NTNU

Det skapende universitet

TDT4110 Informasjonsteknologi grunnkurs:

Tema: Lister og tupler

- 3rd edition: Kapittel 7

Professor Alf Inge Wang

Læringsmål og pensum

- Mål
  - Lære om
    - Sekvenser
    - Lister
    - List Slicing
    - Finne elementer i lister med operatoren in
    - Liste-metoder og nyttige innebygde funksjoner
    - Kopiere og prosessere lister
    - To-dimensjonale lister
    - Tupler
- Pensum
  - Starting out with Pyton:

  Chapter 7 Lists and Tuples

NTNU
Det skapende universitet

Sekvenser

Kapittel 7.1

Sekvenser

- Sekvens: et objekt som inneholder flere dataenheter
  - Enhetene lagres i sekvens (etter hverandre)
- Python har ulike sekvenstyper, inkludert lister og tupler
  - En *liste* kan endre verdien av sine elementer og lengde etter at den er opprettet og kalles *muterbar* (”mutable”).
  - En *tuppel* kan ikke endre verdien av sine elementer eller lengde etter at den er oppretter og kalles ”immutable”

Introduksjon til lister

Kapittel 7.2

Lister

- I Python brukes liste til å holde på all mulig informasjon som ligger etter hverandre som perler på en snor.
- Innholdet i en liste kan være av typen tall (både heltall og flyttall), sannhetsverdier (True eller False), tekst, eller en annen liste.
- Funksjonen `print()` kan brukes til å skrive ut innholdet i ei liste og `list()` kan brukes til å omgjøre visse objekter til en liste.
- Opprette liste i Python gjøres på formen:

```
variabelnavn = [element1, element2, element3, …]
```

- Her er et eksempel på en liste i Python:

```
liste = ['Ole', 'Dole', 'Doffen']
```

Innledning til lister (forts.)

- En liste av heltall:

- En liste av strenger:

- En liste som inneholder flere ulike datatyper:

kode: various_lists.py

Lister med et bestemt antall elementer

- I Python kan man også lage en liste med et bestemt antall elementer ved hjelp av repetisjonsoperatoren `*`.
- Når man skriver en liste etterfulgt av `*` og et heltall, vil man repetere lista like mange ganger som størrelsen på heltallet:

```
liste = [0]*10 # Lager lista [0,0,0,0,0,0,0,0,0,0]
liste = [1,2,3] * 3 # Lager lista [1,2,3,1,2,3,1,2,3]
```

- Ønsker man å lage en liste som er tom, kan man fylle lista med verdien `None`, som er et reservert ord for tomme elementer:

```
liste = [None]*5 # liste med 5 tomme elementer
```

- Vi prøver litt…

kode: repetisjonsoperator.py

Iterere over ei liste ved hjelp av ei for-løkke

- Man kan gå igjennom en liste element for element (iterere) ved å bruke ei for-løkke på følgende format:

```
for element in liste:
    print(element) # Eller gjør noe annet med element
```

  - `liste` er navnet på variabelen som inneholder en liste
  - `element` er en variabel som får verdien av hvert element i lista, element for element

kode: iterasjon_liste.py

Oppgave: lister #1

- Skriv Python-koden for å gjøre følgende:
  - Opprett lista 1,3,5,7,9 repetert fem ganger i variabelen `liste`
  - Skriv ut hvert element i lista ganget med seg selv

kode: lister_1.py

Indeksering

- Indeks: et tall som spesifiserer et elements posisjon i en liste
  - Gir tilgang til de enkelte elementer i en listen
  - Indeksen til det første elementet er 0, det andre 1, og det n-te elementet er n-1
  - Negative indekser identifiserer posisjoner relativt til listens slutt
    - Indeksen-1 identifiser det siste elementet, -2 det nest siste osv.

