



NTNU

Det skapende universitet


TDT4109 Informasjonsteknologi grunnkurs:
Tema: Funksjoner del 1

- (3rd edition: Kapittel 5.1-5.6)

Førstelektor Børge Haugset

www.ntnu.no

1




YouTube-kanal ITGK

- Professor Guttorm Sindre har laget en YouTube-kanal der han gjennomgår deler av Python-pensum for ITGK:

<http://goo.gl/8Q0kNI>

www.ntnu.no

2



Læringsmål og pensum

- Mål
 - Lære hvordan man kan dele opp programmer vha. funksjoner
 - Lære å definere og kalle funksjoner
 - Lære om lokale variabler
 - Lære om overføre argumenter til funksjoner
 - Lære om globale variabler og konstanter
- (Pensum: Simple Functions
 - Starting out with Python 3rd: Chapter 5.1-5.6)

www.ntnu.no

3



Introduksjon til funksjoner

Kapittel 5.1

www.ntnu.no

4

1 desember vil nesten 100% få full pott på å besvare en eksamensoppgave slik:

```
• def calculateScores(homeGoals, awayGoals):  
  
    if homeGoals > awayGoals:  
        print('Hjemmeseier')  
    elif homeGoals < awayGoals:  
        print('Borteseier')  
    else:  
        print('Uavgjort')  
  
>>> calculateScores(2,3)
```

Introduksjon til funksjoner

- En **funksjon** er en gruppe av programlinjer som eksisterer i et program med den hensikt å gjøre en spesifikk oppgave.
- De fleste programmer er så store at det er fornuftig å del de opp i mindre deloppgaver: funksjoner!
- Dere har brukt dem allerede! `str()`, `print()`, `int()`...
- Vi ønsker abstraksjon med den hensikt å gjemme detaljer man ikke trenger i programmeringen.
- I programmering ønsker man å skille mellom *hva* som skal gjøres og *hvordan* det skal gjøres.
 - Eks. Vi kan benytte oss av et Python-program som er skrevet av noen andre uten at vi forstår alt programmet gjør.

Konzept

Programmet uten funksjoner:

[illegible]

Programmet delt opp i funksjoner:

```
def function1():
    statement
    statement
    statement
```

```
def function2():
    statement
    statement
    statement
```

```
def function3():
    statement
    statement
    statement
```

Fordeler med å dele opp et program i funksjoner:

- Enklere kode: Enklere å lese og forstå (mer logisk)
- Gjenbruk av kode: Slipper å skrive samme kode flere ganger
- Bedre testing: Testing og debugging er enklere med funksjoner (kan teste en og en funksjon)
- Raskere utvikling: Kode som trengs i flere deler av systemet kan brukes flere ganger
- Bedre støtte for samarbeid: Kan fordele funksjoner på flere programmerere

Definere og kall av funksjoner

Kapittel 5.2

9

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!!!

10

Lage og kalle en funksjon

- Eksempel på en funksjon som skriver tekst til konsoll:

```
def melding():  
    print('Her er en funksjon')  
    print('Som skriver til konsollet')
```

- Funksjonsdefinisjon definerer hva funksjonen gjør.

- For å kjøre en funksjon må man kalle funksjonen:

```
melding()
```

–Når en funksjon blir kalt, hopper tolkeren til funksjonen og utfører all koden i kodeblokken til funksjonen

–Parentesen viser at dette ikke er en variabel

11

Definisjon og kall av funksjoner

Disse setninger fører til at funksjonen `melding` blir opprettet

```
# Definisjon av funksjon  
def melding():  
    print('Hei du der')  
    print('Hva er dette?')
```

```
# Kall av funksjonen  
melding()
```

Denne setningen kaller funksjonen slik at den blir kjørt.

