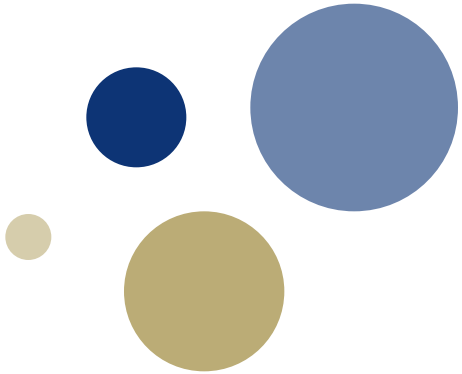
NTNU

Kunnskap for en bedre verden



Python: Intro til funksjoner

TDT4110 IT Grunnkurs

Professor Guttorm Sindre

Snart referansegruppemøte

- Viktig mulighet for å gi tilbakemelding på emnet
  - Pensumbøker
  - Forelesninger
  - Øvingsforelesninger
  - Veiledning på sal
  - Øvingsoppgaver
  - Andre læringsressurser (kollokvier, Piazza, videoer…)
- Ros, kritikk, forbedringsforslag…
- Jeg går ut av auditoriet ca. 10 min før pause i dag
  - Så kan referansegruppemedl. fasilitere en diskusjon
  - Også mulig å kontakte ref.gr.medl. på andre måter
    - Epost, pauser, …
  - Se itgk.idi.ntnu.no for hvem som er i ref.gr. fra din klasse

Læringsmål og pensum

- Mål
  - Forstå hvorfor man deler opp programmer i funksjoner
  - Bli i stand til å definere og kalle funksjoner
  - Klare å bruke lokale variable
  - Klare å overføre parametre til funksjoner
  - Lære fornuftig bruk av globale variable og konstanter
- Pensum
  - 3.utg. …Python: Ch. 5.1-5.7 (2.utg. Ch. 3)

Introduksjon til funksjoner

Kapittel 5.1

Introduksjon til funksjoner

- **Funksjon:**
  - en gruppe kodelinjer som utfører en spesifikk oppgave
- Hvorfor bruke funksjoner?
  - Dele et større program opp i mindre deler
    - Tydelig struktur, bedre oversikt: lettere å forstå
    - Enklere testing: en og en funksjon
    - Lettere samarbeid mellom flere personer
  - Gjenbruk av kode
    - Hvis samme kode trengs flere steder i ett program,
      - kalle funksjonen alle disse stedene
      - Kortere program, raskere skrevet
    - Hvis samme operasjoner trengs i mange programmer
      - alle kalle samme funksjon: enda mer tid spart

Funksjoner vs. tidligere lært

- Funksjoner vs. Variable:
  - Variabel: Huske data (dvs. verdier) til senere bruk
  - Funksjon: Huske operasjoner (dvs. kodelinjer) til senere bruk
- Funksjoner vs. Løkker:
  - Løkker:
    - Gjøre samme kodelinje(r) om igjen en bestemt plass i programmet
    - Formål: REPETISJON
  - Funksjon:
    - Gjøre samme kodelinjer flere ulike steder i ett program,
    - …eller i flere ulike program
    - Formål: GJENBRUK

Oppdeling

```
# Program uten funksjoner
statement
statement
statement
statement
statement
statement
statement
statement
…
```

```
# Program med funk
def func1( ):
    statement
    statement
    statement

def func2( ):
    statement
    statement

Statement
```

Eksempler: sang_plain.py

Vs. sang_m_funk.py

Typer av funksjoner

- Det fins ulike typer funksjoner:
  - Ferdigdefinerte vs. Egendefinerte
    - Ferdigdefinerte: allerede laget av andre, vi kan bruke dem
    - Egendefinerte må vi lage selv før de kan brukes
  - Funksjoner med og uten returverdi
    - Med returverdi: gir en verdi tilbake til hvor funksjonen ble kalt
      - Eksempel på kall: **navn=input('Navn? ')**
      - Funksjoner med returverdi: neste uke
    - Uten returverdi: gir ingen verdi tilbake
      - Eksempel på kall: **print ('Navn? ')**
- Funksjoner kan ha parametre
  - kan tilpasse sin atferd til argumenter som blir gitt inn

