[1, 2, 3,4]

while...

for...

Løkker, lister og ordbøker

Repetisjonskurs 2018 IN1000

Petter Mæhlum, UiO

{'ost':'NN', 'spise':'VT', 'på':'PP'}


KAHOOT

Lister

[[1, 2, 3], [4, 5, 6]]

['a', 'b', 'c']

- Vi bruker klammeparanteser []
- Oppretter tom liste slik:
  - liste = []
- Vi kan også spesifisere elementene direkte:
  - liste = [1,2,3]
- Lister kan inneholde alt mulig!
  - Heltall
  - Strenger
  - Andre lister
  - Alle andre typer objekter
  - …

[Celle(),Celle()]

Ordbøker (Dictionary)

- Ordbøker skriver vi med krøllparantes {}
- Ei tom ordbok oppretter vi slik:
  - Ordbok = {}
- Ei ordbok har nøkkelverdier (key) og innholdsverdier (value)
- Disse skilles med kolon:
  - {nøkkelverdi:innholdsverdi, nøkkelverdi:innholdsverdi}
- Når vi itererer (går gjennom) ei ordbok, er det nøklene vi går gjennom
- ...men vi kan spesifisere nøkler/innholdsverdier hvis vi vil:
  - ordbok.keys()
  - ordbok.values()

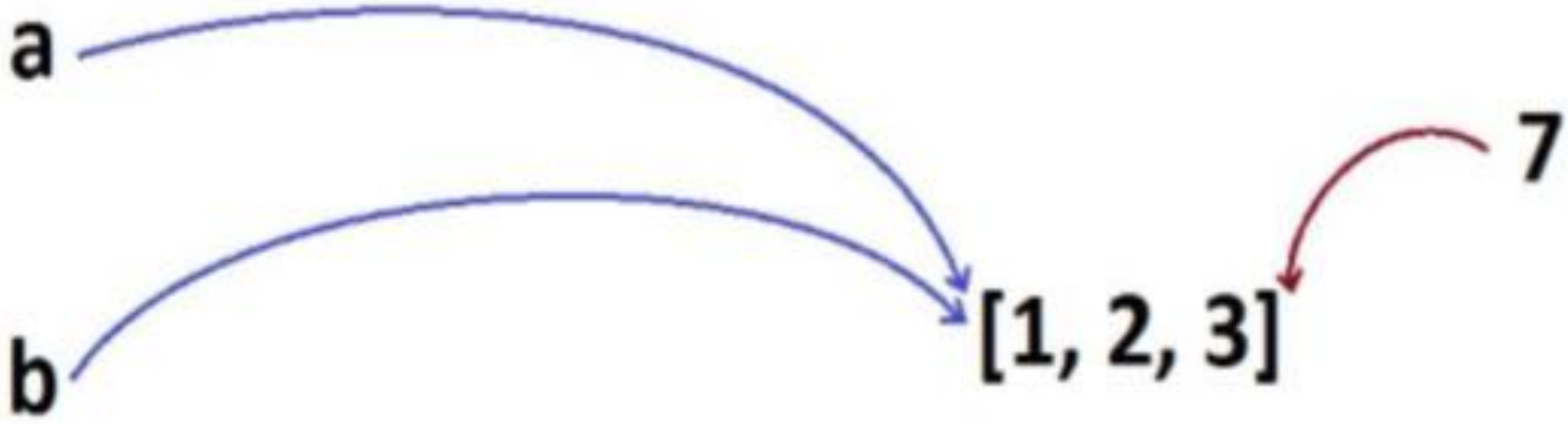
Legge til et nytt element i ordbok

```
>>> ordbok = {'a': 1, 'b': 2, 'c': 3}
>>> ordbok['d'] = 4
>>> print(ordbok)
{'a': 1, 'b': 2, 'd': 4, 'c': 3}
```

- Elementene i ei ordbok er ikke ordna.

