

sets and dicts

ITGK TDT4109 uke 42

Litt info:

- Referansegruppe i går:
 - Teori avvik foiler vs øvinger: Det er ikke meningen at alt skal stå perfekt.
 - Hvis du trenger hjelp: Piazza og øvingsforelesning. Det er faktisk gode kilder! For mange bruker for lite slikt.
 - Jeg rekker ikke igjennom alt, alltid. Jeg dykker ned i ett eksempel, Terje viser koden til flere. Balansegang, uenighet, foilvideo bra i ettertid. Min live, Terje i ettertid? Innspill?
 - LF til øvinger ut raskere. Slakk, likhet for alle, planlegging verre med slakk?
 - Inspiraøving var mye! Meningen. Mer info, jepp. Definitivt.

Læringsmål og pensum

- Mål
 - Forstå prinsippene for, og kunne bruke i praksis
 - Mengder (sets)
 - Ordbøker (dictionary)
 - Vite forskjell på disse og lister, tupler og strenger
- Pensum
 - Starting out with Python, Chapter 9: Dictionaries and Sets
- Når dere går igjennom selv: Sjekk Terje sin fullstendige presentasjon og vedlagt kode. Han inkluderer en drøss ekstra kodeeksempler, men dette er det best at dere studerer selv. Synes jeg. ☺ Videoen hans er også enklere å lete etter spesifikke ting, da han følger presentasjon slavisk.

Datatyper

- Vi har hittil sett på:

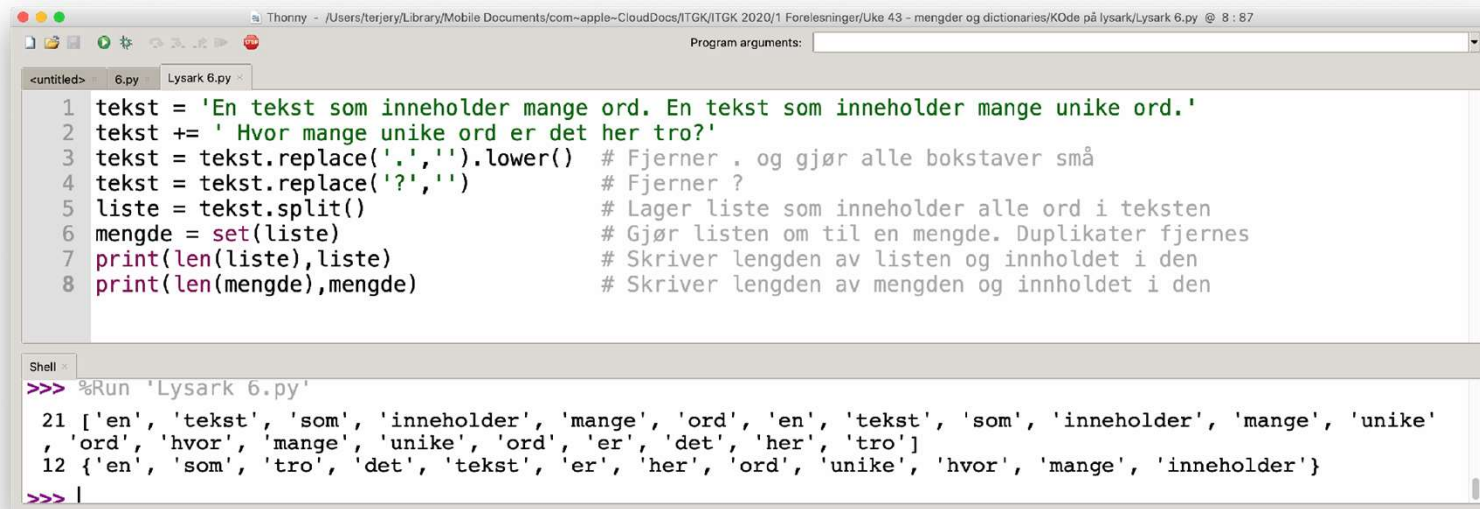
- Lister `[1,3,5,7]` `['a','e','i','o']` muterbare
- Tupler `(1,3,5,7)` `('a','e','i','o')` ikke muterbar
- Strenger `'1357'` `'aeio'` ikke muterbare
- For alle gjelder: Rekkefølge er viktig, man kan ha **duplikater**
- For alle gjelder: Vi kan iterere gjennom dem på samme måte, med eller uten indeksering: *for elem in list* vs *for i in range(len(list))*

