NTNU
Kunnskap for en bedre verden

Kunnskap for en bedre verden

TDT4110 Informasjonsteknologi grunnkurs:
Tema: Løkker/Sløyfer
Utgave 3: Kap. 4
Utgave 2: Kap. 5

Terje Rydland - IDI/NTNU

Læringsmål og pensum

- Mål
  - Lære om begrepet løkker (eller sløyfer hvis du vil)
  - Lære om bruk av while-løkke
  - Lære om bruk av for-løkke
- Pensum
  - Starting out with Pyton:

  Chapter 4 Repetition Structures

Løkker - gjenta kodelinjer flere ganger - Kap. 4.1

- Det er nyttig å kunne gjenta en del av et program flere ganger, uten å skrive det samme mange ganger.
- Til dette bruker man løkker.

Dette er tungvint, bruk heller løkke:

```
print('Løkker er lurt!')
print('Løkker er lurt!')
print('Løkker er lurt!')
print('Løkker er lurt!')
print('Løkker er lurt!')
print('Løkker er lurt!')
print('Løkker er lurt!')
print('Løkker er lurt!')
print('Løkker er lurt!')
...
```

Ordet *løkke* brukes fordi vi skal gjenta noe flere ganger (rundt og rundt)

Med løkke:

```
for i in range(10):
   print('Løkker er lurt!')
```

Begreper

- Løkke:
  - En repeterende struktur som gjør at en kodelinje eller flere kodelinjer blir kjørt flere ganger.
- Betingelseskontrollert løkke (while):
  - Bruker en True/False betingelse for å bestemme hvor mange ganger kodelinjene skal kjøres
  - Kalles en ***pretest***-løkkestruktur fordi man tester betingelsen før løkken utføres.
- Antallkontrollert løkke (for):
  - Repeterer kode et bestemt antall ganger

To typer løkker i Python

- While-løkke
  - Hvis man skal gjenta en "kodebit" flere ganger så lenge en betingelse er oppfylt.
  - Typisk gjentagelse av kode hvor man ikke vet helt hvor mange ganger koden skal gjentas.
- For-løkke
  - Hvis man skal gjenta en "kodebit" et bestemt antall ganger, bruker man en for-løkke.
  - Typisk hvis man skal gjøre noe med alle elementene i ei liste eller en tabell.

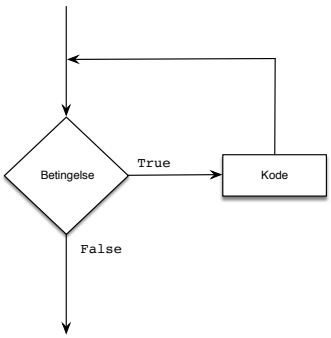
```
a = 0
while a < 10:
    print('Løkker er lurt!')
    a = a + 1
```

Ln: 5 Col: 0

```
for a in range(10):
    print('Løkker er lurt!')
```

Ln: 3 Col: 0

while-løkka Kap. 4.2




```
x = 5
while x > 0:
    print(x)
    x = x - 1
print('Der var løkka slutt.')
```

Ln: 6 Col: 0

```
rary/Mobile Documents/com~ap
ple~CloudDocs/ITGK 2016/Pyth
on/PDF/02eks.py
5
4
3
2
1
Der var løkka slutt.
>>>
```

Ln: 95 Col: 0

Skrive en while-løkke

- Generell format for while-løkke i Python:

```
while betingelse:
    kodelinje1
    kodelinje2
kodelinje3
```

- Hvis **betingelsen** er **True** vil kodelinje1 og kodelinje2 bli utført helt til betingelsen blir False
  - Kun kode med innrykk hører til while-løkka (altså ikke kodelinje3)
  - NB: Pass på at det skjer noe i sløyfa som gjør at betingelsen for å oppholde seg i sløyfa har en mulighet for å feile.

