
print modifiedL
```

Opgave

Lav om på funktionen modifyList(l), så den ikke ændrer på input listen.

Løsning på opgave

```
def modifyList(l):
    clone = l[:]
    clone[0] = “changed”
    return clone

L = [“one”, “two”, “three”]

modifiedL = modifyList(L)

print L
print modifiedL
```