Kap 2: Løkker og lister

Ole Christian Lingjærde, Inst for Informatikk, UiO

26-30 August, 2019 (Del 1 av 2)

Tema denne uken

- Lære om datatypen boolean
- Lære å programmere med løkker
- Lære å bruke lister
- Se mange eksempler på Python-programmer

Før vi starter

Gjennom kurset kommer vil til å gi små quiz-oppgaver på forelesning.

Da gjelder det å huske:

- Oppgavene er laget for å teste *forståelse*.
- Oppgavene er oftest trivielle å løse på datamaskin.
- *Ikke bruk* datamaskin for å løse dem!!!

Litt om formatering av utskrift

- Vi kommer ofte til å skrive ting ut på skjerm fra programmer
- Viktig at du behersker "pen utskrift" (formatert utskrift)
- Mange måter å gjøre det på; du må kjenne til to
- Metode A: bruk av %-operator
- Metode B: bruk av f-strings

Viktig: Læreboka bruker den første metoden. På forelesningene kommer du til å møte begge.

Eksempler på formatert utskrift

Forestill deg at vi har disse fire variablene:

```
pris=95
vekt=10.556
alder1=18
alder2=21
```

og at vi ønsker å bruke variablene til å skrive ut følgende:

```
Prisen er 95 kroner
Vekten er 10.556000 kilo
Vekten er 10.56 kilo
Per er 18 og Anne er 21
```

Eksempler på formatert utskrift (forts.)

Metode A:

```
print("Prisen er %d kroner" % pris)
print("Vekten er %f kilo" % vekt)
print("Vekten er %5.2f kilo" % vekt)
print("Per er %d og Anne er %d" % (alder1, alder2))
```

Metode B:

```
print(f"Prisen er {pris} kroner")
print(f"Vekten er {vekt} kilo")
print(f"Vekten er {vekt:5.2f} kilo")
print(f"Per er {alder1} og Anne er {alder2}")
```

Quiz

Virker disse utskriftene og hva skriver de isåfall ut?

```
x = 0.5
print("Prisen er x kroner")
print(f"Prisen er x kroner")
print("Prisen er {x} kroner)
print(f"Prisen er {x} kroner")
print("Prisen er {x:5.2f} kroner")
print("Prisen er x:{5.2f} kroner")
print(f"Prisen er x:{5.2f} kroner")
print(f"Prisen er {x:5.2f} kroner")
```

Svar på Quiz

Virker disse utskriftene og hva skriver de isåfall ut?

```
x = 0.5
print("Prisen er x kroner")
# UTSKRIFT: Prisen er x kroner

print("Prisen er {x} kroner")
#

print("Prisen er %4.2f kroner" % x)
#

print(f"Prisen er %4.2f kroner" % x)
#

print(f"Prisen er {x} kroner")
#

print("Prisen er {x:4.2f} kroner")
#

print(f"Prisen er x:{4.2f} kroner")
#

print(f"Prisen er {x:4.2f} kroner")
#
```

Svar på Quiz

Virker disse utskriftene og hva skriver de isåfall ut?

```
x = 0.5
print("Prisen er x kroner")
# UTSKRIFT: Prisen er x kroner

print("Prisen er {x} kroner")
# UTSKRIFT: Prisen er {x} kroner

print("Prisen er %4.2f kroner" % x)
#

print(f"Prisen er %4.2f kroner" % x)
#

print(f"Prisen er {x} kroner")
#

print("Prisen er {x:4.2f} kroner")
#

print(f"Prisen er x:{4.2f} kroner")
#

print(f"Prisen er {x:4.2f} kroner")
#
```

Svar på Quiz

Virker disse utskriftene og hva skriver de isåfall ut?

```
x = 0.5
print("Prisen er x kroner")
# UTSKRIFT: Prisen er x kroner

print("Prisen er {x} kroner")
# UTSKRIFT: Prisen er {x} kroner

print("Prisen er %4.2f kroner" % x)
# UTSKRIFT: Prisen er 0.50 kroner

print(f"Prisen er %4.2f kroner" % x)
#

print(f"Prisen er {x} kroner")
#

print("Prisen er {x:4.2f} kroner")
#

print(f"Prisen er x:{4.2f} kroner")
#

print(f"Prisen er {x:4.2f} kroner")
#
```