Kapittel 9.2

Mengder (SETS)

Datastruktur for mengde (sets)

- Python har innebygd en datastruktur for å håndtere **mengder** (sets på engelsk).
- Hvert element i en mengde er unik, så du vil ikke få lagt til et element som finnes fra før.
 - Bruk dette f.eks. til å finne unike ord i en tekst?
- Børge: Jeg velger å se på teksten i filen lyric.py

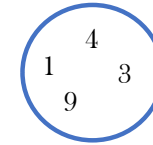


```
1 tekst = 'En tekst som inneholder mange ord. En tekst som inneholder mange unike ord.'
2 tekst += ' Hvor mange unike ord er det her tro?'
3 tekst = tekst.replace('.', '').lower() # Fjerner . og gjør alle bokstaver små
4 tekst = tekst.replace('?', '')         # Fjerner ?
5 liste = tekst.split()                  # Lager liste som inneholder alle ord i teksten
6 mengde = set(liste)                   # Gjør listen om til en mengde. Duplikater fjernes
7 print(len(liste), liste)               # Skriver lengden av listen og innholdet i den
8 print(len(mengde), mengde)            # Skriver lengden av mengden og innholdet i den
```

```
>>> %Run 'Lysark 6.py'
21 ['en', 'tekst', 'som', 'inneholder', 'mange', 'ord', 'en', 'tekst', 'som', 'inneholder', 'mange', 'unike',
   'ord', 'hvor', 'mange', 'unike', 'ord', 'er', 'det', 'her', 'tro']
12 {'en', 'som', 'tro', 'det', 'tekst', 'er', 'her', 'ord', 'unike', 'hvor', 'mange', 'inneholder'}
>>> |
```

Opprette og endre på mengder

- En mengde kan **lages** ved å tilordne en serie elementer inne i {}



```
D = {9,4,1,3} # Lager en mengde
```

- En kan **konvertere** lister og strenger ved å bruke funksjonen **set(x)**

```
A = set([5,3,3,12]) # NB: Mister  
duplikater!
```

```
B = set([True,'ost',23.2,92,False])
```

```
C = set('aaabcd') # gir mengden  
{ 'a', 'b', 'c', 'd' }
```

```
E = set() # gir en tom mengde
```

- Kan ikke referere til indeks `set[i]`
- Kan bruke element in set
- Legge til og fjerne elementer fra en mengde:

```
A.add(9) # Legg til ett element
```

```
A.update([3,4,2]) # Legg til flere  
elementer
```

```
B.remove(92)
```

Mengdeoperasjoner på A og B

- Union av to mengder:

`C = A.union(B)` # C blir union av A og B

- Snitt av to mengder:

`C = A.intersection(B)` # C blir snitt av A og B

- Forskjellen på to mengder:

`C = A.difference(B)` # Finnes i A, men ikke i B

- Symmetrisk forskjell på to mengder:

`C = A.symmetric_difference(B)` # Elementer som ikke deles av A og B

- Sjekke delmengde eller supermengde:

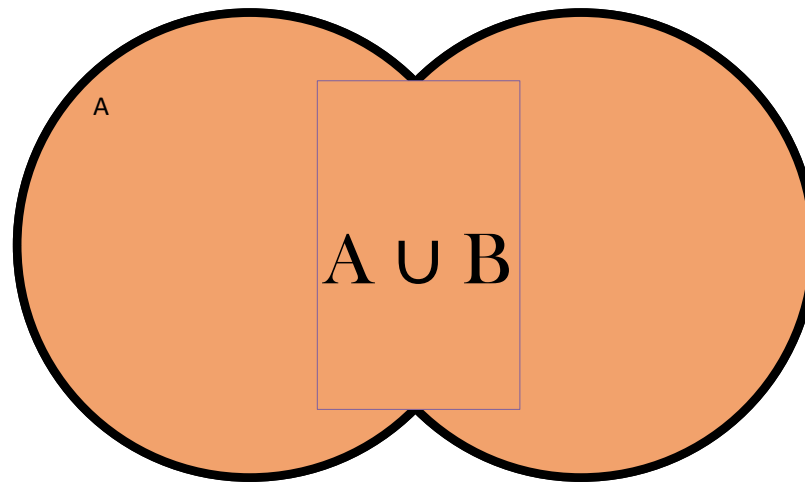
`A.issubset(B)` # Sjekker om A er delmengde av B

`A.issuperset(B)` # Sjekker om A er supermengde av B

Union av to mengder - Venn diagram

- $A \cup B$

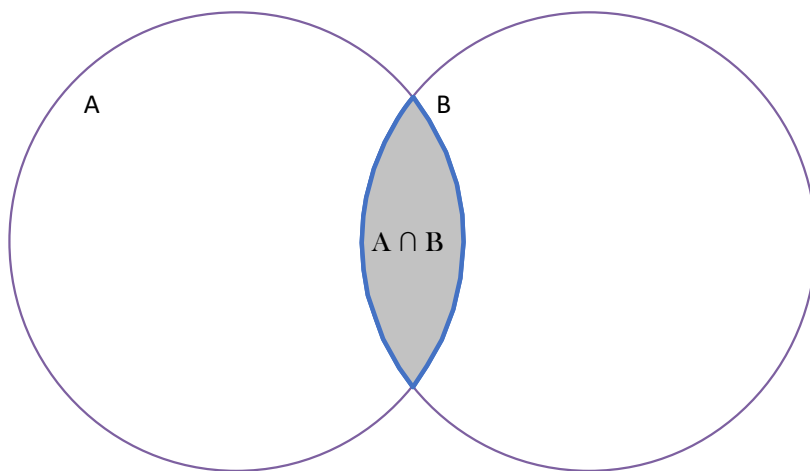
`C = A.union(B)` # C blir union av A og B



Snittet av to mengder - Venn diagram

- $A \cap B$

`C = A.intersection(B)` # C blir snitt av A og B



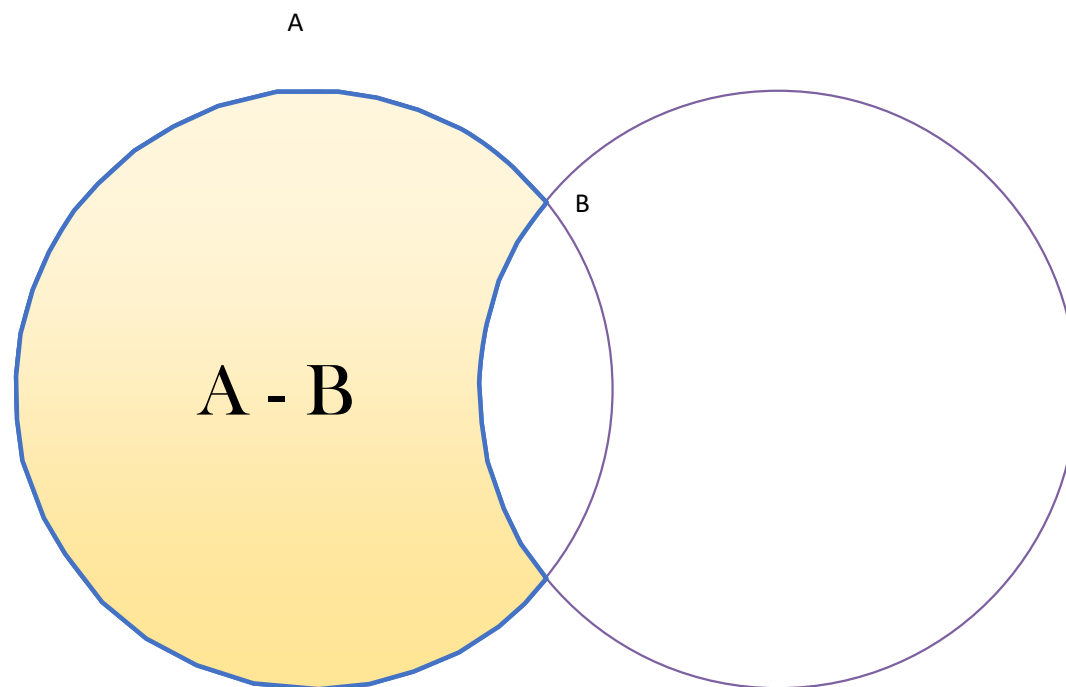
Det kan f.eks. utnyttes i lotto-oppgaven

