Forelesning IN1900 – 29 August 2022

Ole Christian Lingjærde
Institutt for informatikk, Universitetet i Oslo

Uke: 29 August – 4 September, 2022

Kort om min bakgrunn

- PhD i computer science/matematikk fra Ifi, UiO
- Professor i maskinlæring og kunstig intelligens
- Forskerstilling ved Oslo Universitetssykehus hvor vi utvikler nye diagnoseverktøy for kreft basert på DNA-data
- Undervist i programmering i ca 15 år

Tema denne uken

I programmering trenger vi ofte å utføre samme (eller nesten samme) operasjon mange ganger. Denne uken skal vi lære en enkel måte å gjøre dette på, og trene mye på det.

Mer konkret skal vi lære:

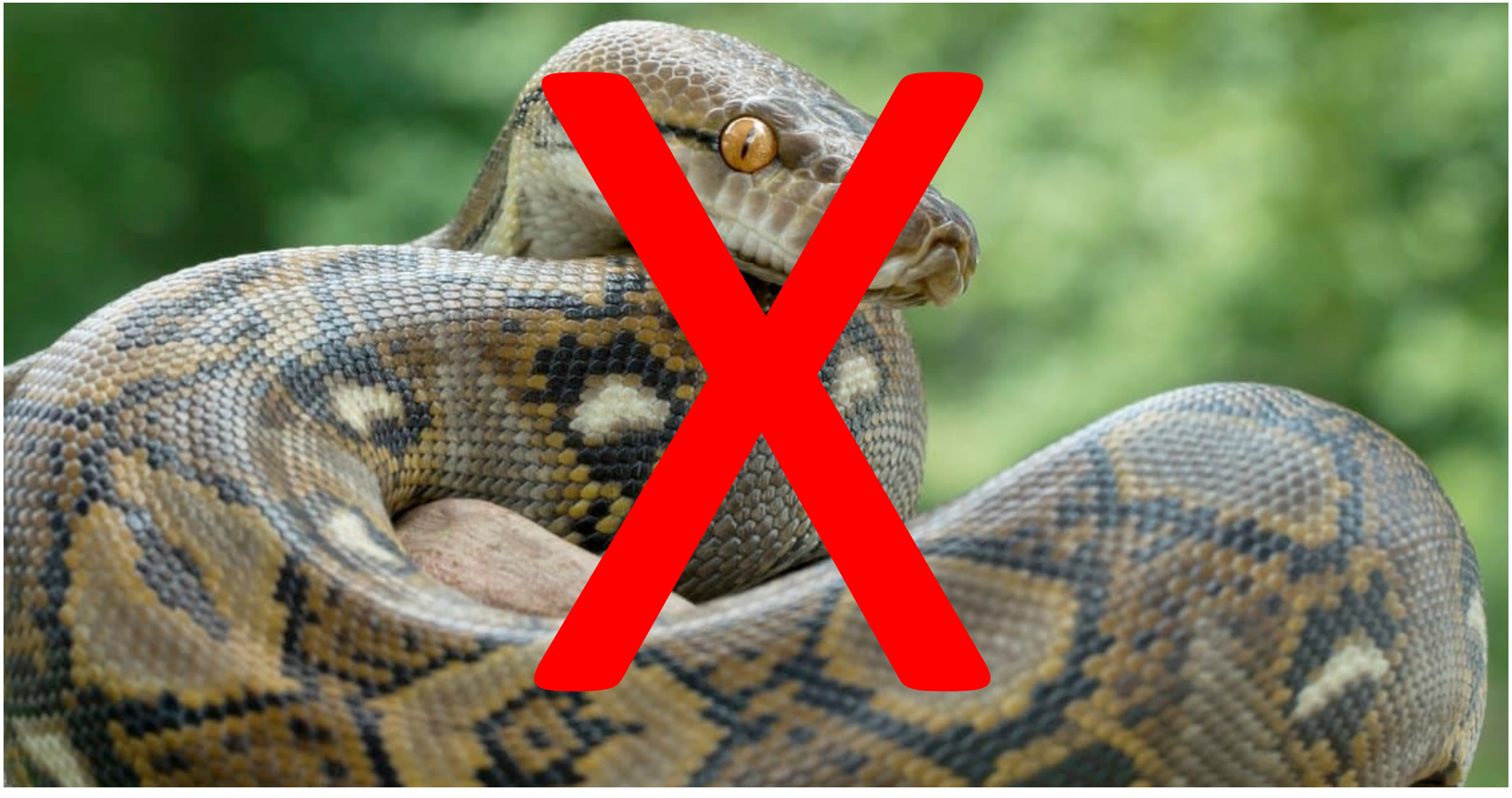
- *Programmering med løkker*
- *Programmering med lister*

