

 **NTNU**  
Det skapende universitet

**TDT4110 Informasjonsteknologi grunnkurs:**  
**Tema: Dictionaries og mengder (sets)**

- Kapittel 9

Professor Alf Inge Wang

[www.ntnu.no](http://www.ntnu.no)

---

---

---

---

---

---


---

---

2

## Læringsmål og pensum

- Mål
  - Lære å forstå og kunne bruke sets
  - Lære å forstå og kunne bruke dictionary
  - Lære å forstå og kunne bruke serialisering av objekter
- Pensum
  - Starting out with Python, Chapter 9: Dictionaries and Sets

 **NTNU**  
Det skapende universitet

[www.ntnu.no](http://www.ntnu.no)

---

---

---

---

---

---


---

---

3

## Kapittel 9.2

# MENGDER (SETS)

 **NTNU**  
Det skapende universitet

[www.ntnu.no](http://www.ntnu.no)

---

---

---

---

---

---

---

---

4

## Datastruktur for mengde (sets)

- Python har innebygd en datastruktur for å håndtere matematiske mengder (sets på engelsk).
- Hvert element i en mengde er unik, så du vil ikke få lagt til et element som finnes fra før.
- Mengder er ikke ordnet: kan ikke anta rekkefølgen.
- De fleste funksjoner for lister kan brukes for mengder.
- Mengder har flere spesifikke operasjoner bla.:
  - *intersection*: Kun elementer representert i begge mengdene
  - *union*: Alle elementer fra begge mengdene (kun en av hver)



www.ntnu.no

---

---

---

---

---

---

---

---

5

## Opprette og endre på mengder

- En mengde opprettes ved å bruke funksjonen set(x)
 

```
A = set([5,3,3,12])
B = set([True,'ost',23.2,92,False])
C = set("aaabcd") # gir 'a','b','c','d'
D = {9,4,1,3} # Lager en mengde
```
- 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)
```
- Vi prøver litt..



www.ntnu.no

---

---

---

---

---

---

---

---

6

## Iterasjon og test av mengder

- På samme måte som for lister og filer, kan for-løkke brukes for å iterere over en mengde:
 

```
for x in mengde:
    print(x) # Skriver ut alle elementer
```
- Kan også undersøke om en verdi finnes eller ikke finnes i en mengde ved bruk av `in` eller `not in`:
 

```
if 1 in A:
    print("Verdien 1 er i mengden A")
if 5 not in A:
    print("Verdien 5 er ikke i mengden A")
```
- Vi prøver litt...



www.ntnu.no

---

---

---

---

---

---


---

---

7

## 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
- Sjekk delmengde eller supermengde:  
`A.issubset(B)` # Sjekker om A er delmengde av B  
`A.issuperset(B)` # Sjekker om A er supermengde av B

 NTNU  
 Det skapende universitetet

[www.ntnu.no](http://www.ntnu.no)

---

---

---

---

---

---

---


---

---

---


8

## Oppgave: mengder



- Skriv Python-koden for å gjøre følgende:
  - Spør bruker om å skrive inn to mengder ved hjelp av input, og lagre dette som mengdene A og B.
  - Skrive ut resultatet til skjerm av følgende:
    - Snitt av A og B (intersection)
    - Union av A og B (union)
    - Forskjell på A og B (difference)
    - Symmetrisk forskjell på A og B (symmetric\_difference)
    - Test om A er delmengde av B (issubset)

**mengder.py**

 NTNU  
 Det skapende universitetet

[www.ntnu.no](http://www.ntnu.no)

---

---

---

---

---

---

---

---


---

---

9

## Kapittel 9.1

# DICTIONARIES

 NTNU  
 Det skapende universitetet

[www.ntnu.no](http://www.ntnu.no)

---

---

---

---

---

---

---

---

---

---

10

## Datastruktur: *Dictionaries*

- Dictionary er et objekt som lagrer en samling av data.
- Minner litt om lister men har klare forskjeller:
  - Defineres ved å bruke krøllparenteser {} (ALT+SHIFT 8/9 på Mac)
  - Kan bruke hva som helst som nøkkel(indeks) (ikke bare tall som i lister):
    - Tekst strenger, Heltall (men trenger ikke å være i rekkefølge), Flyttall, Sannhetsverdier (True eller False), En kombinasjon av de ovenfor

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



www.ntnu.no

---

---

---

---

---

---

---

---

11

## Bruk av operatorene in og not in i Dictionaries

- Man kan bruke `in` og `not in` i dictionaries for å sjekke om elementer finnes (som lister og set):