Forskjell på mengde A og B

- Venn diagram

- $A - B$

$C = A.\text{difference}(B)$ # Finnes i A, men ikke i B

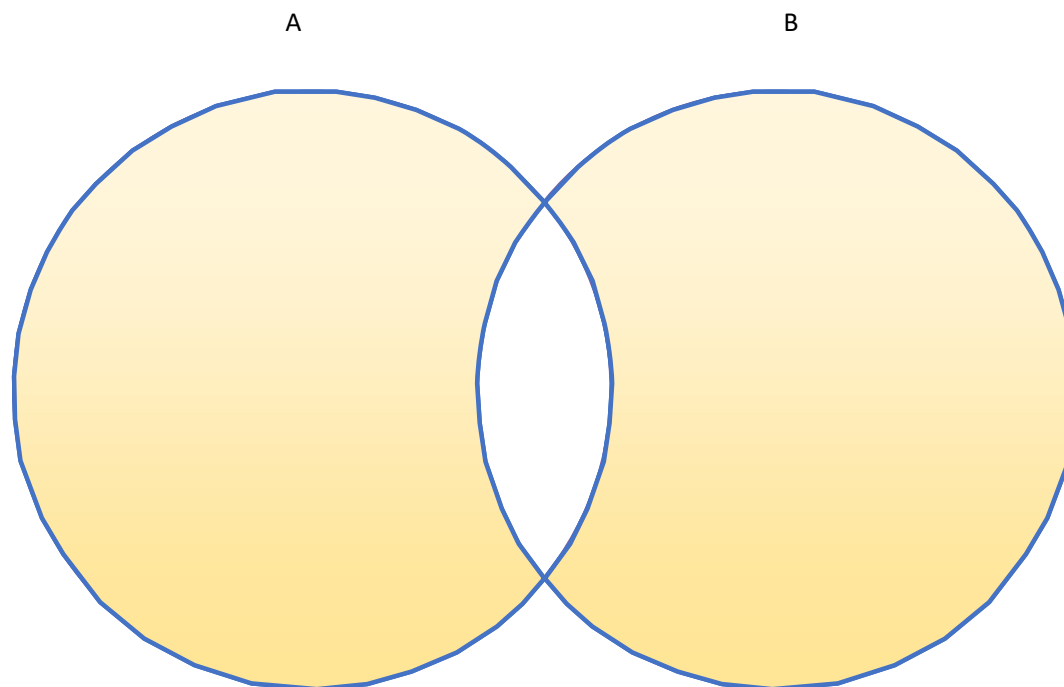


Symmetrisk forskjell på A og B

- Venn diagram

- $A \Delta B \ ((A \cup B) - (A \cap B))$

`C = A.symmetric_difference(B)` # Elementer som ikke deles av A og B

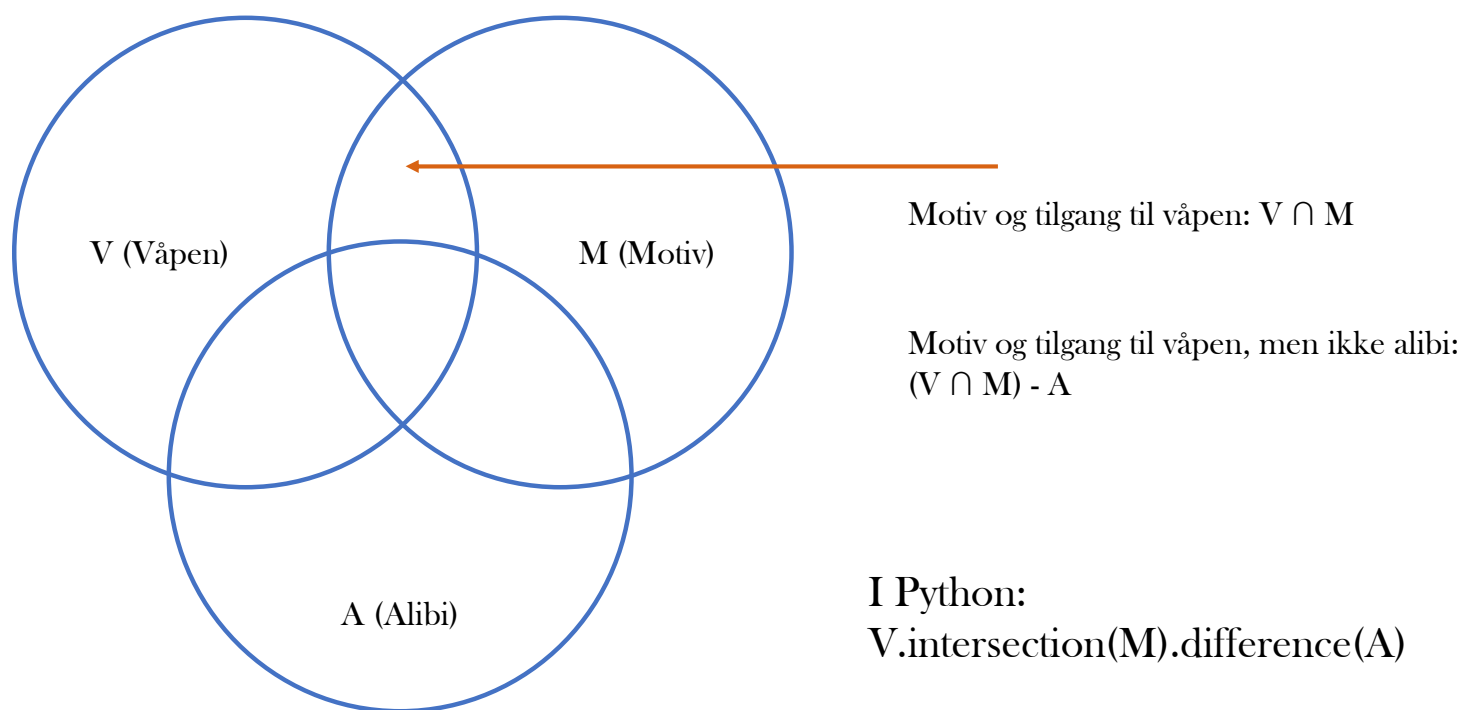


Oppgave -

- oppgave_set.py på gitlab:
- Funksjonen beskriv_set tar inn to LISTER a og b
- Den skal gjøre om listene til set, og så
 - skrive ut ulike ting:
 - - alle unike verdier som er i begge listene til sammen
 - - alle verdier som KUN er i begge listene
 - - alle verdier som er i begge listene, men ikke i begge
 - - elementene som er i a, men ikke i b