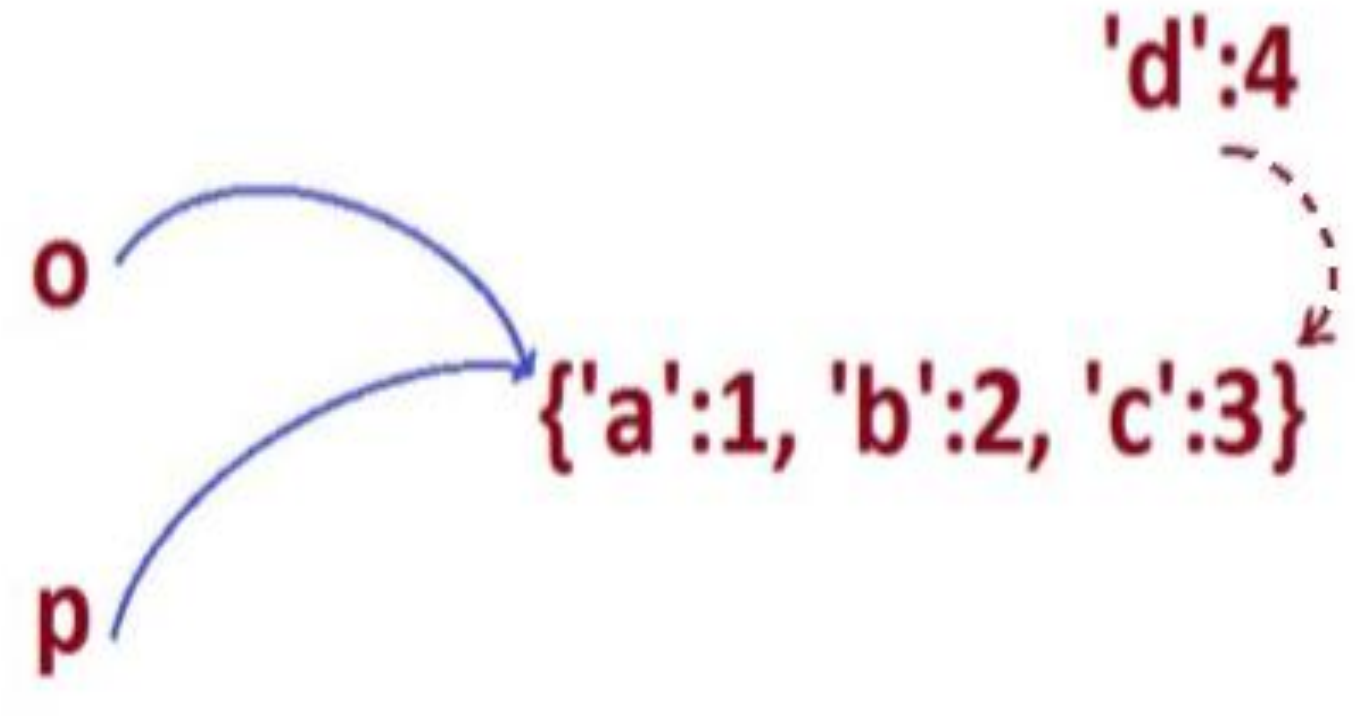
Referanser (lister)

```
>>> a = [1,2,3]
>>> b = a
>>> b.append(7)
>>> print(b)
[1, 2, 3, 7]
```



Referanser (ordbøker)

```
>>> o = {'a':1,'b':2,'c':3}
>>> p = o
>>> p['d'] = 4
>>> print(o)
{'c': 3, 'd': 4, 'b': 2, 'a': 1}
```



While-løkker

< > <= >= ==

not, in, or, and

- Syntaks:

```
while X :
    <kode>
```

- Hvor X er et boolsk uttrykk
- Boolske operatorer
- Hver ‘runde’ (iterasjon) sjekker på nytt om uttrykket er sant
- Kan tenke på det som «så lenge det er sant at..., gjør følgende:»

Viktig at while-løkkene får muligheten til å slutte

```
tall = 3
while tall < 6:
    print(tall)
    tall += 1
```

Utskrift:

```
3
4
5
```

```
liste = []
counter = 0
while len(liste) < 4:
    liste.append(counter**2)
    counter += 1

>>> liste
[0, 1, 4, 9]
```

