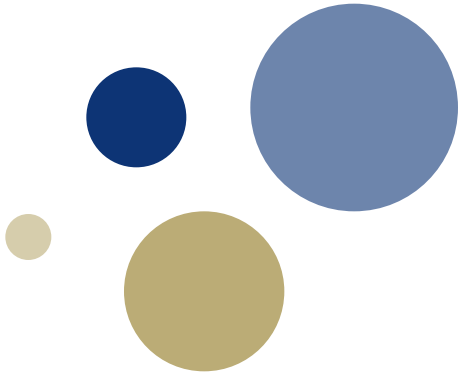
NTNU

Kunnskap for en bedre verden



Python: Lister og tupler

Gaddis: Kapittel 7

TDT4110 IT Grunnkurs

Professor Guttorm Sindre

Læringsmål og pensum

- Mål
  - Kunne forstå, og løse programmeringsproblemer ved hjelp av
    - Sekvenser
      - Lister og tupler
      - En-dimensjonale og flerdimensjonale
  - Mestre noen spesifikke Python-mekanismer for lister
    - Slicing, kopiering
    - Operatoren **in**
    - Vanlige innebygde liste-metoder og funksjoner
- Pensum
  - Starting out with Python, Chapter 7: Lists and Tuples

Sekvenser

Kapittel 7.1

Sekvenser

- *Sekvens:*
  - Et objekt som inneholder flere dataenheter
  - Enhetene lagres i sekvens (etter hverandre)
- Python har ulike sekvenstyper, bl.a.
  - lister
    - Eks.: [2, 3, 5, 7, 11]; [’Anne’, ’Ola’, ’Per’]
    - Er *muterbar* (”mutable”)
      - Kan endre innhold etter at den er opprettet
  - tupler
    - Eks.: (2, 3, 5, 7, 11); (’Anne’, ’Ola’, ’Per’)
    - Er ikke muterbar (”immutable”)
      - Kan ikke endre innhold etter at den er opprettet

Introduksjon til lister

Kapittel 7.2

Lister

- Lister i Python kan inneholde mange verdier i en variabel
  - Data av samme type, eller av ulik type
- Opprette liste i Python kan gjøres ved å…
  - ramse opp hvert element:

    dager = [’Mandag’, ’Tirsdag’, ’Onsdag’, ’Torsdag’, ’Fredag’, ’Lørdag’, ’Søndag’]
  - lage en tom liste (og sette inn elementer senere)

    kundeliste = [ ]

    kundeliste.append(’Petter’)
  - bruke ***** som repetisjonsoperator

    salg_pr_mnd = [0] * 12

    # lager lista [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
  - bruke funksjonen list( ) på et itererbart objekt

    oddetall = list(range(1, 10, 2))

    # lager lista [1,3,5,7,9]

Prosessering av lister

- Kan bruke / endre enkeltelementer ved indeksering
  - Første indeks er 0
  - Siste indeks er [n-1], der n er antall element i lista

  dager = [’Mandag’,’Tirsdag’,’Onsdag’,’Torsdag’,’Fredag’,’Lørdag’,’Søndag’]
  - Hente element, f.eks. **idag = dager[4]**
  - Endre element, f.eks. **dager[5] = ’Laurdag’ # mulig fordi den er muterbar**

- Kan også operere på hele lister, f.eks.
  - **print(dager)** #skriver ut alle elementene i lista
  - **len(dager)** #returnerer antall elementer i lista (dvs. her 7)
  - **dager.sort( )** #sorterer elementer stigende (her alfabetisk)
  - **ny_liste = dager + [0,0,0,0]**

    #gir **[’Mandag’,’Tirsdag’,’Onsdag’,’Torsdag’,’Fredag’,’Lørdag’,’Søndag’,0,0,0,0]**
  - I det siste eksemplet brukes + som konkateneringsoperator