Definere og kalle funksjoner

Kapittel 5.2

Lage en funksjon

- En funksjon lages ved å skrive definisjonen av funksjonen:

```
def funksjons_navn():
    kode
    kode
    etc.
```

  - Første linje kalles funksjonshode:
    - Markerer starten på funksjon med det reserverte ordet `def`,
    - fulgt av navnet på funksjonen, parenteser og et kolon.
  - Resten av koden kalles ei kodeblokk som hører til funksjonen
  - **NB! Kodeblokken må skrives med innrykk!!!**

Definisjon og kall av funksjoner

Disse setninger fører til at funksjonen refreng blir opprettet

```
# Definisjon av funksjon
def refreng():
    print('Jeg gikk meg over sjø og land')
    print('Der møtte jeg...')
    # forts.

# Kall av funksjonen
refreng()
```

Denne setningen kaller funksjonen slik at den blir kjørt.

Bruk av flere funksjoner

- Programmer kan bygges av flere funksjoner.
- Vanlig praksis: hovedfunksjon kalles main( ):
  - hovedlogikken i programmet
  - gjengir overordnet struktur i programmet
  - kaller andre funksjoner som er definert

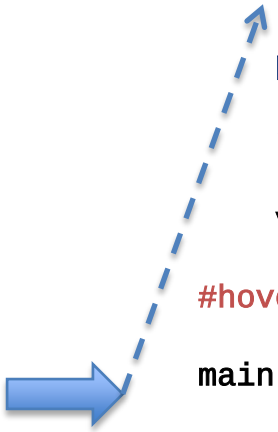
- Men vi må ha minst en kodelinje utenfor main()
  - ”skriptet” / ”hovedprogrammet”
  - Minimal løsning: her utføres kun et kall av main()

Kall av flere funksjoner

```
def refreng():
    print('Jeg gikk meg over sjø og land,')
    # print('... 3 linjer til.')
def vers(verb):
    print('Jeg hører hjemme i ',verb, 'land,', sep='')
    # print('... 3 linjer til.')
def main():
    print('JEG GIKK MEG OVER SJØ OG LAND')
    refreng()
    vers('klappe')
    ...
#hovedprogram
main()
```

Kall av flere funksjoner

```
def refreng():
    print('Jeg gikk meg over sjø og land,')
    # print('... 3 linjer til.')
def vers(verb):
    print('Jeg hører hjemme i ',verb, 'land,', sep='')
    # print('... 3 linjer til.')
def main():
    print('JEG GIKK MEG OVER SJØ OG LAND')
    refreng()
    vers('klappe') ...
#hovedprogram
main()
```



Kall av flere funksjoner

```
def refreng():
    print()
    print('Jeg gikk meg over sjø og land,')
    # print('... 3 linjer til.')
def vers(verb):
    print('Jeg hører hjemme i ',verb, 'land,', sep='')
    # print('... 3 linjer til.')
def main():
    print('JEG GIKK MEG OVER SJØ OG LAND')
    refreng()
    vers('klappe')
    ...
#hovedprogram
main()
```

Siste ufullførte kall

Kall av flere funksjoner

```
def refreng():
    print('Jeg gikk meg over sjø og land,')
    # print('... 3 linjer til.')
def vers(verb):
    print('Jeg hører hjemme i ',verb, 'land,', sep='')
    # print('... 3 linjer til.')
def main():
    print('JEG GIKK MEG OVER SJØ OG LAND')
    refreng()
    vers('klappe')
    ...
#hovedprogram
main()
```

