

Samlinger: lister og ordbøker

Repetisjon – Tirsdag 18. november 2025

Samlinger

- Engelsk *Collections*
- Samling eller beholder for elementer
- Flere innebygde samlinger i Python:
 - Lister
 - Mengder
 - Ordbøker

Lister

- Engelsk *List*
- Opprettes med `[ ]` (klammeparenteser) eller `list()`
- Samling elementer med bestemt rekkefølge
 - Indeksert (starter på 0)
- Kan utvides og reduseres
- Kan inneholde duplikater og identiske verdier
- Kan brukes med alle datatyper (int, bool, float, str, klasser og andre samlinger)

Lister

- **Vanligste metoder**

- `.append(element)` – Legg til `element` til enden av listen
- `.insert(indeks, element)` – Setter inn `element` (og forskyver de andre) på indeksen
- `.remove(element)` – Fjerner `element` hvis den finnes
- `.pop()` – Fjerner siste elementet i listen, returnerer det som verdi
- `.pop(indeks)` – Fjerner elementet på indeksen, og returnerer det

- **Operasjoner**

- `liste_1 + liste_2` – Lager en liste som forlenger `liste_1` med `liste_2`
- `liste[indeks]` – Returnerer verdien som er på indeksen
- `liste[indeks] = element` – Endrer verdien på indeksen til `element`
- `len(liste)` – Returnerer antall elementer i listen

- **Søk og iterasjoner**

- `if element in liste` – Returnerer True hvis `element` er i listen, ellers False
- `if element not in liste` – Returnerer True hvis `element` ikke er i listen, ellers False
- `for verdi in liste` – Itererer (løkke) gjennom listen, altså henter ut hvert element og kopierer det til verdi og kjører løkken

Lister, eksempler

```
    0 1 2 3 4 5    # indeksene
tallene = [1,3,7,4,7,9]
lengde = len(tallene)    # 6
et_tall = tallene[2]    # 7
annet_tall = tallene[3] # 4
tallene.pop()           # fjerner siste element
                        # [1,3,7,4,7]
tallene.append(39)       # legger til 39 til slutt
                        # [1,3,7,4,7,39]
tallene.insert(2,42)      # setter inn 42 på indeks 2
                        # [1,3,42,7,4,7,39]
tallene[0] = 9            # endrer indeks 0 fra 1 til 9

for tall in tallene:
    print(tall)           # hvert tall i listen

if 4 in tallene:          # sjekker om vi har 4 i listen
    print("Vi har 4!")
```

Mengder

- Engelsk *set*
- Opprettes med `set()` og `{ element }` (men ikke tomme mengder)
- Mye felles med lister, *men*
 - Ingen garantert rekkefølge, ikke indeksert
 - Kan ikke inneholde duplikater eller identiske verdier

Mengder

- **Vanligste metoder**

- `.add(element)` – Legger til `element` til mengden
- `.remove(element)` – Fjerner `element` hvis den finnes, ellers feilmelding
- `.discard(element)` – Fjerner `element` hvis den finnes
- `.pop()` – Fjerner et tilfeldig element fra mengden, returnerer det som verdi

- **Operasjoner**

- `menge_1 | mengde_2` – Lager en mengde av både `menge_1` og `menge_2`
- `menge_1 & mengde_2` – Lager en mengde av elementene `menge_1` og `menge_2` har felles
- `menge_1 ^ mengde_2` – Lager en mengde av både `menge_1` og `menge_2` men uten felles verdier
- `menge_1 - mengde_2` – Lager en mengde `menge_1` men uten de i `menge_2`
- `len(menge)` – Returnerer antall elementer i mengden

- **Søk og iterasjoner**

- `if element in mengde` – Returnerer True hvis `element` er i mengden, ellers False
- `if element not in mengde` – Returnerer True hvis `element` ikke er i mengden, ellers False
- `for verdi in mengde` – Itererer (løkke) gjennom mengden, altså henter ut hvert element og kopierer det til verdi og kjører løkken
- `menge_1 == mengde_2` – Returnerer True hvis alle elementene i `menge_1` også er i `menge_2`

Mengder, eksempler

```
tallene = {1,3,7,4,7,9}    # resulterer i mengden {1,3,7,4,9}
lengde = len(tallene)      # 5
tallene.add(39)             # legger til 39
tallene.remove(7)          # fjerner tallet 7 fra mengden

for tall in tallene:
    print(tall)             # hvert tall i mengden

if 4 in tallene:            # sjekker om vi har 4 i mengden
    print("Vi har 4!")
```

Ordbøker

- Engelsk *dictionary*
- Opprettes med `{ }` eller `dict()`
- Består av samlinger med par av nøkler og verdier
- Nøklene fungerer som mengder, og må være unike
 - Nøklene kan ikke være elementer som kan endres (f.eks. lister eller mengder)
- Hver nøkkel har en tilhørende verdi
- Verdiene kan inneholde hva som helst
- Nøkkel og verdier skrives med `:` mellom, og `,` mellom hvert par
 - `{ Nøkkel : Verdi }`

Ordbøker

- **Vanligste metoder**

- `.pop(nøkkel)` – Fjerner nøkkel-verdi paret som har denne nøkkelen
- `.get(nøkkel)` – Returnerer verdien med denne nøkkelen, eller `None` om den ikke finnes

- **Operasjoner**

- `len(ordbok)` – Returnerer antall nøkkel-verdi par i ordboken
- `ordbok[nøkkel]` – Returnerer verdien med denne nøkkelen
- `ordbok[nøkkel] = verdi` – Setter ny verdi til nøkkelen

- **Søk og iterasjoner**

- `if nøkkel in ordbok` – Returnerer `True` hvis `nøkkel` er i ordboken, ellers `False`
- `if nøkkel not in ordbok` – Returnerer `True` hvis `nøkkel` ikke er i ordboken, ellers `False`
- `for nøkkel in ordbok` – Itererer (løkke) gjennom ordboken, henter ut hver nøkkel og kjører løkken
- `for verdi in ordbok.values()` – Itererer (løkke) gjennom ordboken, henter ut hver verdi
- `for nøkkel, verdi in ordbok.items()` – Itererer (løkke) gjennom ordboken, henter ut hver nøkkel og verdi

Ordbøker, eksempler

```
# ordbok med personer; navn : alder
personer = {
    "Jens": 30,
    "Jonas": 28,
    "Hans": 25,
    "Viktoria": 26,
    "Katrine": 28,
}
```

Samlinger i samlinger

- Vi kan nøste samlinger
- Et element i en liste kan være en annen liste, men også f.eks. mengde eller ordbok
- Mengder og nøklene til ordbøker kan ikke inneholde lister, siden de kan endres
- Men verdien til nøklene til ordbøker kan være lister, mengder og andre ordbøker