Oppgave: legge tall i liste

- Ta utgangspunkt i koden listetest-v0.py
- Oppgave (a)
  - Fullfør funksjonen les_inn_data( )
  - Kommentarer # ??? viser hvor du må gjøre noe
- Oppgave (b)
  - Skriv funksjonen er_sortert( )
    - Bruk hodet og retursetninga som står der
  - Du kan anta at ei liste per definisjon er sortert både stigende og synkende hvis den er tom eller bare inneholder identiske elementer

kode: listetest-v0.py **løsning: listetest-v1.py**

Slicing (skiving) av lister

Kapittel 7.3

Å skive/slice lister

- Slice: et spenn av enheter som er tatt fra en sekvens
  - Slicing format: **liste[start : slutt : inkrement]**
  - kopi av elementer fra opprinnelig liste
  - fra og med start, til (men ikke med) slutt
    - Hvis start ikke er spesifisert, brukes 0 som start
    - Hvis slutt ikke spesifiseres brukes len(liste) som slutt
  - negative indekser er relative til listens slutt

```
liste = [1,2,3,4,5,6]
x = liste[0:2]          # gir x = [1,2]
x = liste[3:-1]         # gir x = [4,5]
x = liste[1:6:2]        # gir x = [2,4,6]
```

Endring av lister ved hjelp av slice

- Kan også bruke slice-uttrykk til å endre innhold:

```
A = [1,2,3,4,5,6]       # Lager ei liste A med 6 element
A[:2] = [0,0]           # Endrer to første, gir A=[0,0,3,4,5,6]
A[-2:] = [9,9]          # Endrer to siste, gir A=[0,0,3,4,9,9]
A[0::2] = [5,5,5]       # Endrer annethvert, gir A=[5,0,5,4,5,9]
A[-3:]=[ ]              # Bytter tre siste m. tom liste. Gir A=[5,0,5]
A[3:]=[4,5,6]           # Hekter på [4,5,6] etter siste. Gir A=[5,0,5,4,5,6]
A[3:3]=[1,1,1]          # Setter inn 1-ere v indeks 3. Gir A=[5,0,5,1,1,1,4,5,6]
```

- Innsetting på slutten, kan også bruke extend( )

```
A.extend([9,17])        # Gir A=[5,0,5,1,1,1,4,5,6,9,17]
```

- NB: Understreking _ i eksemplene kommer ikke i lista, er bare for å vise hvilke endringer som skjedde

Teste om et element fins i lista

Kapittel 7.4

in-operatoren

- Eksistens av element i liste kan testes med løkke:

```
funnet = False
i = 0
while (not funnet) and (i < len(liste)):
    if liste[i] == det_vi_leter_etter:
        funnet = True
    i+=1
```

- Enklere: **in**-operatoren… `if item in list:`
  - Vårt eksempel:

```
if det_vi_leter_etter in liste:
    funnet = True
```

  - True hvis den er i listen, ellers False
- Men kan trenge løkke hvis vi skal gjøre noe mer
  - F.eks. vite posisjon til elementet, telle antall treff, fjerne duplikatelementer, eller lignende

Metoder og funksjoner for lister

Kapittel 7.5

Flere nyttige listeoperasjoner

Metoder: liste.metodenavn(), Funksjoner: funknavn(liste), Setninger: verb objekt

- append(item) - innsetting til slutt
- extend(liste) – utvidelse til slutt
- insert(index, item) - inn v index
- index(verdi)
  - Indeks for 1. forekomst av verdien
- sort() – sorterer i stigende rkflg
- reverse() – snur lista
- remove(verdi)
  - Fjerner 1. forekomst av verdien
- max(), min() - størst, minst
  - Hvis tekst: alfabetisk
- del liste[indeks]
  - Fjerner element v indeks
  - del … er en setning, ikke funksjon