Indeks:	**0**	**1**	**2**	**3**		**n-3**	**n-2**	**n-1**
Verdier:	43	95	12	5	…	43	23	54

Hente ut innhold fra et element i en liste

- For å få tak i innholdet for et bestemt element i en liste, skriver man variabelnavnet, firkantparantes og nummeret på elementet, f.eks:


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

- Merk at det første elementet er element 0 (ikke 1)

NTNU
Det skapende universitet

Funksjonen len

- En «exception» av typen IndexError utløses hvis en ugyldig indeks brukes (kommer tilbake til dette)
- Funksjonen len: returnerer lengden til en sekvens, som for eksempel lengden av ei liste

```
size = len(my_list)
```

  - Returnerer antallet elementer i listen, slik at det siste elementet er len(list)-1 ettersom indeks starter på 0 og ikke 1
  - Kan brukes til å hindre en IndexError-exception når man itererer gjennom en liste i en løkke

Oppgave: lister #2

- Skriv Python-koden for å gjøre følgende:
  - Opprett lista 1,3,5,7,9 repetert fem ganger i variabelen `liste`
  - Skriv ut hvert tredje element i lista ganget med seg selv ved hjelp av `for`-løkke, `range()` og `len()`.

kode: lister_2.py

Lister kan endres (muteres)

- Muterbare sekvenser: elementer i en liste som kan endres
  - Lister er muterbare, dermed kan deres elementer endres
  - `liste[1] = ny_verdi` kan brukes til å tilordne en ny verdi til et element i en liste
    - Må bruke en gyldig indeks for å forhindre at en IndexError-exception utløses (dvs. at elementet for gitt indeks eksisterer).
  - Eks på endring av ei liste:

```
liste = [1, 2, 3, 4, 5]
liste[0] = 3 # Endrer element 0 til 3
#Gir liste = [3, 2, 3, 4, 5]
```

Å konkatinere lister

- Konkatinering: å føye to ting sammen
- Operatoren + kan brukes til å konkatinere to lister

```
A=[1,2,3,4,5]
B=[2,4,6,8,10]
C=A+B # gir [1,2,3,4,5,2,4,6,8,10]
```

- Den utvidede operatoren += kan også brukes til å konkatinere lister (legge til element(er) til lista).

```
A=[1,2,3]
A+=[4] # Gir A=[1,2,3,4] samme som A = A +[4]
```

Oppgave: lister #3

- Skriv Python-koden for å gjøre følgende:
  - Opprett liste med tallene 2,4,6,8,10,12,14,16,18,20 i variabelen `liste`
  - Skriv ut lista til skjerm
  - Bytt ut hvert andre element med 3 gangen starter med element med indeks 1 som vil gi lista [2,**3**,6,**6**,10,**9**,14,**12**,18,**15**]
  - Skriv ut lista til skjerm.

kode: lister_3.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]`
  - Spenn er ei liste som inneholder kopier av elementer fra start fram til, men som ikke inkluderer end
    - Hvis start ikke er spesifisert, brukes 0 som startindeks
    - Hvis end ikke spesifiseres brukes len(list) som sluttindeks
  - Slicing-uttrykk kan inneholde stegverdier og negative indekser, som 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

- Man kan endre på innhold på deler av lister ved hjelp av slice på følgene format:

```
liste[start:slutt:inkrement] = […]
```

  - Erstatter den delen av lista spesifisert på venstre side av er-lik tegnet med lista spesifisert på høyre side.

```
A=[1,2,3,4,5,6] # Lager en liste A
A[:2]=[0,0] # Erstatter to første elementer. Gir A=[0,0,3,4,5,6]
A[-2:] = [9,9] # Erstatter to siste elementer. Gir A=[0,0,3,4,9,9]
A[0::2] = [5,5,5] # Erstatter hvert andre element: Gir A=[5,0,5,4,5,9]
A[-3:]=[] # Erstatter tre siste elementer med tom liste. Gir A=[5,0,5]
A[3:]=[4,5,6] # Hekter på [4,5,6] etter siste element. Gir A=[5,0,5,4,5,6]
```

NTNU
Det skapende universitet

Sette inn flere elementer i en liste

- Hvis man ønsker å legge til flere elementer på slutten av ei liste kan man bruke methoden `extend`:

```
A = [1,2,3]
A.extend([4,5,6]) # gir A = [1,2,3,4,5,6]
```

