

Inleiding Programmeren

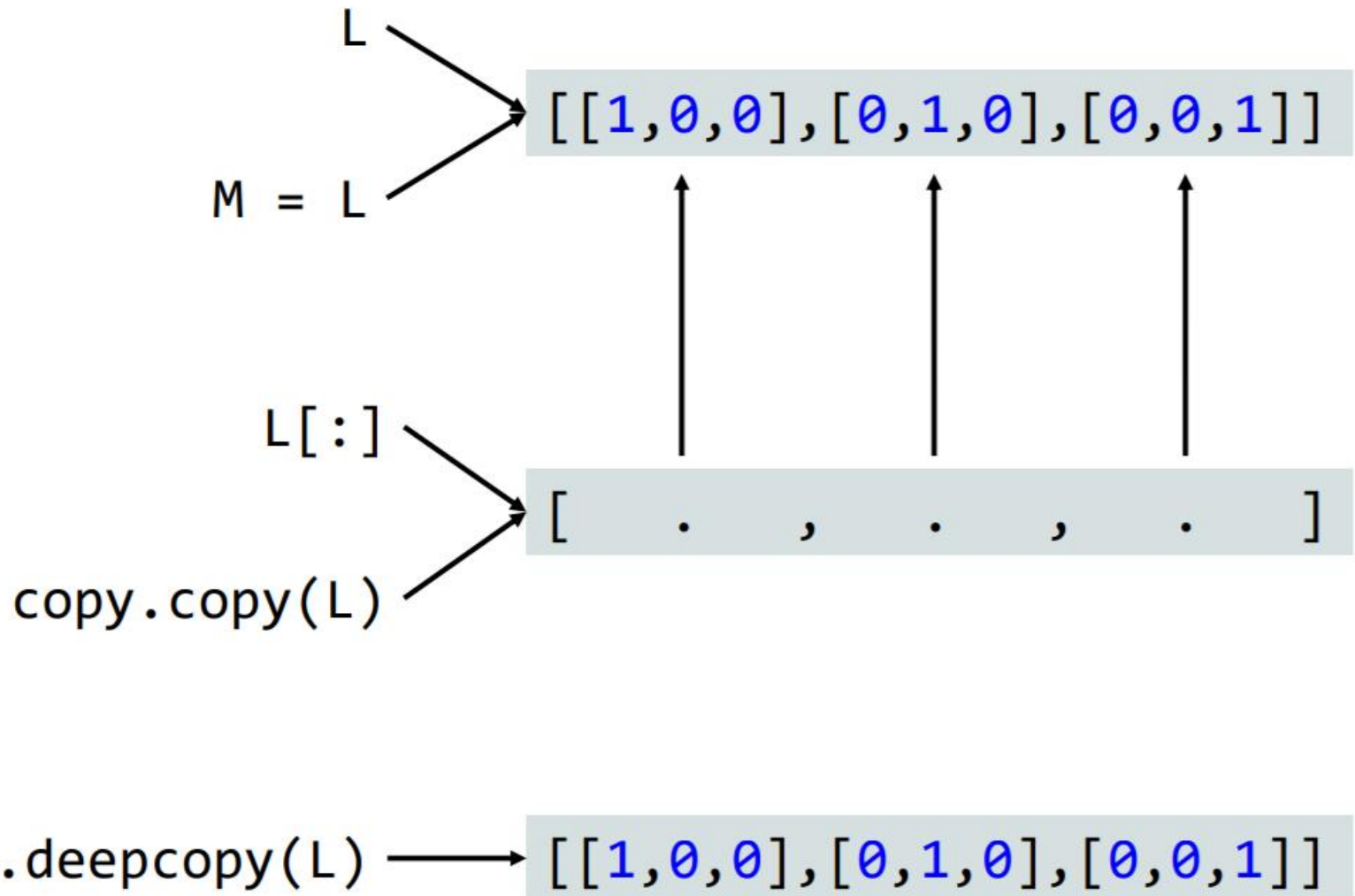
Toon Calders

Les 4: dict, set, tuple, files

Wat zagen we vorige keer?