Quiz-oppgaver

- Kommer til å dukke opp på de fleste forelesninger
- Laget for å teste forståelse
- Oftest trivielle å løse med datamaskin
- Ikke bruk datamaskin når du løser dem!

Hvorfor Python?

Ikke slangen....



... men komikergruppen Monthy Python

Guido van Rossum (Pythons skaper, 1956-)

Pythons historie

Startet opprinnelig som et hobbyprosjekt som Guido startet julen 1989 mens han jobbet i Google, for å ha noe å gjøre i juleferien.

Python 0.9.0:	Første versjon (utgitt 1991)
Python 1.0:	Utvidelse av første versjon (1994)
Python 2.0:	Støtte for unicode, list comprehension++ (2000)
Python 3.0:	Kraftig opprydding (2008)
Python 3.9:	Dagens versjon (2020)

Noen fakta om Python

- Offisielt språk i Google sammen med C++ og Java
- Åpen kildekode og helt gratis
- Et av de mest populære språkene for forskning og utvikling innen kunstig intelligens og big data
- Mange varianter for spesielle formål, f.eks. CPython, Jython, IronPython, MicroPython, ...
- Mer populært enn fransk i en undersøkelse i engelske barneskoler (hvor foreldrene ble spurt)

I dag: helt grunnleggende om Python

- Lære å skrive ut tekst og tall på skjerm (så det ser pent ut)
- Lære å regne ut en enkel tabell og skrive ut resultatet på skjerm
- Lære å utføre samme operasjon mange ganger
- Logiske uttrykk (True og False)
- Hvordan samle mange verdier i en liste

Utskrift på skjerm

Vi må kjenne til tre metoder:

a) Enkel utskrift uten formatering

b) Formatert utskrift med %-operator

c) Formatert utskrift med f-strings

NB: Oppgavene i den gamle læreboka bruker %-operator. Den nye læreboka bruker f-strings. Vi bruker hovedsakelig f-strings på forelesningene.

Utskrift uten formatering

```
print("Dette er en tekst")
print("")
print(3)
print(3.1)
print(1/3)
```

```
Dette er en tekst

3
3.1
0.3333333333333333
```

Formatert utskrift med %-operator

```
høyde = 180
vekt = 75.5
adresse = "Andeby"
print("Høyden er %d centimeter" % høyde)
print("Vekten er %f kilo" % vekt)
print("Vekten er %5.2f kilo" % vekt)
print("Hun bor i %s." % adresse)
print("Hun bor i %10s." % adresse)
```

```
Høyden er 180 centimeter
Vekten er 75.5 kilo
Vekten er 75.50 kilo
Hun bor i Andeby.
Hun bor i     Andeby.
Hun bor i Andeby    .
```

Formatert utskrift med f-strings

```
høyde = 180
vekt = 75.5
adresse = "Andeby"

print(f"Høyden er {høyde} centimeter")
print(f"Vekten er {vekt} kilo")
print(f"Vekten er {vekt:5.2f} kilo")
print(f"Hun bor i {adresse}.")
print(f"Hun bor i {adresse:>10s}.")
print(f"Hun bor i {adresse:<10s}.")
```

```
Høyden er 180 centimeter
Vekten er 75.5 kilo
Vekten er 75.50 kilo
Hun bor i Andeby.
Hun bor i     Andeby.
Hun bor i Andeby    .
```