Her endres verdien av x slik at det er mulig at x > 0 kan bli False og while-løkka avsluttes

02eks.py - /Users/terjery/Library/Mobile Document...

```
x = 5
while x > 0:
    print(x)
    x = x - 1
print('Der var løkka slutt.')
```

Ln: 6 Col: 0

Et eksempel på bruk av while

- Summer et vilkårlig antall positive tall ved at brukeren skal skrive inn tall ved hjelp av `input()`
- Brukeren avslutter ved å skrive inn 0 og deretter skrives summen ut til konsollet (tekstvinduet)
- Vi titter på en mulig løsning

05eks.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/PDF/05eks.py (3.5.1)

```
summen = 0  # Nullstiller variabelen som skal inneholde summen
tall = -1   # Skal brukes i betingelsen i while-sløyfa
print('Programmet summerer tallene du skriver inn.',end = ' ')
print('Avslutt summeringen ved å skrive 0.')

while tall != 0:    # Så lenge tallet ikke er 0
    tall = int(input('Skriv inn tall: '))
    summen = summen + tall

print('Summen av tallene blir',summen)
```

Ln: 10 Col: 0

kode: summer.py

Evig løkke

- Løkker må inneholde kode som gjør at de terminerer (avslutter):
  - Noe inne i while-løkka må føre til at betingelsen blir False
- Evig løkke: En løkke som ikke har kode som gjør at den stopper
  - Programmet gjentar kode helt til brukeren avbryter programmet
  - Oppstår når programmereren glemmer å inkludere kode som avslutter løkka (gjør betingelsen False)

Litt større eksempel

Lag et program som tar inn salgsbeløp fra tastaturet og beregner hvor mye selgeren skal ha i kommisjon. For beløp over 10000 skal han ha 10%, for beløp mellom 5000 og 10000 skal han ha 5%. Programmet skal beregne den totale kommisjonen selgeren skal ha og kjøre helt til brukeren sier det er slutt.

Planlegging:

1. Vi må bruke en while-løkke siden det ikke er forbestemt hvor mange dataverdier som skal legges inn
2. Vi må ha en variabel for å summere sammen kommisjonen.
3. Vi må bruke en if-struktur for å avgjøre kommisjonsbeløpet. Denne må ligge inne i while-løkka
4. Printsetningen for å skrive ut resultatet må ligge utenfor while-løkka

Pseudokode:

1. Nullstill en variabel for å inneholde kommisjonssummen.
2. Nullstill en variabel for å styre while-løkka. Den må få en verdi slik at vi kommer inn i while-løkka første gang
3. Så lenge det er data igjen
    1. Hent inn data fra tastaturet. Verdien må konverteres til tall - float
    2. Sjekk verdien og beregne kommisjon.
    3. Oppdater kommisjonssummen ved å legge til ny kommisjon til den gamle verdien
    4. Spør brukeren om det skal legges inn flere verdier
4. Skriv ut resultatet

Større eksempel

Pseudokode:

1. Nullstill en variabel for å inneholde kommisjonssummen.
2. Nullstill en variabel for å styre while-løkka. Den må få en verdi slik at vi kommer inn i while-løkka første gang
3. Så lenge det er data igjen
    1. Hent inn data fra tastaturet. Verdien må konverteres til tall - float
    2. Sjekk verdien og beregne kommisjon.
    3. Oppdater kommisjonssummen ved å legge til ny kommisjon til den gamle verdien
    4. Spør brukeren om det skal legges inn flere verdier
4. Skriv ut resultatet

Kommisjon.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Py...

```
komSum = 0
fortsett = 'ja'

while fortsett == 'ja':
    salg = float(input('Oppgi salgssum: '))
    if salg >= 10000:
        komm = 0.10
    elif salg >= 5000:
        komm = 0.05
    else:
        komm = 0
    komSum = komSum + salg * komm
    fortsett = input('Flere verdier (ja/nei): ')
print('Din kommisjon blir:',format(komSum,'.2f'),'kr.')
```