- Mutable en immutable data types
 - Lijst objecten kunnen *wijzigen*
- Lijsten als parameter
- Strings zijn immutable
- Copy en deepcopy

Verschillende vormen van copy:

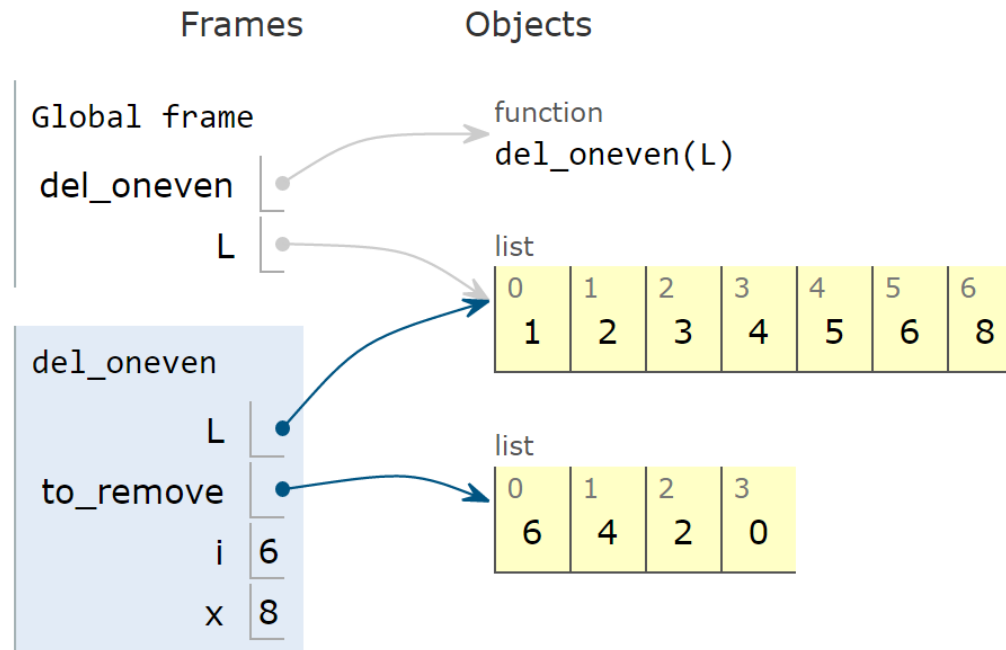


Lijst Parameters

```
def del_oneven(L):  
    to_remove=[]  
    for i in range(len(L)):  
        x=L[i]  
        if x%2==1:  
            to_remove=[i]+to_remove  
    for i in to_remove:  
        del(L[i])
```

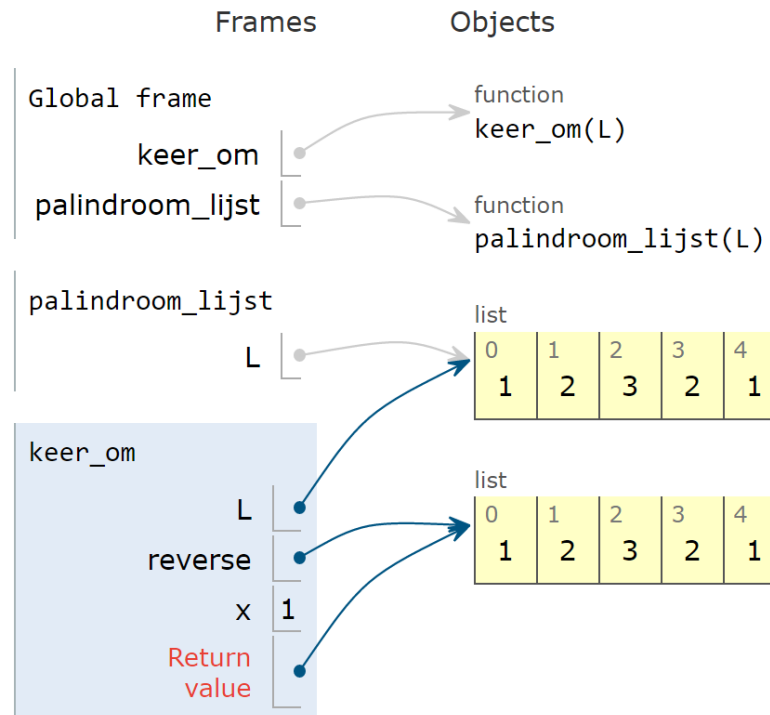
```
L=[1,2,3,4,5,6,7,8]  
del_oneven(L)  
print(L)
```

```
[2, 4, 6, 8]
```