Svar på Quiz

Virker disse utskriftene og hva skriver de isåfall ut?

```
x = 0.5
print("Prisen er x kroner")
# UTSKRIFT: Prisen er x kroner

print("Prisen er {x} kroner")
# UTSKRIFT: Prisen er {x} kroner

print("Prisen er %4.2f kroner" % x)
# UTSKRIFT: Prisen er 0.50 kroner

print(f"Prisen er %4.2f kroner" % x)
# UTSKRIFT: Prisen er 0.50 kroner

print(f"Prisen er {x} kroner")
#

print("Prisen er {x:4.2f} kroner")
#

print(f"Prisen er x:{4.2f} kroner")
#

print(f"Prisen er {x:4.2f} kroner")
#
```

Svar på Quiz

Virker disse utskriftene og hva skriver de isåfall ut?

```
x = 0.5
print("Prisen er x kroner")
# UTSKRIFT: Prisen er x kroner

print("Prisen er {x} kroner")
# UTSKRIFT: Prisen er {x} kroner

print("Prisen er %4.2f kroner" % x)
# UTSKRIFT: Prisen er 0.50 kroner

print(f"Prisen er %4.2f kroner" % x)
# UTSKRIFT: Prisen er 0.50 kroner

print(f"Prisen er {x} kroner")
# UTSKRIFT: Prisen er 0.5 kroner

print("Prisen er {x:4.2f} kroner")
#

print(f"Prisen er x:{4.2f} kroner")
#

print(f"Prisen er {x:4.2f} kroner")
#
```

Svar på Quiz

Virker disse utskriftene og hva skriver de isåfall ut?

```
x = 0.5
print("Prisen er x kroner")
# UTSKRIFT: Prisen er x kroner

print("Prisen er {x} kroner")
# UTSKRIFT: Prisen er {x} kroner

print("Prisen er %4.2f kroner" % x)
# UTSKRIFT: Prisen er 0.50 kroner

print(f"Prisen er %4.2f kroner" % x)
# UTSKRIFT: Prisen er 0.50 kroner

print(f"Prisen er {x} kroner")
# UTSKRIFT: Prisen er 0.5 kroner

print("Prisen er {x:4.2f} kroner")
# UTSKRIFT: Prisen er {x:4.2f} kroner

print(f"Prisen er x:{4.2f} kroner")
#

print(f"Prisen er {x:4.2f} kroner")
#
```

Svar på Quiz

Virker disse utskriftene og hva skriver de isåfall ut?

```
x = 0.5
print("Prisen er x kroner")
# UTSKRIFT: Prisen er x kroner

print("Prisen er {x} kroner")
# UTSKRIFT: Prisen er {x} kroner

print("Prisen er %4.2f kroner" % x)
# UTSKRIFT: Prisen er 0.50 kroner

print(f"Prisen er %4.2f kroner" % x)
# UTSKRIFT: Prisen er 0.50 kroner

print(f"Prisen er {x} kroner")
# UTSKRIFT: Prisen er 0.5 kroner

print("Prisen er {x:4.2f} kroner")
# UTSKRIFT: Prisen er {x:4.2f} kroner

print(f"Prisen er x:{4.2f} kroner")
# FEILMELDING: SyntaxError: invalid syntax

print(f"Prisen er {x:4.2f} kroner")
#
```

Svar på Quiz

Virker disse utskriftene og hva skriver de isåfall ut?

```
x = 0.5
print("Prisen er x kroner")
# UTSKRIFT: Prisen er x kroner

print("Prisen er {x} kroner")
# UTSKRIFT: Prisen er {x} kroner

print("Prisen er %4.2f kroner" % x)
# UTSKRIFT: Prisen er 0.50 kroner

print(f"Prisen er %4.2f kroner" % x)
# UTSKRIFT: Prisen er 0.50 kroner

print(f"Prisen er {x} kroner")
# UTSKRIFT: Prisen er 0.5 kroner

print("Prisen er {x:4.2f} kroner")
# UTSKRIFT: Prisen er {x:4.2f} kroner

print(f"Prisen er x:{4.2f} kroner")
# FEILMELDING: SyntaxError: invalid syntax

print(f"Prisen er {x:4.2f} kroner")
# UTSKRIFT: Prisen er 0.50 kroner
```