While-løkker er fine når antall iterasjoner er ukjent

```
def terning():
    svar = input("Skriv 'kast' for et tilfeldig tall. Alt annet avslutter programmet."
    while svar == "kast":
        tilfeldig = random.randint(0,6)
        print("Terningen viser",  tilfeldig)
        svar = input("Skriv 'kast' for et tilfeldig tall.")
    print("Programmet avsluttes")
```

While og «== True»

```
def erPartall(tall):
    return tall % 2 == 0

def sjekkePartall():
    tall = int(input("Skriv et heltall"))
    while erPartall(tall):
        print(tall, "er et partall.")
        tall = int(input("Skriv et heltall"))
    print("Ikke partall. Ferdig.")

sjekkePartall()
```

For-løkker

- De enkleste for-løkkene har følgende syntaks:

```
for X in Y:
    ...
```

- «in» her betyr «i». Vi bruker den litt annerledes enn når vi bruker «in» som boolsk operator.
- X er en variabel som vi bestemmer selv
- Navnet til X bør si noe om *typen* til verdien.
- Skopet til X er kun i løkka
- Y er noe vi kan telle gjennom: lister, strenger, ordbøker, mengder, tupler

Gå gjennom liste : eksempel

```
matliste = ["potetgull","brus","gulrot"]

for mat in matliste:
    print(mat)
```

```
potetgull
brus
gulrot
```

For-løkker med range

- Range gir oss muligheten til å iterere gjennom tall i et spenn.
- Nyttig hvis vi skal gjøre utregninger, eller hvis vi bare vil gjøre noe et visst antall ganger.
- Eksempel: hva er summen av alle tallene fra og med 0 til og med 10? [0,10]
- OBS: Når vi summerer bruker vi 0,
- men hva bruker vi når vi tar
- produktet av noe?

```
summen = 0
for i in range(11):
    summen += i
print(summen)
```