Lijst parameters

```
def keer_om(L):  
    reverse=[]  
    for x in L:  
        reverse=[x]+reverse  
  
    return reverse  
  
def palindroom_lijst(L):  
    if keer_om(L)==L:  
        return True  
    return False  
  
print(palindroom_lijst([1,2,3,2,1]))
```



Vlugge vraag: wat loopt er hier mis?

```
def priem(x):
    for i in range(2, x):
        if x%i==0:
            return False
    return True
```

```
def verwijder_niet_priem(L):
    for x in L:
        print(x, "wordt getest")
        if not priem(x):
            L.remove(x)
            print(x, "is niet priem")
```

```
priemen=list(range(10))
verwijder_niet_priem(priemen)
print(priemen)
```

0 wordt getest
1 wordt getest
2 wordt getest
3 wordt getest
4 wordt getest
4 is niet priem
6 wordt getest
6 is niet priem
8 wordt getest
8 is niet priem
[0, 1, 2, 3, 5, 7, 9]

Handwritten annotations in red:
- Question marks and arrows pointing to the sequence of operations for x=4 and x=6.
- The number 9 is circled, with an arrow pointing to it from the list output.
- A large red '2' is written at the bottom right.


```
### Juist 1:
def verwijder_niet_priem(L):
    for x in L[:]:
        print(x, "wordt getest")
        if not priem(x):
            L.remove(x)
            print(x, "is niet priem")

### Juist 2:
def verwijder_niet_priem(L):
    to_remove=[]
    for x in L:
        print(x, "wordt getest")
        if not priem(x):
            to_remove.append(x)
            print(x, "is niet priem")
    for x in to_remove:
        L.remove(x)
```

Juist 1:

```
def verwijder_niet_priem(L):  
    for x in L[:]:  
        print(x, "wordt getest")  
        if not priem(x):  
            L.remove(x)  
            print(x, "is niet priem")
```

Handwritten annotations:

- A red circle around `L[:]` with an arrow pointing to the text "kopij van L".
- A red circle around `L.remove(x)` with a red "X" next to it.

Juist 2:

```
def verwijder_niet_priem(L):  
    to_remove=[]  
    for x in L:  
        print(x, "wordt getest")  
        if not priem(x):  
            to_remove.append(x)  
            print(x, "is niet priem")  
    for x in to_remove:  
        L.remove(x)
```

Handwritten annotations:

- A red circle around `to_remove=[]`.
- A red circle around `to_remove.append(x)`.
- A red circle around `to_remove` in the second loop.
- A red circle around `L.remove(x)` with a red "X" next to it.

Vlugge vraag: wat is het resultaat?

```
def verdubbel(L):
    """ Verwacht een lijst getallen en verdubbelt
    elk getal """
    for i in range(len(L)):
        L[i]=2*L[i]
```

```
def twee_maal(L):
    """ Doel: maal L gelijk aan L+L. """
    L=L+L
```

```
L=[1,2,3]
verdubbel(L)
twee_maal(L)
print(L)
```

- (A)[1,2,3]
- (B)[2,4,6]
- (C)[1,2,3,1,2,3]
- (D)[2,4,6,2,4,6]



Lijsten als parameters

```
def verdubbel(L):  
    """ Verwacht een lijst getallen en verdubbelt  
    elk getal """  
    for i in range(len(L)):  
        L[i]=2*L[i]
```



```
def twee_maal(L):  
    """ Doel: maal L gelijk aan L+L. """  
    for x in L[:]:  
        L.append(x)
```

```
L=[1,2,3]  
verdubbel(L)  
twee_maal(L)  
print(L)
```