Siste ufullførte kall

Kall av flere funksjoner

```
def refreng():
    print('Jeg gikk meg over sjø og land,')
    # print('... 3 linjer til.')
def vers(verb):
    print('Jeg hører hjemme i ',verb, 'land,', sep='')
    # print('... 3 linjer til.')
def main():
    print('JEG GIKK MEG OVER SJØ OG LAND')
    refreng()
    vers('klappe')
    ...
#hovedprogram
main()V
```

Siste ufullførte kall

Nest siste…

Kall av flere funksjoner

```
def refreng():
    print('Jeg gikk meg over sjø og land,')
    # print('... 3 linjer til.')
def vers(verb):
    print('Jeg hører hjemme i ',verb, 'land,', sep='')
    # print('... 3 linjer til.')
def main():
    print('JEG GIKK MEG OVER SJØ OG LAND')
    refreng()
    vers('klappe')
    ...
#hovedprogram
main()
```

Siste ufullførte kall

Nest siste…

Kall av flere funksjoner

```
def refreng():
    print('Jeg gikk meg over sjø og land,')
    # print('... 3 linjer til.')
def vers(verb):
    print('Jeg hører hjemme i ',verb, 'land,', sep='')
    # print('... 3 linjer til.')
def main():
    print('JEG GIKK MEG OVER SJØ OG LAND')
    refreng()
    vers('klappe')
    ...
#hovedprogram
main()
```

Siste ufullførte kall

Kall av flere funksjoner

```
def refreng():
    print('Jeg gikk meg over sjø og land,')
    # print('... 3 linjer til.')
def vers(verb):
    print('Jeg hører hjemme i ',verb, 'land,', sep='')
    # print('... 3 linjer til.')
def main():
    print('JEG GIKK MEG OVER SJØ OG LAND')
    refreng()
    vers('klappe')
    ...
#hovedprogram
main()
```

Siste ufullførte kall

Nest siste…

Kall av flere funksjoner

```
def refreng():
    print('Jeg gikk meg over sjø og land,')
    # print('... 3 linjer til.')
def vers(verb):
    print('Jeg hører hjemme i ',verb, 'land,', sep='')
    # print('... 3 linjer til.')
def main():
    print('JEG GIKK MEG OVER SJØ OG LAND')
    refreng()
    vers('klappe')
    ...
#hovedprogram
main()
```

Siste ufullførte kall

Nest siste…

Kall av flere funksjoner

```
def refreng():
    print('Jeg gikk meg over sjø og land,')
    # print('... 3 linjer til.')
def vers(verb):
    print('Jeg hører hjemme i ',verb, 'land,', sep='')
    # print('... 3 linjer til.')
def main():
    print('JEG GIKK MEG OVER SJØ OG LAND')
    refreng()
    vers('klappe')
    ...
#hovedprogram
main()
```

Siste ufullførte kall

Kall av flere funksjoner

```
def refreng():
    print('Jeg gikk meg over sjø og land,')
    # print('... 3 linjer til.')
def vers(verb):
    print('Jeg hører hjemme i ',verb, 'land,', sep='')
    # print('... 3 linjer til.')
def main():
    print('JEG GIKK MEG OVER SJØ OG LAND')
    refreng()
    vers('klappe')
    ...
#hovedprogram
main()
```

Ingen flere instruksjoner, programmet ferdig!