Lage en tabell

Anta at vi ønsker å lage en Celsius-Fahrenheit tabell:

```
-20.0  -4.0
-15.0   5.0
-10.0  14.0
 -5.0  23.0
  0.0  32.0
  5.0  41.0
 10.0  50.0
 15.0  59.0
 20.0  68.0
 25.0  77.0
 30.0  86.0
 35.0  95.0
 40.0 104.0
```

Tallene i venstre kolonne er Celcius-grader, tallene i høyre kolonne er tilsvarende Fahrenheit-grader. Vi skal skrive et Python-program som lager en slik tabell.

Enkel (naiv) løsning

Det er lett å lage en enkelt linje i tabellen:

```
C = -20
F = 9.0/5 * C + 32
print("%3.1f  %3.1f" % (C, F))
```

Vi kan bruke metoden over mange ganger:

```
C = -20;  F = 9.0/5*C + 32;  print("%3.1f  %3.1f" % (C, F))
C = -15;  F = 9.0/5*C + 32;  print("%3.1f  %3.1f" % (C, F))
...
C =  35;  F = 9.0/5*C + 32;  print("%3.1f  %3.1f" % (C, F))
C =  40;  F = 9.0/5*C + 32;  print("%3.1f  %3.1f" % (C, F))
```

Kommentarer:

- Tungvint hvis tabellen som skal lages er stor!
- Lett å programmere feil (f.eks. hoppe over tabellrad)
- Tidkrevende å sjekke at det ikke er noen feil
- Bedre alternativ: lag en programløkke!

Programløkker i Python

Vi kan løse oppgaven over med to forskjellige løkketyper:

While-løkke:

```
C = -20
while C <= 40:
    F = (9.0/5)*C + 32
    print("%3.1f  %3.1f" % (C, F))
    C = C + 5
```

For-løkke:

```
for C in range(-20,45,5):
    F = (9.0/5)*C + 32
    print("%3.1f  %3.1f" % (C, F))
```

(Vi kommer tilbake senere til detaljer i programmene ovenfor)

While-løkker

While-løkker brukes for å repetere en samling programsetninger inntil en gitt logisk betingelse ikke lenger holder.

```
while logiskBetingelse:
    <setning 1>
    <setning 2>
    ...
    <setning n>
```

Viktig å merke seg:

- Innmaten i løkka (= setningene som skal utføres flere ganger) må alle ha samme innrykk
- Første setning uten innrykk markerer at while-løkken er ferdig.

Eksempel: while-løkke for tabellen over

```
# Sett start-verdien til C:
C = -20

# Utfør løkke-innmaten hvis C <= 40:
while C <= 40:
  # Regn ut F:
  F = (9.0/5)*C + 32

  # Skriv ut C og F:
  print("%3.1f  %3.1f" % (C, F))

  # Øk C med 5:
  C = C + 5

  # Gå til starten av løkken

# Første setning etter while-løkken:
print("Ferdig")
```

Programflyten

```
C = -20
while C <= 40:
  F = (9.0/5)*C + 32
  print("%3.1f  %3.1f" % (C, F))
  C = C + 5
print("Ferdig")
```

Instruksjonene som utføres:

```
C = -20

C <= 40?                              ### True
F = (9.0/5)*C + 32                    ### F beregnes til -4
print("%3.1f  %3.1f" % (C, F))        ### Skriv ut "-20.0, -4.0"
C = C + 5

C <= 40?                              ### True
F = (9.0/5)*C + 32                    ### F beregnes til 5
print("%3.1f  %3.1f" % (C, F))        ### Skriv ut "-15.0, 5.0"
C = C + 5

C <= 40?                              ### True
(osv ...)
```

Logiske uttrykk (= Boolske uttrykk)

- Uttrykket 2+3 gir et heltall som svar (5)
- Uttrykket 1.5 + 4.0 gir et desimaltall som svar (5.5)
- Uttrykket 2 <= 3 gir en logisk verdi som svar (True)