Vlugge vraag

- Wat doet de volgende functie?

```
def bubble(L):  
    swap=True  
    while swap:  
        swap=False  
        for i in range(len(L)-1):  
            if L[i]>L[i+1]:  
                L[i], L[i + 1] = L[i + 1], L[i] # wissel L[i] en L[i+1]  
                swap=True
```

Opmerking: lijst sorteren

- Sorteert een lijst L:
 - `L.sort()`
- Geef gesorteerde versie van L, maar laat L zelf ongemoeid:
 - `sorted_L = sorted(L)`
- We kunnen een key en reverse geven:
 - `key = functie`
 - `reverse = True` of `= False` ; by default `reverse == False`

Voorbeeld : lijst sorteren

```
L=["Michael","Jim","Dwight","Pam","Ryan","Stanley","Kevin"  
  ,"Meredith","Angela","Oscar","Phyllis","Toby","Kelly"]
```

```
L.sort(reverse=True)
```

```
print(sorted(L))
```

```
print(L)
```

```
M=[[1,2,3],[2,0,1],[3,5,7]]
```

```
M.sort()
```

```
print(M)
```

```
['Angela', 'Dwight', 'Jim', 'Kelly', 'Kevin', 'Meredith', 'Michael', 'Oscar', 'Pam', ...  
['Toby', 'Stanley', 'Ryan', 'Phyllis', 'Pam', 'Oscar', 'Michael', 'Meredith', 'Kevin', ...  
[[1, 2, 3], [2, 0, 1], [3, 5, 7]]
```

Voorbeeld : lijst sorteren

```
M=[[1,2,3],[2,0,1],[3,5,7]]
```

```
def rowsum(r):  
    som=0  
    for x in r:  
        som += x  
    return som
```

```
print(sorted(M,key=rowsum))
```

```
[[2, 0, 1], [1, 2, 3], [3, 5, 7]]
```


Opmerking: strings

- strings zijn in python immutable
 - ~~s.append("abc")~~
- We moeten expliciet een nieuwe string maken:
 - `s = s+"abc"`
- Strings zijn *itereerbaar* ; we kunnen dus een for-loop gebruiken om over de karakters te lopen

Voorbeeld: strings

```
s="abcdadbcde"
```

```
a = 0
```

```
for letter in s:
```

```
    if letter == "a":
```

```
        a += 1
```

```
print("De string bevat",a,"maal de letter a")
```

String functies

- Er bestaan heel wat handige stringfuncties:
 - `s.trim()` : verwijder witruimte
 - `s.split()` : splits de string
 - `s.upper()` : maak upper case
 - ...
- Merk op: strings zijn immutable; al deze functies geven een nieuwe string als resultaat maar laten de string waarop ze worden aangeroepen ongewijzigd

Oefening

- Gegeven:
 - Lijst met woorden woordenBoek
 - Patroon P dat bestaat uit letters en wildcards (“?”)
- Geef alle woorden uit woordenBoek die je kan maken door de wildcards te vervangen door letters
- “aap” voldoet aan “?ap” maar niet aan “b??” of “a?ap”

```
from words import woordenBoek  
print(match("p??h??", woordenBoek))
```

Andere Collectie Types

- Collectie type: type dat verschillende waarden kan bevatten
- Enkele erg handige andere data structuren
 - dict
 - tuple
 - set

set

- Een verzameling
 - Elk element komt maximaal 1 maal voor
 - Volgorde is niet belangrijk (**en wordt ook niet bijgehouden!**)
- “*x in set*” is veel efficiënter dan “*x in lijst*”
- Net zoals bij lijsten is een variabele van type set, eigenlijk een referentie naar een set

set: gebruik

```
s=set()          # nieuwe lege verzameling
s={ 1, 2, 3, 4 } # nieuwe verzameling met elementen 1, 2, 3, 4
s.add(x)          # voeg element x toe aan de verzameling
                  # doet niets als x al in de verzameling zit
s.remove(x)       # verwijder element x uit de verzameling
                  # Geeft een fout als x niet in de
                  # verzameling zit
s.discard(x)      # verwijder element x uit de verzameling
                  # Geeft geen fout als x niet in de
                  # verzameling zit
y=s.pop()         # verwijder een willekeurig element;
                  # dat element wordt toegekend aan y
```

set operaties

$s t$	# unie van de verzamelingen s en t
$s \& t$	# doorsnede van s en t
$s - t$	# alle elementen van s die niet in t zitten
$x \text{ in } s$	# waar als x een element is van de verzameling s
 for x in s :	# Druk alle elementen uit s af
$\text{print}(x)$	#in willekeurige volgorde