Designe et program for å bruke funksjoner

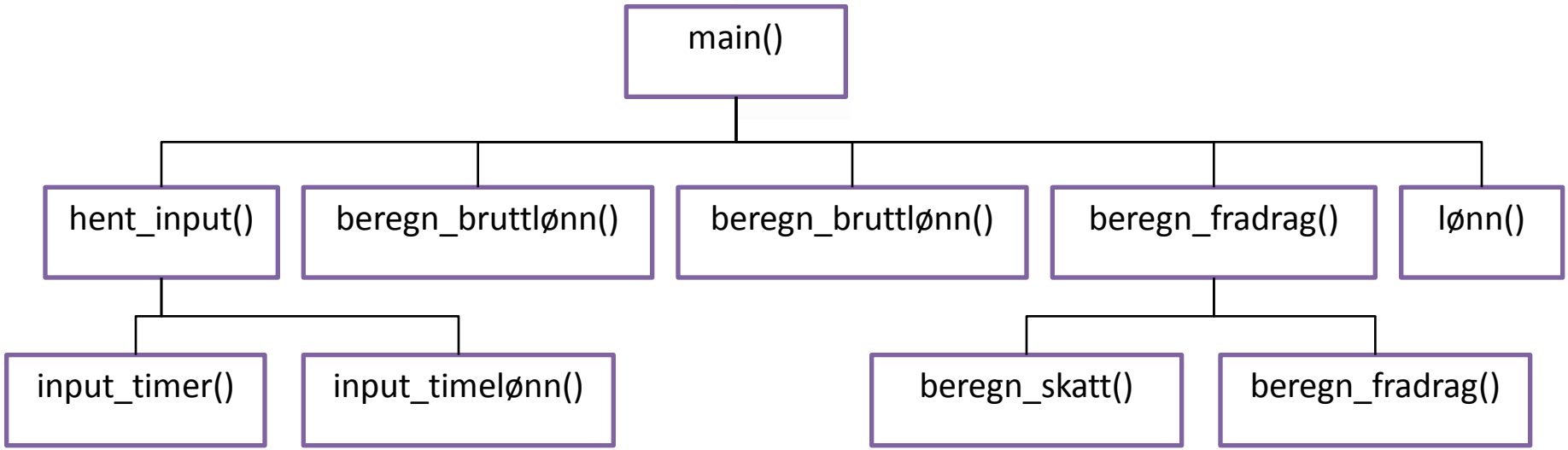
Kapittel 5.3

Top-down design

- En måte å tenke på
  - Bryt programmet ned i en serie av deloppgaver
  - For hver deloppgave: Kan den brytes ned videre?
  - Når alle deloppgaver er så enkle at videre nedbrytning er poengløs, skrives koden
- Bygger skjelettet først, så fylle inn detaljene
- Motsatte: bottom-up design, først lage de detaljerte kodelinjene, så tenke på funksjonsoppdeling

Hierarkisk skjema

- Hierarkisk skjema kan brukes for å lage en grafisk oversikt over et program:



Pause kjøring til brukeren trykker ’enter’

- Av og til er det ønskelig at programmet pauser
  - F.eks.: Mer tekst enn det som får plass på skjermen på en gang
  - Mye å lese for brukeren
  - Vente til brukeren sier det er greit å fortsette
- Til å gjøre dette kan man bruke funksjonen input:

```
input(’Trykk på ENTER for å gå videre’)
```

Overføring av argumenter til funksjoner

Kapittel 5.5

Argumenter og parametre

- Ulike behov for funksjoner:
  - Gjøre eksakt det samme hver gang den kalles
    - Trenger INGEN parametre, jfr. def refreng( ):
  - Kunne gjøre litt ulike ting fra gang til gang
    - Bruker en eller flere parametre , jfr. def vers(verb):
- Parameter:
  - Står inni funksjonsparentesen i def-linja
  - Fungerer som en lokal variabel i funksjonskoden
- Argument:
  - Står inni funksjonsparentesen der den kalles
  - Kan være en verdi, variabel, eller uttrykk som blir en verdi
  - Argumentverdien overføres til funksjonen,
    - Dvs. puttes inn for parametervariabelen når funksjonen utføres