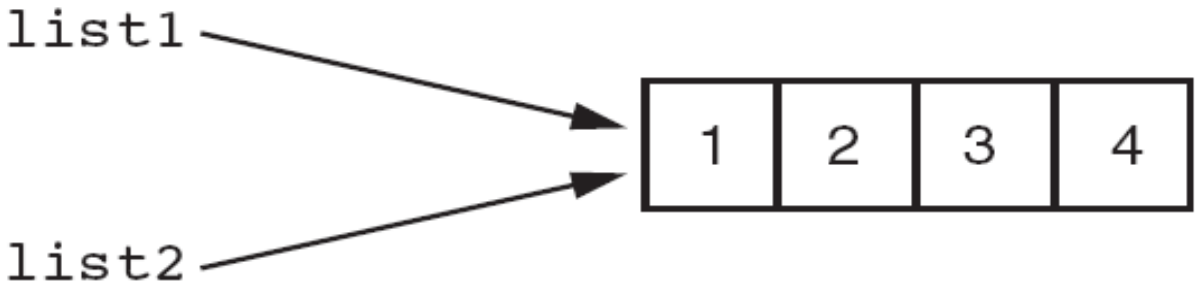
```
A=[1,2,3]
A.append(5) eller
A.extend([5])      # Gir A=[1,2,3,5]
A.insert(2,9)      # Gir A=[1,2,9,3,5]
n = A.index(9)     # Gir n = 3
# gir feilmelding hvis ikke funnet
A.sort( )      # gir A=[1,2,3,5,9]
A.reverse( )  # gir A=[9,5,3,2,1]
A.remove(2)  # gir A=[9,5,3,1]

m = max(A)  # gir m = 9

del   A[2]      # gir A=[9,5,1]
```

Kopiering av lister

Kapittel 7.6

Å kopiere lister

- Hvis man skriver **list1 = list2**, …
  - refererer begge variable til samme liste
  - Gjør vi deretter **list2[1]=9**, vil også list1 være endret



- Hvis man ønsker en kopi av ei liste og deretter vil jobbe på denne uavhengig, må man gjøre noe annet

```
list1 = [1,2,3,4]
list2 = [] + list1          # Tom liste+legger til list1
#kan også bruke for-løkke og kopiere ett og ett element
#men det er litt mer tungvint
```

Oppgave: hovedstadsquiz

- Ta utgangspunkt i koden quiz-v0.py
- PROBLEM:
  - Ingen fasit, gir poeng på alt
  - Av og til samme om igjen
- Din oppgave: Legg til fasitsjekking
  a) Opprett ei liste h_stad med hovedsteder for landene
  b) Endre koden så det kun gis poeng for riktige svar
  c) (for de kjappe) Endre så vi unngår å gjenta spørsmål vi allerede har stilt.
     - (NB: vi ønsker fortsatt å stille spørsmål i tilfeldig rekkefølge, så ei for-løkke som går gjennom landene fra første til siste er ikke løsning)

kode: quiz-v0.py

Løsn a/b: quiz-v1.py

Løsn c: quiz-v2.py, quiz-v3.py

To-dimensjonale lister

Kapittel 7.8

To-dimensjonale lister

- To-dimensjonale tabeller brukes i mange sammenhenger
- I Python realiseres 2D-tabeller som lister av lister
  - Ytterste liste: hvert element er en hel rad
  - Innerste liste (kolonnene): hvert element er en verdi
- For å prosessere 2D lister trenger man 2 indekser
- Typisk brukes nøstede løkker til å prosessere dem

2D lister (liste av lister)

```
data = ([  ['land', 'hovedstaden', 'arealet'],
           ['Norge', 'Oslo',  385186],
           ['Sverige', 'Stockholm', 449964]])
```

	Kolonne 0	Kolonne 1	Kolonne 2
Rad 0	’land’	’hovedstaden’	’arealet’
Rad 1	’Norge’	’Oslo’	385186
Rad 2	’Sverige’	’Stockholm’	449964

```
print(data[1][0])
fasit = data[2][1]
data[0][0] = ‘Land’
```

```
# Skriver ut Norge
# fasit blir Stockholm
# Endrer ‘land’ til ‘Land’
```