Set vergelijkingen

$s \leq t$ # s deelverzameling van t;

$s \subset t$ # s is een strikte deelverzameling

$s \geq t$ # omgekeerde vergelijkingen

$s > t$

$s==t$ # s en t hebben “dezelfde” elementen

Set: Voorbeeld

```
def all_valid(L,valid):  
    """ alle elementen uit lijst L zitten in lijst valid """  
    for x in L:  
        if not x in valid:  
            return False  
  
    return True  
  
def all_valid(L,valid):  
    """ alle elementen uit lijst L zitten in lijst valid """  
    return set(L)<=set(valid)
```

Set: aantal unieke elementen lijst

```
L=[1,2,5,5,7,8,9,10,3,4,5,6]
```

```
aantal = 0
for i in range(len(L)):
    first = True
    for j in range(i):
        if L[i] == L[j]:
            first = False
            break
    if first:
        aantal += 1
```

```
aantal = 0
for i in
range(len(L)):
    if L[i] not in L[:i]:
        aantal += 1
```

```
aantal = len(set(L))
```

dict

- Een “dictionary” of een woordenboek is een datastructuur die een sleutel (key) met een waarde (value) verbindt
- Een dictionary is een uitbreiding van een lijst, waarbij de indices niet noodzakelijk opeenvolgende getallen hoeven te zijn

dict: gebruik

<code>d=dict()</code>	<code># d is een nieuwe, lege dictionary</code>
<code>d[key]=value</code>	<code># associeer value met key</code> <code># als key al bestaat, wordt value</code> <code># overschreven</code>
<code>d[key]</code>	<code># de value geassocieerd met key</code> <code># geeft een key-error als die niet bestaat</code>
<code>key in d</code>	<code># true als er een (key,value)-paar bestaat</code>
<code>for key in d:</code>	<code># print alle key-value paren uit d af</code>
<code> print(key,d[key])</code>	

dict: voorbeeld

```
tekst="to be or not to be that is the question"
```

```
d=dict()
for token in tekst.split():
    if token in d:
        d[token] += 1
    else:
        d[token] = 1

print(d)
```

```
{'to': 2, 'be': 2, 'or': 1, 'not': 1, 'that': 1, 'is': 1, 'the': 1, 'question': 1}
```

tuple

- Een statische lijst; eens aangemaakt kunnen geen elementen toegevoegd of gewijzigd worden
- Handig om meerdere waarden terug te geven in een functie
- Gebruik:

```
t=(1,2,"abc")
```

```
# nieuw tuple met 3 waarden
```

```
print(t[1])
```

```
# print de tweede waarde af
```

```
s=t+(3,4)
```

```
# maak een nieuw tuple door 3 en 4 aan t toe  
te voegen
```

```
>>> t=(1,2,3,[],"abc")
```

```
>>> t[0]
```

```
1
```

```
>>> t[3]
```

```
[]
```

```
>>> t[2]=7
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#159>", line 1, in <module>
```

```
    t[2]=7
```

```
TypeError: 'tuple' object does not support item assignment
```


Merk op

- Net zoals list zijn ook dict en set mutable
- Tuple en string zijn niet mutable
- We kunnen een set, dictionary, string, tuple niet sorteren
- Wel een gesorteerde lijst van maken met sorted