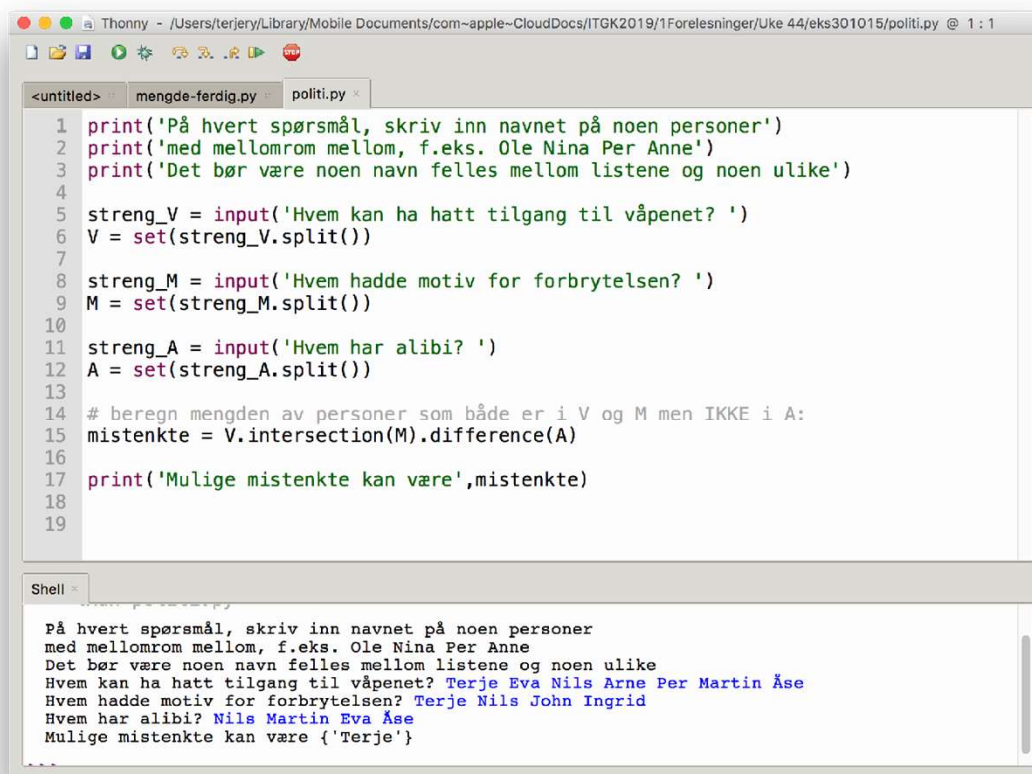
Finne den skyldige

- Bruke mengder for å finne potensielt skyldige personer
 - En mengde som inneholder hvem som har hatt tilgang på våpenet, en som inneholder hvem som har alibi, og en som inneholder hvem som har motiv



Finne den skyldige

- Bruke mengder for å finne potensielt skyldige personer
 - En mengde som inneholder hvem som har hatt tilgang på våpenet, en som inneholder hvem som har alibi, og en som inneholder hvem som har motiv



```
<untitled> mengde-ferdig.py politi.py
1 print('På hvert spørsmål, skriv inn navnet på noen personer')
2 print('med mellomrom mellom, f.eks. Ole Nina Per Anne')
3 print('Det bør være noen navn felles mellom listene og noen ulike')
4
5 streng_V = input('Hvem kan ha hatt tilgang til våpenet? ')
6 V = set(streng_V.split())
7
8 streng_M = input('Hvem hadde motiv for forbrytelsen? ')
9 M = set(streng_M.split())
10
11 streng_A = input('Hvem har alibi? ')
12 A = set(streng_A.split())
13
14 # beregn mengden av personer som både er i V og M men IKKE i A:
15 mistenkte = V.intersection(M).difference(A)
16
17 print('Mulige mistenkte kan være',mistenkte)
18
19

Shell
På hvert spørsmål, skriv inn navnet på noen personer
med mellomrom mellom, f.eks. Ole Nina Per Anne
Det bør være noen navn felles mellom listene og noen ulike
Hvem kan ha hatt tilgang til våpenet? Terje Eva Nils Arne Per Martin Åse
Hvem hadde motiv for forbrytelsen? Terje Nils John Ingrid
Hvem har alibi? Nils Martin Eva Åse
Mulige mistenkte kan være {'Terje'}
```