Eksempel:

```
# Vi setter C lik 18:
C = 18

# Logiske uttrykk:
C == 18  # Er C lik 18? (Verdi: True)
C == 19  # Er C lik 19? (Verdi: False)
C != 20  # Er C ulik 20? (Verdi: True)
C >  20  # Er C større enn 20? (Verdi: False)
C >= 20  # Er C større enn eller lik 20? (Verdi: False)
C <  20  # Er C mindre enn 20? (Verdi: True)
C <= 20  # Er C mindre enn eller lik 20? (Verdi: True)
```

Kombinasjon av logiske uttrykk med **and**

```
x = 0; y = 0
while x < 10 and y < 10:
    print("x+y = %d" % (x+y))
    x = x + 1
    y = 2 * y + x
```

Resultat:

```
WHILE-TEST:       UTSKRIFT:   OPPDATERING x:   OPPDATERING y:
0<10 and 0<10     x+y = 0     x = 0+1 = 1      y = 2*0+1 = 1
1<10 and 1<10     x+y = 2     x = 1+1 = 2      y = 2*1+2 = 4
2<10 and 4<10     x+y = 6     x = 2+1 = 3      y = 2*4+3 = 11
3<10 and 11<10    (slutt)
```

Kombinasjon av logiske uttrykk med **or**

```
x = 0; y = 0
while x < 10 or y < 10:
    print("x+y = %d" % (x+y))
    x = x + 1
    y = 2 * y + x
```

Resultat:

```
WHILE-TEST:          UTSKRIFT:   OPPDATERING x:   OPPDATERING y:
0<10   or 0<10       x+y = 0     x = 0+1 = 1      y = 2*0+1 = 1
1<10   or 1<10       x+y = 2     x = 1+1 = 2      y = 2*1+2 = 4
2<10   or 4<10       x+y = 6     x = 2+1 = 3      y = 2*4+3 = 11
3<10   or 11<10      x+y = 14    x = 3+1 = 4      y = 2*11+4 = 26
....
9<10   or 1013<10    x+y = 1022  x = 9+1 = 10     y = 2*1013+10 = 2036
10<10  or 2036<10    (slutt)
```

Flere eksempler på bruk av **and**/**or**

```
>>> x = 0;  y = 1.2

>>> x >= 0 and y < 1
False

>>> x >= 0 or y < 1
True

>>> x > 0 or y > 1
True

>>> x > 0 or not y > 1
False

>>> -1 < x <= 0
True

>>> not (x > 0 or y > 0)
False
```

Quiz

Hvilke logiske verdier gir disse uttrykkene?

```
x = 1; y = 2

x > y
#

(x != y) and (x < y)
#

not (x < y and y <= x)
#

(x < y and x > y) or (x != y)
#

y**y > x + 2*y
#
```

Svar på Quiz

Hvilke logiske verdier gir disse uttrykkene?

```
x = 1; y = 2

x > y
# Svar: False

(x != y) and (x < y)
#

not (x < y and y <= x)
#

(x < y and x > y) or (x != y)
#

y**y > x + 2*y
#
```

Svar på Quiz

Hvilke logiske verdier gir disse uttrykkene?

```
x = 1; y = 2

x > y
# Svar: False

(x != y) and (x < y)
# Svar: True      (True and True --> True)

not (x < y and y <= x)
#

(x < y and x > y) or (x != y)
#

y**y > x + 2*y
#
```

Svar på Quiz

Hvilke logiske verdier gir disse uttrykkene?

```
x = 1; y = 2

x > y
# Svar: False

(x != y) and (x < y)
# Svar: True

not (x < y and y <= x)
# Svar: True      (not (True and False) --> not (False) --> True)

(x < y and x > y) or (x != y)
#

y**y > x + 2*y
#
```

Svar på Quiz

Hvilke logiske verdier gir disse uttrykkene?

```
x = 1; y = 2

x > y
# Svar: False

(x != y) and (x < y)
# Svar: True

not (x < y and y <= x)
# Svar: True

(x < y and x > y) or (x != y)
# Svar: True      (True and False) or (True) --> True

y**y > x + 2*y
#
```