Voorbeeld: sorteren

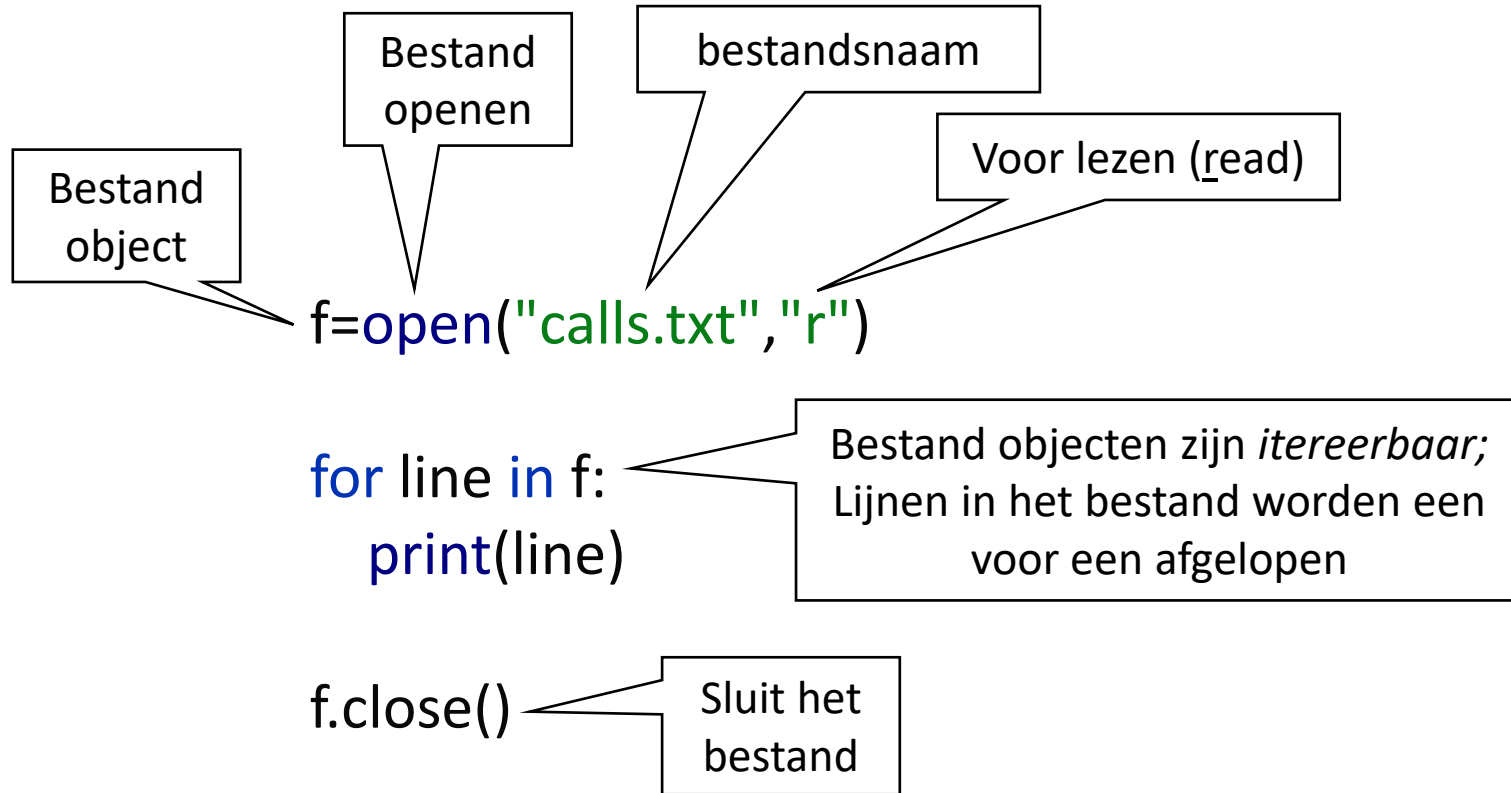
```
tekst="to be or not to be that is the question"  
print(sorted(tekst))  
print(sorted(set(tekst)))  
d={ "c":5, "d":3, "a":5, "b":3 }  
for key in sorted(d):  
    print(key,d[key])
```

```
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', 'a', 'b', 'b', ...  
[' ', 'a', 'b', 'e', 'h', 'i', 'n', 'o', 'q', 'r', 's', 't', 'u']  
a 5  
b 3  
c 5  
d 3
```

Oefening: Analyse gesprekken

- Bestand met paren van namen:
 - Beller
 - Opgebelde
- Enkele vragen die we gaan beantwoorden:
 - Wie belde het vaakst?
 - Naar wie werd het vaakst gebeld?
 - Welke personen belden het vaakst met elkaar?