Quiz

Anta at $x = 0.5$. Hva skriver hver av linjene ut?

```
print("Prisen er x kroner")
#

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

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

print(f"Prisen er x:{5.2f} kroner")
#

print(f"Prisen er {x:5.2f} kroner")
#

print("Prisen er %5.2f kroner" % x)
#
```

Quiz

Anta at $x = 0.5$. Hva skriver hver av linjene ut?

```
print("Prisen er x kroner")
# Prisen er x kroner

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

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

print(f"Prisen er x:{5.2f} kroner")
#

print(f"Prisen er {x:5.2f} kroner")
#

print("Prisen er %5.2f kroner" % x)
#
```

Quiz

Anta at $x = 0.5$. Hva skriver hver av linjene ut?

```
print("Prisen er x kroner")
# Prisen er x kroner

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

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

print(f"Prisen er x:{5.2f} kroner")
#

print(f"Prisen er {x:5.2f} kroner")
#

print("Prisen er %5.2f kroner" % x)
#
```

Quiz

Anta at $x = 0.5$. Hva skriver hver av linjene ut?

```
print("Prisen er x kroner")
# Prisen er x kroner

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

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

print(f"Prisen er x:{5.2f} kroner")
#

print(f"Prisen er {x:5.2f} kroner")
#

print("Prisen er %5.2f kroner" % x)
#
```

Quiz

Anta at $x = 0.5$. Hva skriver hver av linjene ut?

```
print("Prisen er x kroner")
# Prisen er x kroner

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

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

print(f"Prisen er x:{5.2f} kroner")
# FEILMELDING (SyntaxError: f-string: invalid syntax)

print(f"Prisen er {x:5.2f} kroner")
#

print("Prisen er %5.2f kroner" % x)
#
```

Quiz

Anta at $x = 0.5$. Hva skriver hver av linjene ut?

```
print("Prisen er x kroner")
# Prisen er x kroner

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

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

print(f"Prisen er x:{5.2f} kroner")
# FEILMELDING (SyntaxError: f-string: invalid syntax)

print(f"Prisen er {x:5.2f} kroner")
# Prisen er 0.50 kroner

print("Prisen er %5.2f kroner" % x)
#
```

Quiz

Anta at $x = 0.5$. Hva skriver hver av linjene ut?

```
print("Prisen er x kroner")
# Prisen er x kroner

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

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

print(f"Prisen er x:{5.2f} kroner")
# FEILMELDING (SyntaxError: f-string: invalid syntax)

print(f"Prisen er {x:5.2f} kroner")
# Prisen er 0.50 kroner

print("Prisen er %5.2f kroner" % x)
# Prisen er 0.50 kroner
```

Lage en tabell

Vi ønsker å lage et program som skriver ut denne tabellen:

```
-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
```

Venstre kolonne er Celcius-grader (C), høyre kolonne er tilsvarende Fahrenheit-grader (F). Formelen er $F = (9/5)C + 32$.

Enkel (naiv) løsning

Det er lett å lage en enkelt linje i tabellen:

```
C = -20
F = 9/5 * C + 32
print(f"{C:3.1f}  {F:3.1f}")
```

Vi kan gjøre det samme mange ganger:

```
C = -20;  F = 9/5*C + 32;  print(f"{C:3.1f}  {F:3.1f}")
C = -15;  F = 9/5*C + 32;  print(f"{C:3.1f}  {F:3.1f}")
C = -10;  F = 9/5*C + 32;  print(f"{C:3.1f}  {F:3.1f}")
...
C =  40;  F = 9/5*C + 32;  print(f"{C:3.1f}  {F:3.1f}")
```

Bedre alternativ: løkke

Løsningen over er ikke spesielt god:

- Slitsomt hvis tabellen har mange rader (f.eks. 1000)
- Lett å programmere feil, f.eks. hoppe over en tabellrad
- Utnytter ikke at samme operasjon utføres hver gang

Bedre alternativ: lage en programløkke.