Ln: 2 Col: 0

Oppgave: while-løkke

- Skriv Python-koden til følgende pseudokode:
  - Opprette en variabel n og sette den lik 5
  - Så lenge at n er større enn 0, gjenta:
    - skriv ut verdien av n til konsoll
    - minsk verdien av n med 1

kode: oppgave-while-py

Oppgave: while-løkke

- Skriv Python-koden til følgende pseudokode:
  - Opprette en variabel n og sette den lik 5
  - Så lenge at n er større enn 0, gjenta:
    - skriv ut verdien av n til konsoll
    - minsk verdien av n med 1

oppgave-while.py - /Users/terj...

```
n = 5

while(n>0):
    print(n)
    n = n - 1
```

Ln: 1 Col: 0

Et enkelt spill vha while

- Vi skal lage et enkelt gjettespill, der spilleren skal gjette på et tilfeldig tall mellom 1 og 100.
- Hvis man gjetter for lavt eller høyt, skal spilleren få beskjed om dette.
- Hvis man gjetter riktig, er spillet ferdig.
- For å få til et tilfeldig tall brukes:

```
import random # Laster inn random bibliotek
random.random() #Tilfeldig flyttall mellom 0 og 1
```

- For å gjøre om desimaltall til heltall brukes int()

```
heltall = int(desimaltall)
```

- Alternativ for å få et tilfeldig heltall mellom start og slutt:

```
random.randint(start, slutt)
```

kode: gjettespill.py

Et enkelt spill vha while

- Vi skal lage et enkelt gjettespill, der spilleren skal gjette på et tilfeldig tall mellom 1 og 100.
- Hvis man gjetter for lavt eller høyt, skal spilleren få beskjed om dette.
- Hvis man gjetter riktig, er spillet ferdig.
- For å få til et tilfeldig tall brukes:

```
import random # Laster inn random bibliotek
random.random() #Tilfeldig flyttall mellom 0 og 1
```

- For å gjøre om desimaltall til heltall brukes int()

```
heltall = int(desimaltall)
```

- Alternativ for å få et tilfeldig heltall mellom start og slutt:

```
random.randint(start, slutt)
```


Pseudokode

1. Importer biblioteket random
2. Plukk ut et tilfeldig tall
3. Sett gjettevariabelen lik en verdi du vet ikke er riktig slik at man kommer inn i while-løkka
4. Så lenge dette tallet ikke er riktig
   1. Les inn en gjettverdi
   2. Sjekk om den er rett og skriv om den er for liten eller for stor
5. Skriv melding om at du har funnet svaret

Et enkelt spill vha while

Pseudokode

1. Importer biblioteket random
2. Plukk ut et tilfeldig tall
3. Sett gjettevariabelen lik en verdi du vet ikke er riktig slik at man kommer inn i while-løkka
4. Så lenge dette tallet ikke er riktig
   1. Les inn en gjettverdi
   2. Sjekk om den er rett og skriv om den er for liten eller for stor
5. Skriv melding om at du har funnet svaret

gjettespill.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/Keynote/Uke...

```
import random # Henter inn random-biblioteket

tall = random.randint(1,100) # tall mellom 1 og 100

gjett = -1 # while maa fungerer forste gang

while (tall != gjett): #saa lenge man gjetter feil
    gjett = int(input("Gjett tall mellom 1 og 100: "))
    if (gjett > tall):
        print("Psst: Litt for stort!")
    elif (gjett < tall):
        print("Psst: Litt for smaatt!")

print("RIKTIG!!!! Hvordan klarte du det????")
```

Ln: 14 Col: 45

Et enkelt spill vha while

while-kriteriet kan utformes på mange måter

gjettespill2.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/Keynot...

```
import random # Henter inn random-biblioteket

tall = random.randint(1,100) # tall mellom 1 og 100

riktig = False # while maa fungerer forste gang

while not riktig: #saa lenge man gjetter feil
    gjett = int(input("Gjett tall mellom 1 og 100: "))
    if (gjett > tall):
        print("Psst: Litt for stort!")
    elif (gjett < tall):
        print("Psst: Litt for smaatt!")
    else:
        riktig = True

print("RIKTIG!!!! Hvordan klarte du det????")
```