Argumenter -> parametre

- Vanligst:
  - Samme antall argumenter som parametre
  - Samme rekkefølge
    - Kalles ”positional arguments”
    - Se kodeeksempel sang_m_funk_position.py
- Korrekt rekkefølge kan avvikes i kallet hvis vi
  - oppgir hvilket argument som skal gis til hvilken parameter
    - Kalles ”keyword arguments”
    - Se kodeeksempel sang_m_funk_keyword.py
- Kan også i noen tilfeller avvike antall
  - Bruke færre argumenter enn funksjonen har parametre
    - NB: Kun hvis de som droppes har default-verdier
    - Se kodeeksempel sang_m_funk_default.py

Lokale og globale variable

Kapittel 5.4 + 5.6

Lokale vs. globale variable

- En lokal variabel
  - blir opprettet inni en funksjon
  - skal kun brukes lokalt i funksjonen
  - er umulig å nevne i kode utenom variabelens skop
- En global variabel
  - Defineres i hovedprogrammet
  - Er synlig der og i alle programmets funksjoner
  - Bruk av globale variable
    - Kan uten videre brukes / vises av alle funksjoner
    - Kan endres hvis man i funk. skriver **global variabelnavn**
  - IKKE anbefalt, annet enn for konstanter

Se eksempel fylkesmann_u_forklaring.py / fylkesmann_m_forklaring.py

Oppsummering

- Funksjoner gir mange fordeler:
  - Bedre struktur, arbeidsdeling, kortere programmer, gjenbruk
  - Særlig hvis like eller lignende oppgaver skal gjøres mange steder
- En funksjon må defineres og består av hode og kropp:

```
def funksjonsnavn():
    kode...
```

- En funksjon kalles (kjøres) med funksjonsnavnet:

```
funksjonsnavn()
```

- Lokale variable lurt for interne beregninger i funksjonen
  - Plass frigjøres igjen straks funksjon er slutt
- Globale variable mindre lurt (unntatt konstanter)

Oppsummering

- Funksjoner kan ta imot verdier i parametre.
  - en variabel som tar imot en verdi når funksjonen blir kalt:

```
def funksjon(param1, param2):          # Variable
    kode...
```

- Verdier kan overføres til funksjoner ved hjelp av argumenter:

```
funksjon(argument1, argument2)         # Verdier
```

  - Vanligst: posisjonelle argumenter
    - rekkefølge av argumenter matcher parameterlista i funksjonshodet
  - Men kan avvike
    - keyword-argumenter: muliggjør annen rekkefølge
    - default-verdier: gjør at man kan droppe parametre

Neste uke: Mer om funksjoner, moduler (kap.5.7-5.10)

- Aktuelle spørsmål for quiz:
  - «…what happens when the end … is reached» (Checkpoint 5.8)
  - «What is a variable's scope?» (Checkpoint 5.11)
  - «What are the pieces / variables… called?» (Checkpoint 5.13, 5.14)
  - «What is a parameter variable's scope?» (Checkpoint 5.15)
  - «When a parameter is changed…?» (Checkpoint 5.16)
  - «What is the scope of a global variable?» (Checkpoint 5.18)
  - «What does the following statement do?» (Checkpoint 5.25)
  - «What happens if the same seed value is always used…» (Checkpoint 5.30)
  - «A variable created inside a function block is known as a…» (Review, Multiple Choice 4)
  - «When possible you should avoid using ____ …» (Review, Multiple Choice 12)
  - «A global variable whose value cannot be changed…» (Review, Multiple Choice 14)
  - «This standard library function returns a random…» (Review, Multiple Choice 15/16)
  - «This statement causes a function to end…» (Review, Multiple Choice 17)
  - « Void functions do not return any value …» (Review, True or False 2)
  - «A statement in a function can access a local variable…» (Review, True or False 7)
  - «Name and describe two parts of a function definition.» (Review, Short Answer 2)
  - «…what happens when the end of the function…» (Review, Short Answer 3)
  - «What will the following program display?» (Algorithm Workbench 4)
  - «Write a function named area…» (Algorithm Workbench 9)
- Noen av disse gis i «kahootisert form»
- Pluss 1-2 helt uannonserte spørsmål, men også om funksjoner