```
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')
```



www.ntnu.no

---

---

---

---

---

---

---

---

12

## Operasjoner for *dictionaries*

Operasjon	Forklaring	Oprasjon	Forklaring
len(d)	Antall elementer i d	d.items()	Returmerer liste av (nøkkel,verdi) par
d[k]	Verdi til element i d med nøkkel k	d.keys()	Returmerer liste av nøkler i d
d[k] = v	Sett element k til verdi v	d.values()	Returmerer liste av verdier i d
d.clear()	Slett element k i d	d.get(k)	Samme som d[k]
d.copy()	Fjerne alle element i d	d.get(k,v)	Returmerer d[k] hvis k er gyldig, ellers v
d.has_key(k)	Lag kopi av d		
	Fjernet i Python 3. Må bruke in i stedet.		

dictionary\_metoder.py



www.ntnu.no

---

---

---

---

---

---

---

---

13

## Ulike datatyper i dictionaries

- Dictionaries kan brukes til å representere sammensatt informasjon ved hjelp av lister.
- Lister gjør det mulig å knytte en nøkkel til ulike typer data. Eks. lagre telefonnummer, adresse og om vedkommende har betalt i en dictionary:

```
db = {} # Oppretter dictionary
db['Jo']=[90503020,'Logata 4, 7000 Trd',True]
db['Ann']=[50201020,'Gaamannv. 2, 5000 Bergen',False]
print(db['Jo']) # skriver ut lista med data for Jo
```



www.ntnu.no

---

---

---

---

---

---

---

---

14

## for-løkke og dictionary

- Bruker man for-løkke på en *dictionary*, løper man gjennom nøklene til *dictionary*:

```
d = {'navn': 'Per', 'alder': 28, 'IQ': 29}
for x in d: # går gjennom nøkler i d
    print(x) # skriver ut nøklene navn, alder, IQ
```

- For å sikre gjennomgang av nøkler alfabetisk:

```
for x in sorted(d.keys()): # sorterer nøkler
    print(x)
```

for\_dictionary.py



www.ntnu.no

---

---

---

---

---

---

---

---

15

## Nyttig bruk av dictionary

- 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.

```
f = {} # Oppretter dictionary for å telle forekomster
for element in liste: # sjekker elementer i ei liste
    f[element] = f.get(element,0) +1
    # Legg til +1 i dictionary med nøkkel element
```

- Vi ser på et eksempel...

telling\_dictionary.py



www.ntnu.no

---

---

---

---

---


---

---

---

## Kapittel 9.3

# SERIALISERING AV OBJEKTER

 NTNU  
Det skapende universitet

---

---

---

---

---

---

---

---

## Serialisering av objekter

- Serialisering av objekter er prosessen å konvertere et objekt til en strøm av bytes som kan lagres til fil, som senere kan lastes inn igjen.
- En persistent variabel, betyr at verdier overlever selv om man avslutter Python og avslutter datamaskinen.
- Python har biblioteket pickle som gjør det mulig å lagre og laste dictionary til/fra disk.

---

---

---

---

---

---

---

---

## Lagre dictionary til disk

- For å lagre en dictionary til disk kan biblioteket pickle brukes sammen med metoden dump.
- Må åpne en fil som binærfil og ikke som tekst, ettersom det er mer enn tekst som lagres (struktur):

```
db={'Jo':[10,'Skogata 3'],'Per':[20,'Heigt 2']}
import pickle # Importerer modulen shelve
f = open('database.dat','wb') # b=binary
pickle.dump(db,f) # Dumper db til disk
f.close() # Stenger fila
```

lagre\_dictionary.py

---

---

---

---

---

---

---

---

## Laste inn dictionary fra disk

- Laste inn dictionary fra disk gjøres ved hjelp av load metoden i pickle-biblioteket.
- Husk at åpne fila som binærfil og ikke tekst.

```
import pickle
f = open('database.dat', 'rb') # r=read, b=binary
data = pickle.load(f) # Laster inn dict fra disk
f.close()
```

laste\_dictionary.py

---

---

---

---

---

---

---

---

## 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.

---

---

---

---

---

---

---

---