Ln: 14 Col: 21

for-løkker Kap. 4.3 og 4.4

- Brukes for å gjenta en blokk et bestemt antall ganger.
- for-løkke er designet for å jobbe med en sekvens (**liste**) av dataelementer
  - Gå igjennom sekvensen (liste), element for element
- Liste
  - sekvens av dataelementer skilt med komma og omkranset av [ ]
- Generelt format:

```
for variabel in [verdi1, verdi2,…, verdiN]:
    kodelinjer
```

- Variabel får verdier fra lista fra første element til siste, element for element for hver runde.
- Koden som blir utført av for-løkka må ha innrykk

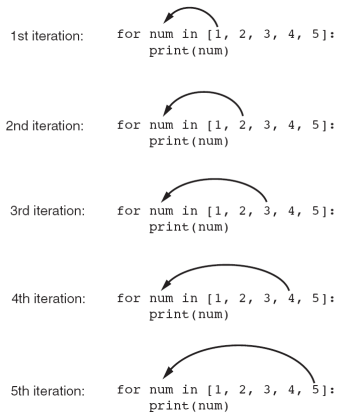
*tulletest2.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016/Python/1...

```
for tall in [1,2,3,4,5,6,7,8,9,10]:
    verdi = tall * 5
    print(format(tall,'2d'),'*5 =',format(verdi,'3d'))
```

Ln: 1 Col: 0

Hva skjer når man kjører ei for-løkke

- Kode som skriver ut tallene 1 til 5.
- Variabelen num får verdier fra lista, element for element, for hver runde løkka kjører



for-løkke for variabel som inneholder en liste

- En for-løkke kan også kjøres på en variabel som inneholder en liste av verdier:

```
navneliste=['Ole','Per','Oline','Anna','Frida']
for navn in navneliste:
    print(navn)
```

  - Skriver ut navnene til skjerm i rekkefølge
  - En for-løkke kan iterere (gå igjennom) ei liste av alle typer verdier (strenger, heltall, flyttall, sannhetsverdier osv).
- Ei liste defineres ved å bruke [ ] rundt verdier og komma mellom elementene

Bruk av range-funksjonen i for-løkker

- range er en funksjon som gjør det enklere å skrive tellende for-løkker.
- range-funksjonen lager et objekt av typen **iterable**.
  - **iterable** er et objekt som ligner på en liste og inneholder verdier som en liste kan iterere over.
- range kan ha tre varianter:
  - range(til) # tilsvarer [0,1,2…,til-1]
  - range(fra,til) # tilsvarer [fra, fra+1, fra+…n, til-1]
  - range(fra, til, steg) # tilsvarer [fra, fra+steg, fra+2*steg,… ,til-1]
- Det siste elementet i en liste generert av range er alltid en mindre enn verdien som oppgis som stoppverdi
  - les det som opp **til** og ***ikke*** opp **til og med**

Kodeeksempler på bruk av range og liste

- Range:

```
for n in range(5):
    print(n)

for n in range(1,5):
    print(n)

for n in range(1,6,2):
    print(n)

for n in range(10,1,-3):
    print(n)
```

Liste:

```
for n in [0,1,2,3,4]:
    print(n)

for num in [1,2,3,4]:
    print(n)

for num in [1,3,5]:
    print(n)

for num in [10, 7, 4]:
    print(n)
```


Oppgave: for-løkke

- Skriv Python-koden en for-løkke som skriver ut:
  - Alle tallene i 3-gangen sammen med kvadratet (3^2, 6^2 …)
  - Det skal være en tabulator mellom 3 gangen og kvadratet.

kode: kvadrat.py

Oppgave: for-løkke

- Skriv Python-koden en for-løkke som skriver ut:
  - Alle tallene i 3-gangen sammen med kvadratet (3^2, 6^2 …)
  - Det skal være en tabulator mellom 3 gangen og kvadratet.