Kapittel 9.1 (Chapter 9.1)

Dictionaries



En dictionary er som en kommode

På fronten av hver skuff står nøkkelveidene (hva skuffen inneholder)

I skuffene ligger innholdet

Du kan si at mens dictionary er en kommode, så er sett bare skuffefrontene.

```
d = {"skuffe0" : "sokker", "skuffe1" :  
    "trøyer", ..., "nøkkeln" : verdi_n}
```

passwordliste.py

Datastruktur: *Dictionaries* Kap 9.1

```
A = {}          # Tom dictionary
A['Kari'] = 92925492    # Oppretter et element
tlf={'Jo':73540000, 'Per':92542312, 'Else':54239212}

print(tlf['Per'])    # Skriver ut verdien 92542312
```

- Vi slår opp på nøkkelverdien
- Vi bruker IKKE tallindekser som for lister

Bruk av operatorene in og not in i Dictionaries

- Man kan bruke **in** og **not in** i dictionaries for å sjekke om elementer finnes (nøkler!):

```
tlf={'Jo':73540000,'Per':92542312,'Else':54239212}
if ('Per' in tlf):
    print(tlf['Per'])
if ('Lars' not in tlf):
    print('Lars er ikke i dictionaryen')
```

Operasjoner for *dictionaries*

Operasjon	Forklaring
<code>len(d)</code>	Antall elementer i d
<code>d[k]</code>	Verdi til element i d med nøkkel k
<code>d[k] = v</code>	Sett element k til verdi v
<code>del d[k]</code>	Slett element k i d
<code>d.clear()</code>	Fjern alle elementer i d
<code>d.copy()</code>	Lag kopi av d

Operasjon	Forklaring
<code>d.items()</code>	Returnerer liste av (nøkkel,verdi) par
<code>d.keys()</code>	Returnerer liste av nøkler i d
<code>d.values()</code>	Returnerer liste av verdier i d
<code>d.get(k)</code>	Samme som <code>d[k]</code>
<code>d.get(k,v)</code>	Returnerer <code>d[k]</code> hvis k er gyldig, ellers v

`dictionary_metoder.py`

Nyttig bruk av dictionary – Oppgave!

- oppgave_count_words.py
- Ettersom vi kan bruke hva som helst som nøkkel i en *dictionary*, egner den seg svært godt til å telle forekomster i strenger, i lister eller lignende.
- Funksjonen count_words(streng) teller forekomster av hvert ord i strengen
 - Opprett liste
 - Gjør om strengen til liste av ord
 - Gå igjennom hvert ord og:
 - Er ordet i dict fra før? Legg på 1 på verdien som er der
 - Hva skal du gjøre hvis ordet ikke er det fra før? Se på d.get igjen.

BTW: `for key in sorted(f) # itererer over nøklene, sortert`

Hvis tid

- Passordliste.py inneholder en todimensjonal liste med nettsted og password.
- Funksjonen finn_passord(passordliste) spør etter nettsted, og leter opp passordet til dette hvis det finnes.
- Hvordan gjør jeg om dette til en dictionary i stedet?
- Hvis enda mere tid: vise at en dictionary gjerne kan ha en liste som verdi (i skuffen), og at dette kan være utrolig praktisk.

Oppsummering

- Sets er nyttige for å gjøre operasjoner på mengder:
 - intersection og union
 - Sets defineres ved å bruke funksjonen `set(liste)`
- Dictionary ligner på lister men man kan bruke hva som helst som nøkkel (indeks).
 - Opprettes ved å bruke `A = {}` eller `A = {'navn': 'Petter Ole'}`
 - En fordel med dictionary er at man ikke trenger en indeks som er i rekkefølge.
 - Det finnes flere kommandoer som man kan bruke på dictionary.
 - Dictionary oppfører seg som en database.