Programløkker i Python

To forskjellige løkketyper i Python, begge er viktige:

While-løkke:

Generell løkketype. Kan *alltid* benyttes.

For-løkke:

Kan ofte (men ikke alltid) brukes. Enklere kode.

(Noen programspråk har flere løkketyper)

Eksempel på while-løkke

```
while a < 50:
  print(a)
  a = a * 2
```

Løkke-hode

Løkke-kropp

Innrykk (indentation)

Eksempel på while-løkke

```
while a < 50:
  print(a)
  a = a * 2
```

Løkkehode

Løkkekropp

Innrykk (indentation)

Mening: så lenge variabelen a sin verdi er < 50 så skrives verdien til a ut og a dobles.

Eksempel på while-løkke

```
a = 1
while a < 50:
   print(a)
   a = a * 2
```

```
1
2
4
8
16
32
```

Fakta om while-løkker

De repeterer programsetninger helt til en gitt logisk betingelse ikke lenger er sann.

De har generelt formen:

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

Alle setningene i løkkekroppen må ha innrykk.

While-løkke for temperatur-tabellen

```
C = -20

while C <= 40:
  F = (9/5)*C + 32
  print(f"{C:3.1f}  {F:3.1f}")
  C = C + 5

print("Når vi kommer hit er løkka slutt")
print(f"Nå er C = {C}")
```

While-løkke for temperatur-tabellen

```
-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
Når vi kommer hit er løkka slutt
Nå er C = 45
```

Hva skjer egentlig når løkka kjører?

```
C = -20
while C <= 40:
  F = (9/5)*C + 32
  print(f"{C:3.1f}  {F:3.1f}")
  C = C + 5
```

Når løkka over kjører skjer dette:

```
C = -20

C <= 40                                  # True
F = (9/5)*C + 32                         # F beregnes til -4
print(f"{C:3.1f}     {F:3.1f}")          # Skriv ut "-20.0, -4.0"
C = C + 5

C <= 40                                  # True
F = (9/5)*C + 32                         # F beregnes til 5
print(f"{C:3.1f}     {F:3.1f}")          # Skriv ut "-15.0, 5.0"
C = C + 5

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

Logiske uttrykk

Et *logisk uttrykk* (= Boolsk uttrykk) er noe Python kan regne ut og som gir True eller False som resultat:

```
C = 18      # Ett likhetstegn betyr tilordning
C == 18     # Er C lik 18? (True)
C == 19     # Er C lik 19? (False)
C != 20     # Er C forskjellig fra 20 (True)
C > 20      # Er C større en 20? (False)
C >= 20     # Er C større enn eller lik 20? (False)
C < 20      # Er C mindre enn 20? (True)
C <= 20     # Er C mindre enn eller lik 20? (True)
```

Operatoren **and**

```
True and True
True and False
False and True
False and False
```

```
True
False
False
False
```

Operatoren **and**

Eksempel på bruk av **and**:

```
x = 0
y = 0
while x < 3 and y < 3:
    print(f"x+y = {x+y:d}")
    x = x + 1
    y = 2*y + x
```

x	y	x<3 and y<3	Utskrift	Oppdatering x	Oppdatering y

Operatoren **and**

Eksempel på bruk av **and**:

```
x = 0
y = 0
while x < 3 and y < 3:
    print(f"x+y = {x+y:d}")
    x = x + 1
    y = 2*y + x
```

x	y	x<3 and y<3	Utskrift	Oppdatering x	Oppdatering y
0	0	True	x+y = 0	x = 0 + 1	y = 2*0 + 1

Operatoren **and**

Eksempel på bruk av **and**:

```
x = 0
y = 0
while x < 3 and y < 3:
    print(f"x+y = {x+y:d}")
    x = x + 1
    y = 2*y + x
```

x	y	x<3 and y<3	Utskrift	Oppdatering x	Oppdatering y
0	0	True	x+y = 0	x = 0 + 1	y = 2*0 + 1
1	1	True	x+y = 2	x = 1 + 1	y = 2*1 + 1

Operatoren **and**

Eksempel på bruk av **and**:

```
x = 0
y = 0
while x < 3 and y < 3:
    print(f"x+y = {x+y:d}")
    x = x + 1
    y = 2*y + x
```

x	y	x<3 and y<3	Utskrift	Oppdatering x	Oppdatering y
0	0	True	x+y = 0	x = 0 + 1	y = 2*0 + 1
1	1	True	x+y = 2	x = 1 + 1	y = 2*1 + 1
2	4	False			

Operatoren **or**

```
True or True
True or False
False or True
False or False
```

```
True
True
True
False
```

Quiz

Hva skriver dette programmet ut?

```
x = 0
y = 0
while x<=2 or y<2:
    print(f"x+y = {x+y:d}")
    x = x + 1
    y = 2*x + 2
```

Quiz

Hva skriver dette programmet ut?

```
x = 0
y = 0
while x<=2 or y<2:
    print(f"x+y = {x+y:d}")
    x = x + 1
    y = 2*x + 2
```

x	y	x<=2 or y<2	Utskrift	Oppdatering x	Oppdatering y

Quiz

Hva skriver dette programmet ut?

```
x = 0
y = 0
while x<=2 or y<2:
    print(f"x+y = {x+y:d}")
    x = x + 1
    y = 2*x + 2
```

x	y	x<=2 or y<2	Utskrift	Oppdatering x	Oppdatering y
0	0	True	x+y = 0	x = 0 + 1	y = 2*1 + 2

Quiz

Hva skriver dette programmet ut?

```
x = 0
y = 0
while x<=2 or y<2:
    print(f"x+y = {x+y:d}")
    x = x + 1
    y = 2*x + 2
```

x	y	x<=2 or y<2	Utskrift	Oppdatering x	Oppdatering y
0	0	True	x+y = 0	x = 0 + 1	y = 2*1 + 2
1	4	True	x+y = 5	x = 1 + 1	y = 2*2 + 2

Quiz

Hva skriver dette programmet ut?

```
x = 0
y = 0
while x<=2 or y<2:
    print(f"x+y = {x+y:d}")
    x = x + 1
    y = 2*x + 2
```

x	y	x<=2 or y<2	Utskrift	Oppdatering x	Oppdatering y
0	0	True	x+y = 0	x = 0 + 1	y = 2*1 + 2
1	4	True	x+y = 5	x = 1 + 1	y = 2*2 + 2
2	6	True	x+y = 8	x = 2 + 1	y = 2*3 + 2

Quiz

Hva skriver dette programmet ut?

```
x = 0
y = 0
while x<=2 or y<2:
    print(f"x+y = {x+y:d}")
    x = x + 1
    y = 2*x + 2
```

x	y	x<=2 or y<2	Utskrift	Oppdatering x	Oppdatering y
0	0	True	x+y = 0	x = 0 + 1	y = 2*1 + 2
1	4	True	x+y = 5	x = 1 + 1	y = 2*2 + 2
2	6	True	x+y = 8	x = 2 + 1	y = 2*3 + 2
3	8	False			

Quiz

Hva skriver dette programmet ut?