3 løsningsforslag

kvadrat-2.py - /Users/terjery/Library/Mobile Documents/co...

```
for x in range(3,31):
    if x % 3 == 0:
        print(x,'\t',x**2)
```

Ln: 3 Col: 26

kvadrat.py - /Users/terjery/Library/Mobile Docu...

```
for x in range(3,31,3):
    print(x,"\t",x**2)
```

Ln: 3 Col: 0

kvadrat format.py - /Users/terjery/Library/Mobile Documents/com~apple~CloudDocs/ITGK 2016...

```
for x in range(3,31,3):
    print(format(x,'3d'),'\t',format(x**2,'3d'))
```

Ln: 4 Col: 0

Endring av verdier i variabler Kap 4.5

Lang-form	Kompakt-form	Hva gjøres
`x = x + 4`	`x += 4`	Øker verdien av x med 4
`x = x – 3`	`x -= 3`	Minsker verdien av x med 3
`x = x * 10`	`x *= 10`	Multipliserer verdien av x med 10
`x = x / 2`	`x /= 2`	Dividerer verdien av x med 2
`x = x % 4`	`x %= 4`	Resten av x delt på 4

Det sparer skriving og øker lesbarheten

Validering av input vha løkker Kap. 4.6

- Ofte ønsker man å sikre seg at riktig verdier på input fra brukeren før programmet går videre.
- Dette kan man gjøre vha while-løkke:

```
alder = int(input("Hvor gammel er du? "))
while (alder<0 or alder>120):
    print("Feil: Alder må være mellom 0 og 120!")
    alder = int(input("Hvor gammel er du? "))

print("Takk for den du!!!")
```

kode: alder.py

Nøstede løkker Kap. 4.7

- En løkke inne i en annen løkke kalles nøstede løkker.
- Noen fenomener er av en slik art at man trenger en hierarkisk gjennomkjøring av løkke.
- Tid er et godt eksempel der man teller først 60 sekunder, før man øker minutt med 1 osv.
- Utskrift av tid som nøstede løkker:

```
for t in range(24):
    for m in range(60):
        for s in range(60):
            print(t,":",m,":",s)
```

kode: tid.py

Oppgave: nøstede for-løkke

- Skriv Python-koden til følgende pseudokode:
  - La y løpe igjennom tallene fra 1 til 3
  - La x løpe igjennom tallene fra 1 til 3
  - Sett z lik x multiplisert med y
  - Skriv ut verdien av z til konsoll
- Hva blir skrevet ut til konsoll?

kode: oppgave_dobbel_lokke.py

Oppgave: nøstede for-løkke

- Skriv Python-koden til følgende pseudokode:
  - La y løpe igjennom tallene fra 1 til 3
  - La x løpe igjennom tallene fra 1 til 3
  - Sett z lik x multiplisert med y
  - Skriv ut verdien av z til konsoll
- Hva blir skrevet ut til konsoll?

oppgave_dobbel_lokke.py - /Users/terjery/Library/Mobile Docum...

```
for y in range(1,4):
    for x in range(1,4):
        z = x*y
        print(z)
```

Ln: 1 Col: 0

Python 3.5.1 Shell

```
eynote/uke 39/Python/o
ppgave_dobbel_lokke.py
1
2
3
2
4
6
3
6
9
>>>
```

Ln: 36 Col: 0

Oppsummering

- while-løkke brukes når en betingelse avgjør antall iterasjoner (gjennomganger):
  - while(betingelse): …
- for-løkke brukes når man skal iterere over en liste eller et bestemt antall ganger:

```
for x in [1,2,3,4]:
for y in ["test",3.14,True,9]:
for z in range(1,5,2):
for i in range(1,5):
```

- Nøstede løkker er løkker inne i andre løkker:

```
for x in [1,3,5]:
    for y in range [5,7,12]:
        …
```