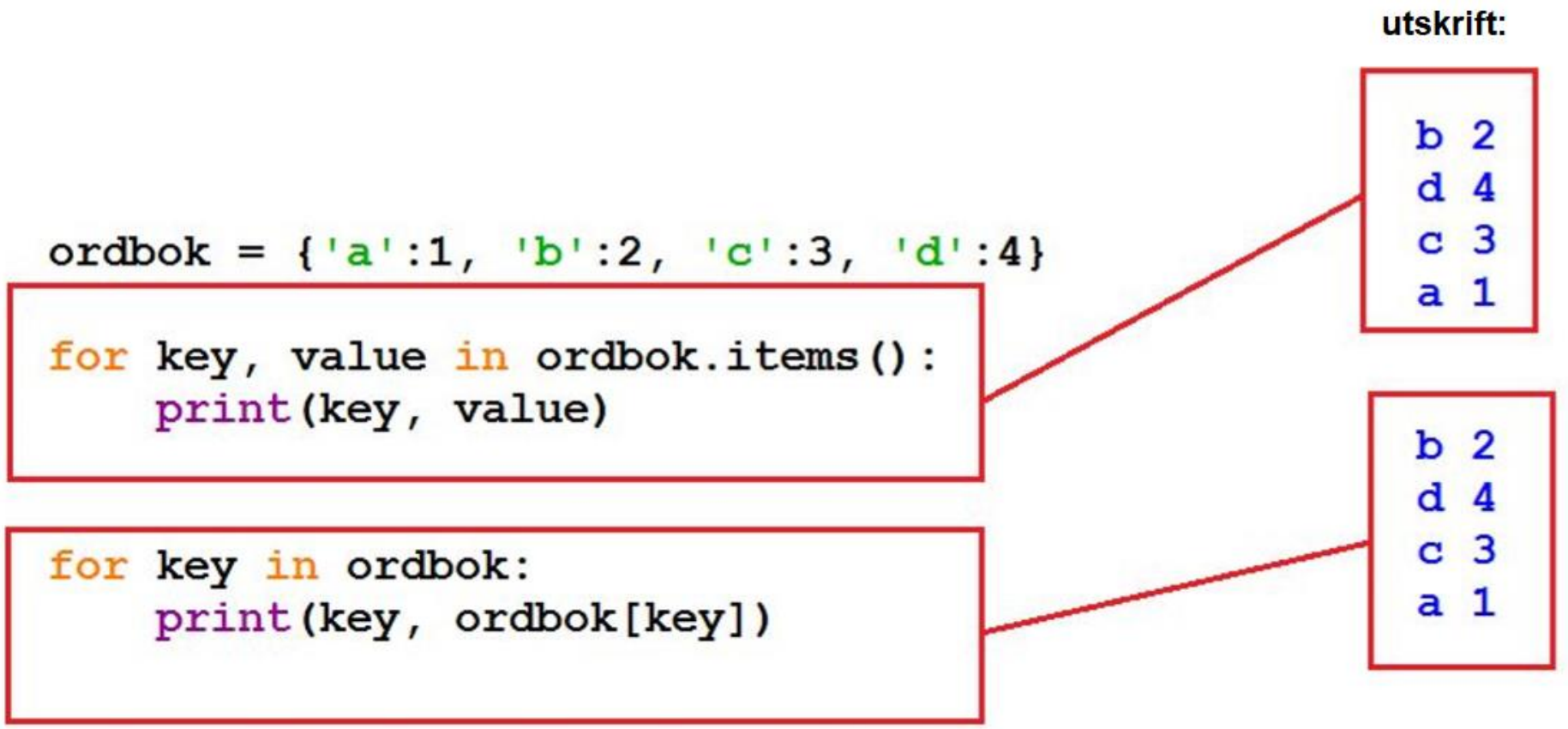
```
55
```

For-løkker med range ()

- Hvis vi bare skriver ett tall blir det den øverste grensa
- Hvis vi skriver to tall er det fra og med det første, og til men ikke inkludert det andre.
- Det tredje tallet indikerer hvor mange steg vi tar
- Uspesifisert starter vi på 0, og hopper 1 tall om gangen.

```
range(5) impliserer range(0, 5, 1)
range(1, 5) impliserer range(1, 5, 1)
range(1, 5, 2) impliserer range(1, 5, 2)
```