```
x = 0
y = 0
while x<=2 or y<2:
    print(f"x+y = {x+y:d}")
    x = x + 1
    y = 2*x + 2
```

x	y	x<=2 or y<2	Utskrift	Oppdatering x	Oppdatering y
0	0	True	x+y = 0	x = 0 + 1	y = 2*1 + 2
1	4	True	x+y = 5	x = 1 + 1	y = 2*2 + 2
2	6	True	x+y = 8	x = 2 + 1	y = 2*3 + 2
3	8	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
#
```

Quiz

Hvilke logiske verdier gir disse uttrykkene?

```
x = 1; y = 2

x > y
# 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
#
```

Quiz

Hvilke logiske verdier gir disse uttrykkene?

```
x = 1; y = 2

x > y
# False

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

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

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

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

Quiz

Hvilke logiske verdier gir disse uttrykkene?

```
x = 1; y = 2

x > y
# False

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

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

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

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

Quiz

Hvilke logiske verdier gir disse uttrykkene?

```
x = 1; y = 2

x > y
# False

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

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

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

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

Quiz

Hvilke logiske verdier gir disse uttrykkene?

```
x = 1; y = 2

x > y
# False

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

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

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

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

For-løkker

For-løkker er et alternativ til while-løkker. Generell form:

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

Merk:

- Alle setninger i løkkekroppen må ha innrykk
- For-løkken trenger en *liste* (som vi skal lære om nå)

Lister

Frem til nå har en variabel hatt plass til en verdi:

```
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 enkeltverdier i en liste

Vi kan lett få tak i enkeltverdier i en liste ved å bruke plasseringen (indeksen):

```
a = [3.14, 3.1415926, 999999]
print(len(a))
print(a[0])
print(a[1])
print(a[2])
```

```
3
3.14
3.1415926
999999
```

Endre verdier i en liste

Vi kan endre enkeltelementer i en liste:

```
a = [3.14, 3.1415926, 999999]
a[1] = 0.1
print(a)
```

```
[3.14, 0.1, 999999]
```

Fire nyttige liste-operasjoner

Nyttige operasjoner: *append, extend, insert, delete*

```
a = [-10, -5, 0, 5, 10]
a.append(99)
print(a)

a = a + [100, 200]
print(a)

a.insert(2, -99)
print(a)

del a[2]
print(a)
```

```
[-10, -5, 0, 5, 10, 99]
[-10, -5, 0, 5, 10, 99, 100, 200]
[-10, -5, -99, 0, 5, 10, 99, 100, 200]
[-10, -5, 0, 5, 10, 99, 100, 200]
```

Søke etter verdier i en liste

```
a = [-10, -5, 0, 5, 10]
print(a.index(10))          # Hvor ligger verdien 10?
print(5 in a)               # Er verdien 5 i listen?

b = [1, 2, 1]
print(b.index(1))           # Posisjon til verdien 1
print(b.count(1))           # Antall forekomster av 1

k1 = b.index(1)             # Første forekomst av 1
k2 = b.index(1, k1+1)       # Andre forekomst av 1
print(k1, k2)
```

```
4
True
0
2
0 2
```

Negative indekser

Vi kan indeksere en liste fra høyre (siste element):

```
a = [-10, -5, 0, 5, 10]
print(a[-1])
print(a[-2])

b = ['eple', 'appelsin', 'ananas']
print(b[-1], b[-2], b[-3])
```

```
10
5
ananas appelsin eple
```

Løpe gjennom en liste med en for-løkke

Vi kan bruke en for-løkke til å løpe gjennom alle verdiene i en liste:

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

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

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(f"{C:3.1f}  {F:3.1f}")
```

```
-20.0  -4.0
-15.0  5.0
-10.0  14.0
-5.0  23.0
.....  (hopper over noen linjer).....
35.0  95.0
40.0  104.0
```

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

Vi kan oversette for-løkker til while-løkker.
Eksempel:

```
minliste = [0,2,4]
for x in minliste:
  print(x**2)
```

Denne kan også skrives slik:

```
minliste = [0,2,4]
k = 0
while k < len(minliste):
  x = minliste[k]
  print(x**2)
  k += 1
```

Motsatt vei er ikke alltid mulig (med pen programmering).