Lage lister av vilkårlig størrelse

- Fremgangsmåte 1:
  - Først lage tom liste, legge til elementer etter hvert
- Fremgangsmåte 2:
  - Masse-opprette elementer med gitt verdi
    - Bruke ”list comprehensions”
    - Eks.: Lage en 2-dimensjonal 10x10 matrise med 0’er:

```
tabell_10x10 = [[0 for col in range(10)]
                for row in range(10)]
```

    - Eks.: Lage en 3-dimensjonal 3x3x3 matrise med 1’ere:

```
tabell_3d = [[[1 for x in range(3)]
              for y in range(3)]
              for z in range(3)]
```

Tupler

Kapittel 7.9

Tupler

- Tuppel: en ikke-muterbar sekvens (kan ikke endres)
  - Likner ellers på lister
  - Når den er opprettet kan innholdet ikke endres
  - Format: **tuple_name = (item1, item2, …)**
  - Tupler støtter operasjoner slik som lister gjør det
    - Elementer kan hentes med indekser
    - Har metoder som index
    - Innebygde funksjoner som len, min, max
    - Har slicing-uttrykk
    - Har operatorene in, + og *

Tupler (forts.)

- Tupler støtter ikke endringsmetoder:
  - ~~append, remove, insert, reverse, sort~~
  - Ulempe: mindre fleksible
    - Men noen Python-operasjoner krever tupler
  - Fordel:
    - Programmet kjører raskere
    - Tryggere for data som ikke skal endres (f.eks. ukedager, måneder)
- Funksjonene list() og tuple(): konverterer mellom de to

```
tuppel = (1,2,3)
print(tuppel[0])          # skriver 1
liste = list(tuppel)      # gir liste = [1,2,3]
tuppel2 = tuple(liste)    # gir tuppel2 = (1,2,3)
```

Oppgave: 2D-lister og tuppel

- Ønsker å utvide quiz til å spørre om mer enn hovedsteder i landene
  - F.eks. areal, folketall, …
- Passer å bruke 2D-liste (tabell)
  - en rad for hvert land
  - en kolonne for hver type opplysning
  - Spørre om tilfeldig faktum for tilfeldig land
  - Huske hvilke spørsmål som fortsatt er ledig
- Nesten ferdig program, klarer du å skrive linjene som mangler?

kode: quiz-v4-oppgave.py

løsn: quiz-v4.py

Oppsummering

- Dette kapitlet dekket:
  - Lister
    - Repetisjons- og konkateneringsoperatorer
    - Indeksering
    - Teknikker for å prossesere lister (gå igjennom lister)
    - Å slice (plukke ut deler) og kopiere lister
    - Listemetoder og innebygde funksjoner for lister
    - To-dimensjonale lister
  - Tupler
    - Ikke muterbar (kan ikke endres)
    - Forskjeller fra lister, fordeler og ulemper

Neste uke: Strenger (kap 8)

- Aktuelle spørsmål for quiz:
  - «What is wrong with the following code?» (Checkpoint 8.6)
  - «What will the following code display?» (Checkpoint 8.7)
  - «What will the following code display?» (Checkpoint 8.10)
  - «What is the output of the following code?» (Checkpoint 8.14)
  - «Write a loop that counts the number of uppercase…» (Checkpoint 8.17)
  - «Assume that the following statement…. Write …» (Checkpoint 8.18, 8.19)
  - «The statement print('QWERTY'*2) will print…» (Review, Multiple Choice 3)
  - «This function returns the length of a string» (Review, Multiple Choice 4)
  - «This string method returns a copy… leading whitespace…» (Review, Multiple Choice 5)
  - «This string method returns True if a string contains only numeric…» (Review, Multiple Choice 9)
  - «What does the following code display?» (Review, Short Answer 1)
  - «What does the following code display?» (Review, Short Answer 4)
- Noen av disse gis i «kahootisert form»
- Pluss 1-2 helt uannonserte spørsmål, men også om strenger