- Hvis man ønsker å legge til flere elementer til en liste på gitt indeks kan man bruke slicing til dette:

```
A=[1,2,3,4,5,6,7,8,9,10]
A[3:3]=[100,101,102] # Setter inn tre elementer på
                     # indeks 3 gir
                    # A=[1,2,3,100,101,102,4,5,6,7,8,9]
```

Finne elementer i liste ved hjelp av operatoren in

Kapittel 7.4

Å finne enheter i listen med operatoren in

- Du kan bruke operatoren in til å avgjøre om en enhet finnes i en liste
  - Generelt format: `if item in liste:`
  - Returnerer `True` hvis enheten er i listen, eller False hvis den ikke er der
- Tilsvarende kan du bruke operatoren `not` in for å avgjøre om et element ikke finnes i listen

```
navneliste=["Petter","Kalle","Jo"]
if "Frank" not in navneliste:
    print("Frank er ikke på lista!")
```

NTNU
Det skapende universitet

kode: in_operatoren.py

Metoder og funksjoner for lister

Kapittel 7.5

Listemetoder og nyttige, innebygde funksjoner

- append(item): brukes til å legge enheter til en liste – item tilføyes til slutt i lista:

```
A=[1,2,3]
A.append(5)        # Gir A=[1,2,3,5]
```

- index(item): brukes til å finne hvor man finner en enhet i listen
  - Returnerer indeksen til det første elementet i listen som inneholder elementet item
  - Utløser en ValueError-exception hvis item ikke finnes i listen

```
A=[1,2,7,4,3,2]
print(A.index(7)) # Gir resultatet 2
```

Listemetoder, og nyttige, innebygde funksjoner (forts.)

- insert(index, item): brukes til å smette enheten item inn i listen ved posisjon index i listen
- sort(): brukes til å sortere elementene i listen i stigende rekkefølge
- remove(item): fjerner den første forekomsten av enheten `item` i listen
- reverse(): snur rekkefølgen på elementene i listen

Listemetoder og nyttige, innebygde funksjoner (forts.)

- del: fjerner et element fra en spesifikk indeks i en liste
  - Generelt format: `del liste[i] # Merk ingen paranteser`
- Funksjonene min og max: innebygde funksjoner som returnerer enheten som har den laveste eller høyeste verdien i en sekvens
  - Sekvensen sendes som argument
  - Fungerer også på tekststrenger

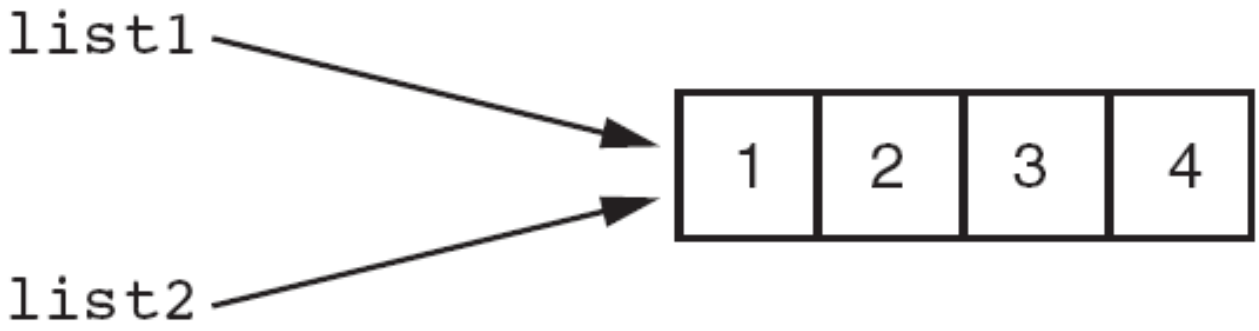
```
A=[8,3,6,45,12,23,5,65,76,34,2,2]
print(min(A)) # gir 2
print(max(A)) # gir 76
```

Kopiering av lister

Kapittel 7.6

Å kopiere lister

- Hvis man skriver `list1 = list2`, refererer dette til samme liste.
  - Det vil si at man ikke kopierer og lager en ny liste, men at begge variablene peker til akkurat samme liste.