Svar på Quiz

Hvilke logiske verdier gir disse uttrykkene?

```
x = 1; y = 2

x > y
# Svar: False

(x != y) and (x < y)
# Svar: True

not (x < y and y <= x)
# Svar: True

(x < y and x > y) or (x != y)
# Svar: True

y**y > x + 2*y
# Svar: False      (4 > 1 + 2*2 --> False)
```

For-løkker

For-løkker er et nyttig alternativ til while-løkker når man ønsker å gå gjennom en definert liste av verdier eller elementer:

```
for e in <liste av elementer>:
    <setninger som bruker verdien e>
```

Merk:

- Innmaten i løkka må ha innrykk (analogt som for while-løkker)
- Vi trenger å lære hvordan man lager *lister* i Python

Lister

Hittil har hver variabel hatt plass til ett tall eller én string:

```
C = -10
x = 2.255
s = 'Hello'
```

Vi kan også lage dataobjekter og variable som inneholder flere verdier:

```
C = [-10, -5, 0, 5, 10]
x = [2.255, 3.634, 6.4]
s = ['Hello', 'my', 'friend']
u = [3, 3.14, 'pi']
```

Få tak i enkeltelementer i lister

Vi kan få tak i enkeltverdier i en liste.

Til dette brukes elementenes **indeks** 0, 1, 2, ...

```
>>> a = [3.14, 3.1415926, 999999]

>>> a[0]
3.14

>>> a[1]
3.1415926

>>> a[2]
999999

>>> a[3]
Traceback (most recent call last):

File "<ipython-input-62-f75b6be7d8e3>", line 1, in <module>
    a[3]

IndexError: list index out of range

>>> len(a)  # Finn lengden til listen
3
```

Endre enkeltelementer i lister

Å aksessere et listeelement (f.eks. `a[0]`) endrer ikke listens innhold.

Vi kan også endre enkeltelementer i en liste.

```
>>> a = [3.14, 3.1415926, 999999]

>>> a
[3.14, 3.1415926, 999999]

>>> a[1] = 0.1
>>> a
[3.14, 0.1, 999999]

>>> a[3] = 7
Traceback (most recent call last):

File "<ipython-input-68-7cddb6269663>", line 1, in <module>
    a[3] = 7

IndexError: list assignment index out of range
>>> a.append(7)
>>> a
[3.14, 0.1, 999999, 7]
```

Fire måter å endre lister på

Fire nyttige listeoperasjoner: append, extend, insert, delete

```
>>> a = [-10, -5, 0, 5, 10]

>>> a.append(99)          # Legg til et nytt element på slutten
>>> a
[-10, -5, 0, 5, 10, 99]

>>> a = a + [100, 200]    # Slå sammen (=konkatenér) to lister
>>> a
[-10, -5, 0, 5, 10, 99, 100, 200]

>>> a.insert(2, -99)      # Sett inn -99 slik at den får indeks 2
>>> a
[-10, -5, -99, 0, 5, 10, 99, 100, 200]

>>> del a[2]              # Fjern elementet med indeks 2
>>> a
[-10, -5, 0, 5, 10, 99, 100, 200]

>>> del a[0]              # Fjern elementet med indeks 0
>>> a
[-5, 0, 5, 10, 99, 100, 200]
```

Søke etter elementer i liste

```
>>> a = [-10, -5, 0, 5, 10]

>>> a.index(10)  # Hvor ligger verdien 10?
4

>>> 5 in a       # Finnes verdien 5 i listen?
True

>>> b = [1, 2, 1]
>>> b.index(1)   # Bare treff nr 1 rapporteres
0

>>> b.count(1)   # Hvor mange ganger fins verdien 1 i listen?
2

>>> k1 = b.index(1)          # Treff nr 1
>>> k2 = b.index(1, k1+1)    # Treff nr 2
>>> k1, k2
(0, 2)
```

Negative indekser

```
>>> a = [-10, -5, 0, 5, 10]

>>> a[-1]  # Siste element
10

>>> a[-2]  # Nest siste element
5

>>> somelist = ['book.tex', 'book.log', 'book.pdf']
>>> texfile, logfile, pdf = somelist  # assign directly to variables
>>> texfile
'book.tex'
>>> logfile
'book.log'
>>> pdf
'book.pdf'
```