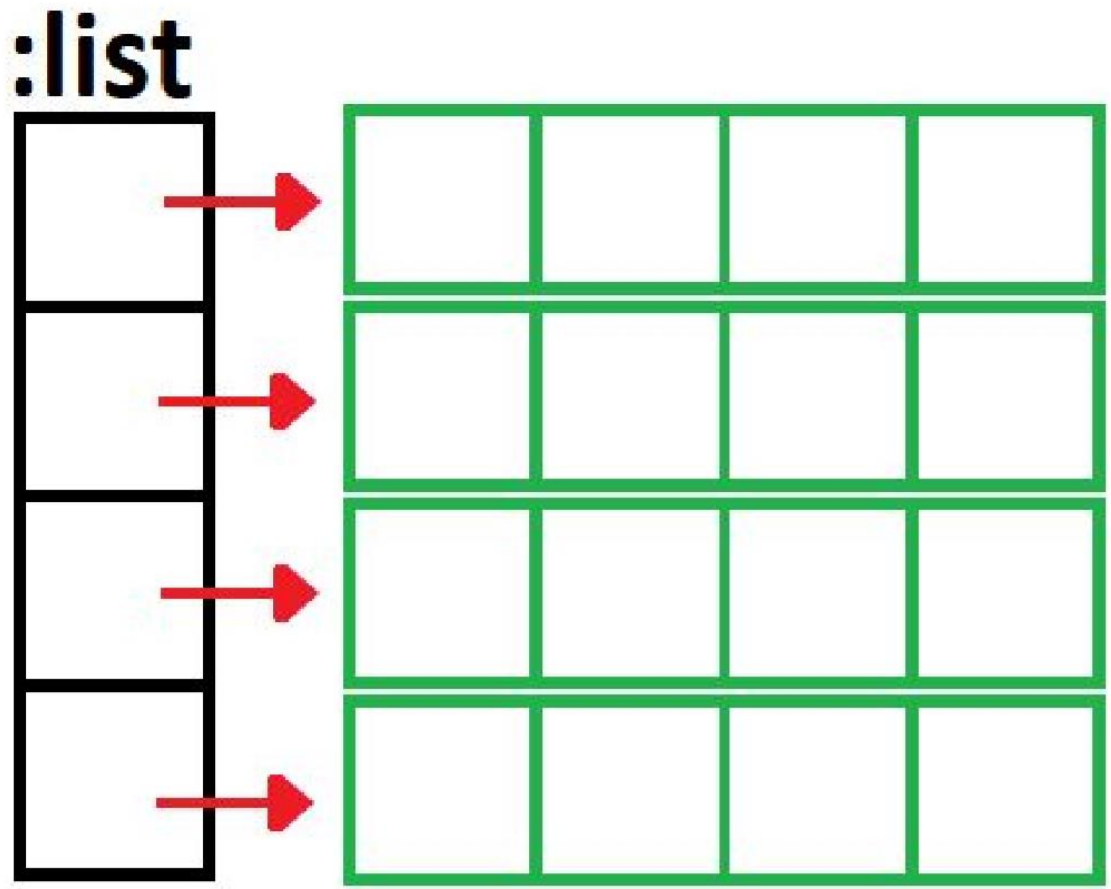
Iterering gjennom ei ordbok: to måter

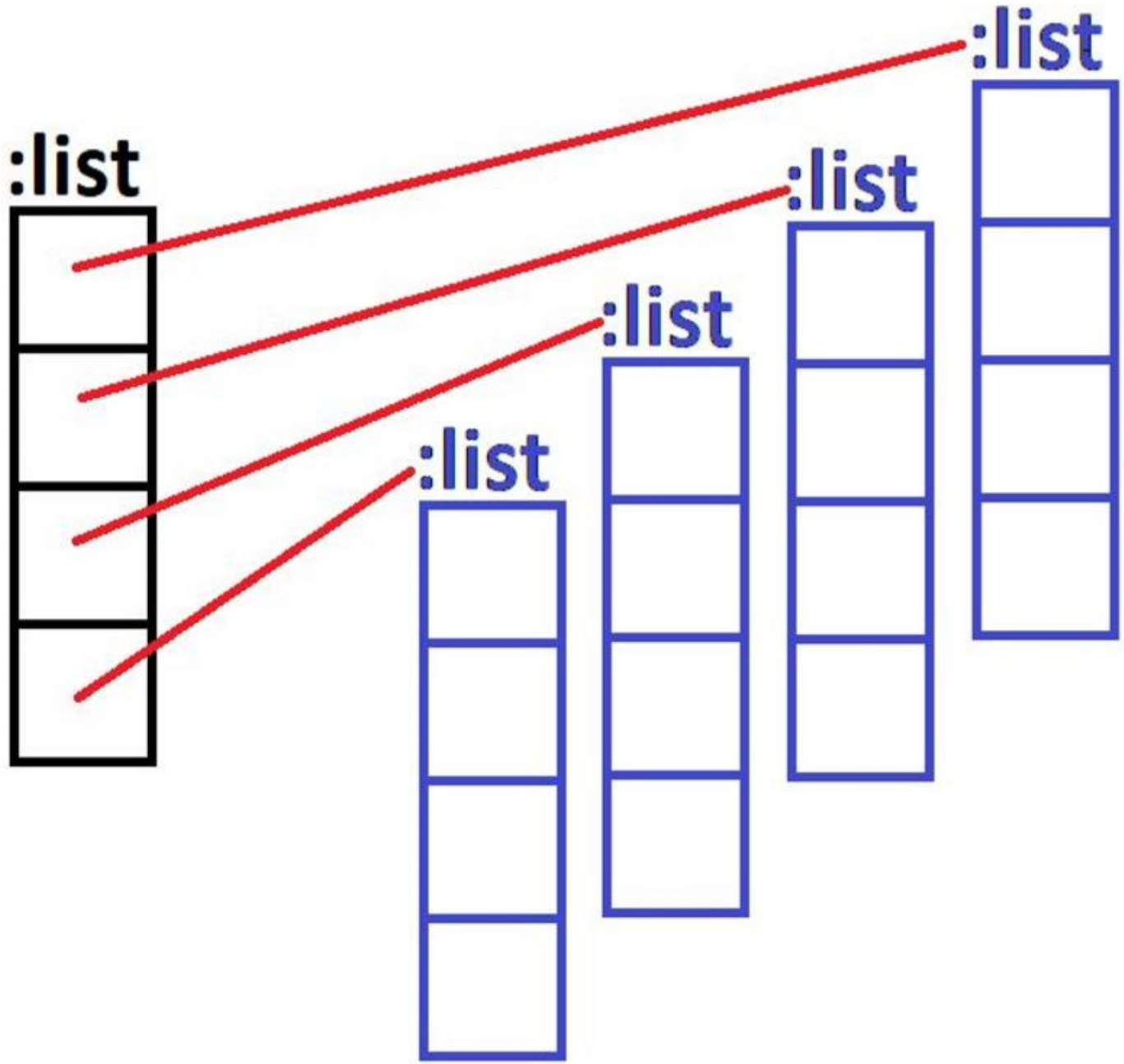


Nøstet liste, nøstet løkke

- Ei nøsta lista er ei liste hvor hver plass refererer til ei anna liste
- De er nyttige for blant annet brett og andre todimensjonale strukturer
- Listene ser ikke *innover* uten videre. Når vi sjekker lengden av ei liste, er det antall elementer i akkurat den lista som teller.

```
>>> liste = [[1,2,3],[4,5,6],[7,8,9]]
>>> print(liste[1][1])
5
>>> len(liste)
3
>>> len(liste[0])
3
```


:list


:list
:list
:list
:list
:list

Oppgave 9 (15 poeng)

a) (7 poeng)

Skriv en funksjon **trimZeros (a)** som tar inn en liste med heltall og returnerer en liste med heltall hvor alle (eventuelle) nuller i starten og slutten av listen er fjernet. Dersom det er nuller inne i listen (dvs som har andre tall foran og bak seg) skal disse *ikke* fjernes. Gitt en liste [0,0,1,2,0,3,0,0,4,0] som argument, skal funksjonen altså returnere listen [1,2,0,3,0,0,4]. Effektiviteten av løsningen blir ikke tillagt vekt, formålet er kun at koden skal gi ønsket resultat.

Oppgave 5, 2014

Skriv en funksjon som har en liste med tall som parameter og som returnerer en verdi av type boolean. Funksjonen skal sjekke om alle verdiene i listen er i stigende rekkefølge (sortert). Dersom alle verdiene