Å kopiere lister (2)

- For å ta en kopi av en liste så må du kopiere hvert element i listen
  - To metoder gjør dette
    - Å lage en ny, tom liste og bruke en for-løkke til å legge til kopier av hvert element fra den opprinnelige listen til den nye listen

```
liste1 = [1,2,3,4]
liste2 = [] # Lager tom liste
for item in liste1:
        liste2.append(item)
```

    - Å lage en ny, tom liste og konkatinere den gamle listen til den nye

```
liste1 = [1,2,3,4]
liste2 = [] + liste1 # Tom liste + legger til liste1
```

NTNU

Det skapende universitet

TDT4110 Informasjonsteknologi grunnkurs:

Tema: Lister og tupler – Del 2

- 3rd edition: Kapittel 7

Professor Alf Inge Wang

Prosessering av lister

Kapittel 7.7

Å prosessere lister

- Listeelementer kan brukes i beregninger
- For å beregne antallet numeriske verdier i en liste, kan du bruke løkke med en tellende variabel
- For å finne gjennomsnittlig verdi i en liste:
  - Beregne summen av verdier i listen
  - Dividere summen med len(list)
- Lister kan brukes som argumenter til funksjoner
- En funksjon kan returnere en referanse til ei liste

NTNU
Det skapende universitet

kode: snitt.py

To-dimensjonale lister

Kapittel 7.8

To-dimensjonale lister

- To-dimensjonale liste: liste som inneholder andre lister som elementer
  - Også kjent som nøstede lister
  - Vanlig å betrakte to-dimensjonale lister som om de har rekker og kolonner
  - Nyttig til å jobbe med flere datasett
- For å prosessere data i to-dimensjonale lister trenger man å bruke indekser
- Typisk brukes nøstede løkker til å prosessere dem

Lage to-dimensjonale lister (liste av lister)

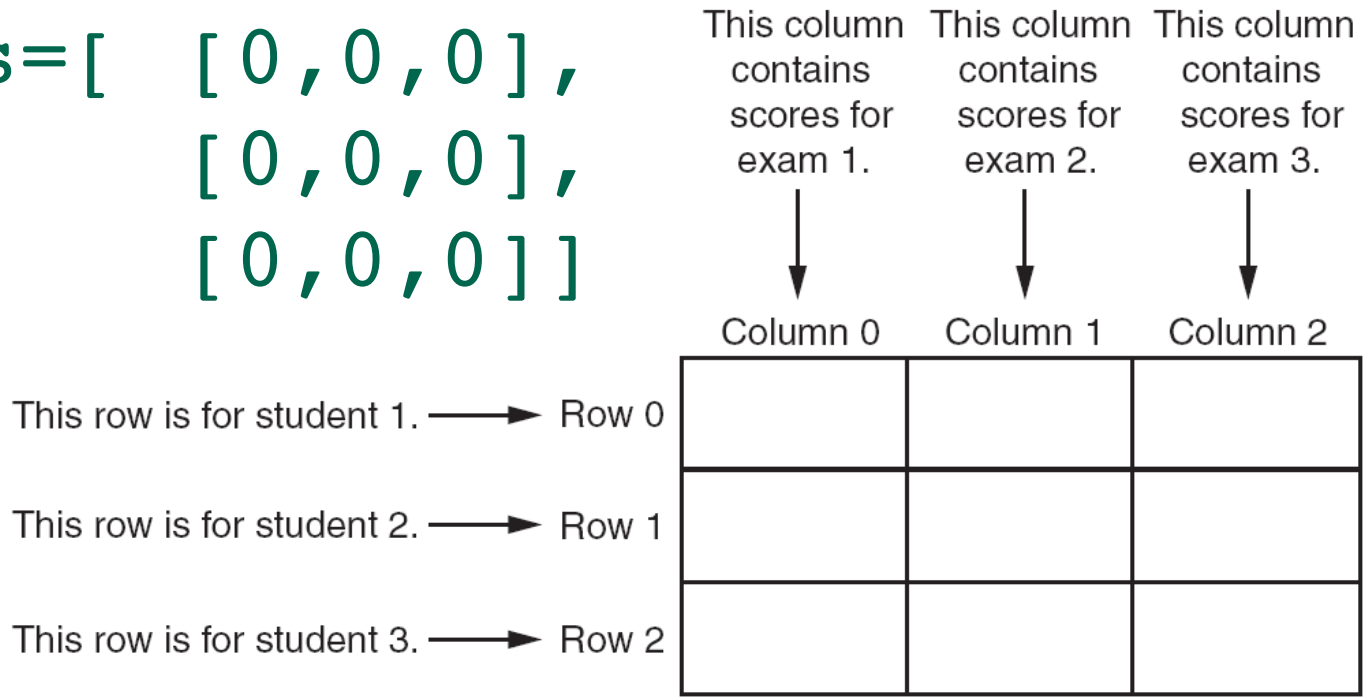
```
students=[ ['Joe','Kim'],
           ['Sam','Sue'],
           ['Kelly','Chris']]
```