Bestand lezen



Bestand schrijven

```
f=open("calls.txt","r")  
f2=open("calls2.txt","w")
```

Voor schrijven (write)

```
for line in f:  
    f2.write(line)
```

```
f.close()  
f2.close()
```

Schrijf weg naar een bestand

Ingelezen lijn interpreteren

```
f=open("calls.txt","r")
```

De meeste lijnen eindigen met een newline karakter; enkel de allerlaatste lijn in het bestand mogelijk niet

```
for line in f:
```

```
    line=line.strip()
```

```
    record=line.split()
```

```
    print((record[0],record[1]),
```

Dat newline karakter halen we weg

string.split() splitst een string op basis van witruimte (spatie, tab) en geeft een lijst van strings terug

```
f.close()
```

We gaan ervan uit dat er twee strings gescheiden door witruimte per lijn staan; we maken een tuple met deze twee strings

Uitvoer:

Jawad Isabelle
Omar Dayenne
Thibault Anass
Lune Stephen
Thibault Arthur
...

calls.txt

```
f=open("calls.txt","r")  
  
for line in f:  
    line=line.strip()  
    record=line.split()  
  
    print((record[0],record[1]))  
  
f.close()
```

('Jawad', 'Isabelle')
('Omar', 'Dayenne')
('Thibault', 'Anass')
('Lune', 'Stephen')
('Thibault', 'Arthur')

Vragen

- Hoeveel personen zijn er in totaal?
- Wat is het totale aantal gesprekken en wat is het aantal unieke gesprekken?

Vragen

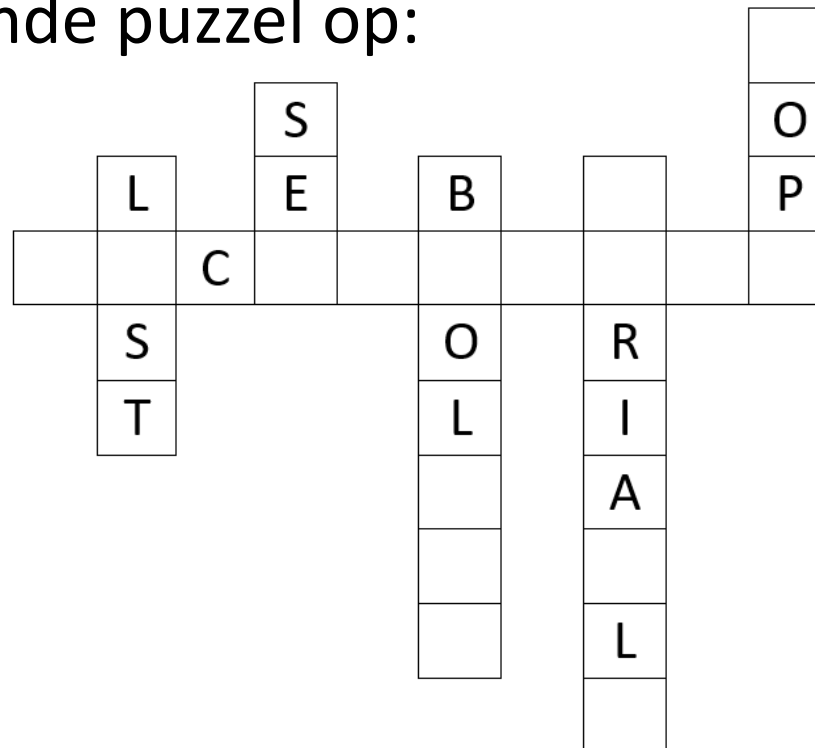
- Welke personen waren bij het grootste aantal gesprekken betrokken (als beller of als opgebelde) en bij hoeveel gesprekken is dat ?
- Welke personen hadden contact met het grootste aantal andere mensen?

Vragen

- Welke paren van mensen hadden het meeste contact met elkaar?

Uitdaging

- Los volgende puzzel op:



```
from words import woordenBoek
print(match("p??h??", woordenBoek))
```