er i sortert rekkefølge skal funksjonen returnere true, ellers skal funksjonen returnere false. Du kan anta at alle verdiene i listen er ulike. Funksjonen trenger altså ikke ta hensyn til eventuelle like verdier.

Oppgave 6, 2014

- a) Skriv en funksjon med liste av tall som parameter og som returnerer en verdi av type int (heltall). Dersom alle verdiene i listen er like, skal metoden returnere denne verdien. Dersom ikke alle verdiene er like, skal den returnere tallet -1. Du kan anta at listen inneholder minst en verdi.
- b) Dersom du kaller funksjonen fra a) med en ikke-tom liste av tall og får -1 tilbake, kan du da være sikker på at ikke alle tallene i listen du sendte inn var like? Begrunn svaret.
-

Oppgave 9, 2014

- Dersom du kaster 3 terninger, er det 6*6*6=216 mulige utfall av antall øyne på de tre terningene (1-1-1, 1-1-2, …, 6-6-6). Bare i 6 av disse 216 utfallene er det samme antall øyne på alle de tre terningene (1-1-1, 2-2-2 osv). Skriv de nødvendige programlinjene for å printe ut alle kombinasjoner av antall øyne på de tre terningene på terminalen. Print til slutt ut hvor mange kombinasjoner som hadde minst 2 like terninger. Merk at 1-1-2 og 1-2-1 i denne sammenhengen er to ulike kombinasjoner, slik at begge skal printes ut og telle med i antall kombinasjoner med minst 2 like terninger.
-