	Column 0	Column 1
Row 0	'Joe'	'Kim'
Row 1	'Sam'	'Sue'
Row 2	'Kelly'	'Chris'

NTNU
Det skapende universitet

Lage to-dimensjonale lister (lister av lister)

```
scores=[ [0,0,0],
         [0,0,0],
         [0,0,0]]
```



Hente ut verdier fra to-dimensjonale lister:

	Column 0	Column 1	Column 2
Row 0	scores[0][0]	scores[0][1]	scores[0][2]
Row 1	scores[1][0]	scores[1][1]	scores[1][2]
Row 2	scores[2][0]	scores[2][1]	scores[2][2]

```
print(scores[1][1]) # Skriver ut element (1,1)
x=scores[2][0] # Setter x lik element (2,0)
scores[0][0] = 9 # Setter element (0,0) lik 9
```

Lage to-dimensjonale tabeller av vilkårlig størrelse

- Man kan også opprette en fler-dimensjonal tabell av en gitt størrelser uten å angi alle elementene:
  - Lage en 2-dimensjonal 10x10 matrise med 0er:

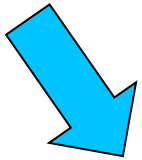
```
tabell_10x10 = [[0 for col in range(10)]
                for row in range(10)]
```

  - Lage en 3-dimensjonal 3x3x3 matrise med 1ere:

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

Oppgave: matriser

- Skriv Python-koden for å gjøre følgende:
  - Opprett en 3 x 3 tabell av 0er
  - Fyll tabellen så den blir som følgende:
  - Skriv ut tabellen rekke for rekke



2	4	6
8	10	12
14	16	18

NTNU
Det skapende universitet

kode: matriser.py

Tupler

Kapittel 7.9

Tupler

- Tuppel: en ikke-muterbar sekvens (kan ikke endres)
  - Likner ellers på lister
  - Når den er opprettet kan den 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 metoder som innebærer endring av sekvensen (naturlig nok):
  - append
  - remove
  - insert
  - reverse
  - sort

Tupler (forts.)

- Fordeler med å bruke tupler fremfor lister:
  - Det går raskere å prossesere dem
  - Tupler er trygge (de kan ikke tukles med)
  - Noen Python-operasjoner krever tupler
- Funksjonen list(): gjør tuppel om til liste

```
tuppel = (1,2,3)
liste = list(tuppel) # gir liste = [1,2,3]
```

- Funksjonen tuple(): gjør liste om til tuppel

```
liste=[4,5,6]
tuppel = tuple(liste) # gir tuppel = (4,5,6)
```

Generering av store lister

Hvordan lage en liste av svært mange tall som teller oppover?

Liste med tall fra 1 til 1000: [1,2,3,4,5...998,999,1000]:

```
liste = [i+1 for i in range(1000)]
```

Liste med 3-gangen opp til 1002: [3,6,9,12...999,1002]:

```
liste = [i+3 for i in range(0,1000,3)]
```

Liste med kvadratet til 2-gangen opp til 198*198:

```
liste = [i*i for i in range(2,200,2)]
```

Oppsummering

- Dette kapittelet dekket:
  - Lister, som i
    - Repetisjons- og konkatineringsoperatorer
    - 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, som i
    - Ikke muterbar (kan ikke endres)
    - Forskjeller fra lister, og fordeler fremfor lister