Løpe gjennom en liste med en for-løkke

Vi kan bruke en for-løkke til å løpe gjennom alle elementene i en liste og prosessere dem:

```
CListe = [-20, -15, -10, 5, 0]
for C in CListe:
    F = (9.0/5) * C + 32
    print("%3.1f  %3.1f" % (C, F))
```

Resultat:

```
-20.0  -4.0
-15.0  5.0
-10.0  14.0
5.0  41.0
0.0  32.0
```

Simuler en for-løkke for hånd

```
CListe = [-20, -15, -10, 5, 0]
for C in CListe:
    F = (9.0/5) * C + 32
    print("%3.1f  %3.1f" % (C, F))
```

Simulering for hånd:

- Første gjennomløp: `C` er -20 og innmaten utføres
- Andre gjennomløp: `C` er -15 og innmaten utføres
- ....
- Femte gjennomløp: `C` er 0 og innmaten utføres for siste gang

Tilbake til eksemplet med temperaturtabell

```
Cdegrees = [-20, -15, -10, -5, 0, 5, 10, 15,
             20, 25, 30, 35, 40]
for C in Cdegrees:
    F = (9.0/5)*C + 32
    print("%3.1f  %3.1f" % (C, F))
```

Resultat:

```
-20  -4.0
-15   5.0
-10  14.0
 -5  23.0
  0  32.0
  ......
 35  95.0
 40 104.0
```

En for-løkke kan alltid oversettes til en while-løkke

Det kan vises at en for-løkke

```
for element in somelist:
    # process element
```

alltid kan oversettes til en while-løkke

```
index = 0
while index < len(somelist):
    element = somelist[index]
    # process element
    index += 1
```

Det omvendte er ikke tilfelle (med pen programmering).

Plenumsoppgaver

Oppgaver i plenum:

- 1.3 (seconds2years.py, side 43)
- 1.4 (lengthconversion.py, side 43)
- 1.12 (egg.py, side 46)
- 2.1 (f2ctablewhile.py, side 82)
- 2.3 (primes.py, side 82)
- 2.4 (odd.py, side 82)
- 2.8 (balltable1.py, side 83)

Oppgave 1.3

Exercise 1.3: Derive and compute a formula.
Can a newborn baby in Norway expect to live for one billion (10**9) seconds? Write a Python program for doing arithmetics to answer the question.
Filename:seconds2years

Oppgave 1.3: Løsningsskisse

- seconds = 10**9
- minutes = seconds/60
- hours = minutes/60
- days = hours/24
- years = days/365

Oppgave 1.3: Python-kode

seconds2years.py

```
seconds = 10**9
minutes = seconds/60.0
hours = minutes/60.0
days = hours/24.0
years = days/365.0
print("%d seconds equals %g years" % (seconds, years))
```

Resultat når programmet kjøres:

```
> python seconds2years.py
1000000000 seconds equals 31.7098 years
```

Oppgave 1.4

Exercise 1.4: Convert from meters to British length units.
Make a program where you set a length given in meters and then compute and write out the corresponding length measured in inches, in feet, in yards, and in miles.
Use that one inch is 2.54 cm, one foot is 12 inches, one yard is 3 feet, and one British mile is 1760 yards.
For verification, a length of 640 meters corresponds to 25196.85 inches, 2099.74 feet, 699.91 yards, or 0.3977 miles.
Filename:lengthconversion.

Oppgave 1.4: Løsningsskisse

- 1 inch = 2.54 cm = 0.0254 m –> 1 m = 1/0.0254 inch
- **Thus:** inches = meters/0.0254
- 1 foot = 12 inch –> 1 inch = 1/12 foot
- **Thus:** feet = inches/12
- 1 yard = 3 feet –> 1 foot = 1/3 yard
- **Thus:** yards = feet/3
- 1 British mile = 1760 yards –> 1 yard = 1/1760 British mile
- **Thus:** British mile = yards/1760

Oppgave 1.4: Python-kode

lengthconversion.py

```
meters = 640
inches = meters/0.0254
feet = inches/12.0
yards = feet/3.0
bm = yards/1760.0
print("%d meters equals %g British miles" % (meters, bm))
```

Resultat når programmet kjøres:

```
> python lengthconversion.py
640 meters equals 0.397678 British miles
```