12

13

Oppgave: Skriv funksjon



Skriv koden for funksjonen weather:

- Funksjonen skal spørre brukeren om å skrive navn på sted og temperatur og skrive ut "I [navn på sted] er det..." samt ulik respons avhengig av temperaturen:
 - Hvis temperaturen er 0 eller kaldere: "kaldt vintervær"
 - Hvis temperaturen er mellom 1 og 9: "typisk Trønder-vær"
 - Hvis temperaturen er 10 eller mer: "sommervær"
- F.eks: *I Oslo er det sommervær (hvis 10+C)*

NTNU
Det skapende universitet

www.ntnu.no

13

14

Bruk av flere funksjoner

- Programmer kan bygges opp ved å definere flere funksjoner.
- Det er vanlig at et program har en hovedfunksjon som kalles main:
 - main-funksjonen inneholder hovedlogikken i programmet og gjengir overordnet struktur i programmet
 - main-funksjonen kaller de andre funksjonene som er definert
 - DET MÅ IKKE VÆRE SLIK!**
 - (I python er dette først viktig når en skal lage sine egne bibliotek, og en har kode en ikke ønsker å kjøre når man importerer.)

NTNU
Det skapende universitet

www.ntnu.no

14

15

Kall av flere funksjoner

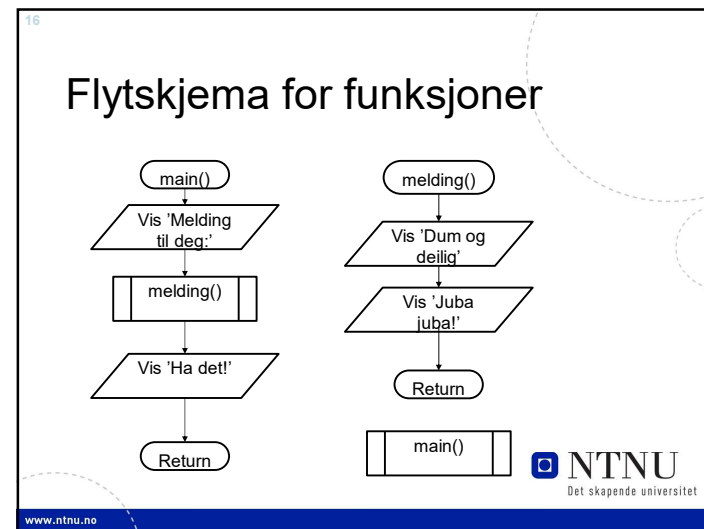
```

# Programmet har to funksjoner
# Definisjon av funksjonen main
def main():
    print('Melding til deg:')
    melding()
    print('Ha det!')
# Definisjon av funksjonen melding
def melding():
    print('Dum og deilig')
    print('Juba, juba!')
# Kall av funksjonen
main()
  
```

NTNU
Det skapende universitet

www.ntnu.no

15



16

17

Design et program til å bruke funksjoner

Kapittel 5.3

NTNU
Det skapende universitet

www.ntnu.no

17

18

Top-down design

- Top-down design er en måte å tenke på når man skal designe programmer:
 - Hovedoppgaven til programmet blir brutt ned i en serie av deloppgaver
 - Hver deloppgave blir analysert for å se om de kan brytes ned eller ikke
 - Når alle deloppgaver er identifisert, blir koden skrevet
- I top-down design bygger man skjelettet først og deretter fyller inn detaljene

NTNU
Det skapende universitet

www.ntnu.no

18

19

Hierarkisk skjema

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

```

graph TD
    main["main()"] --> hent["hent_input()"]
    main --> br1["bereg_n_brutto_lonn()"]
    main --> br2["bereg_n_brutto_lonn()"]
    main --> fradrag["bereg_n_fradrag()"]
    main --> lonn["lonn()"]
    br1 --> timer["input_timer()"]
    br1 --> timelonn["input_timelønn()"]
    br2 --> skatt["bereg_n_skatt()"]
    br2 --> fradrag2["bereg_n_fradrag()"]
  
```

NTNU
Det skapende universitet

www.ntnu.no

19

20

Lokale variabler

Kapittel 5.4

NTNU
Det skapende universitet

www.ntnu.no

20


21

Lokale variabler

- En lokal variabel blir opprettet inni en funksjon og kan ikke aksesseres (fås tak i) av kode utenom funksjonen.
- Begrepet lokal variabel indikerer at variabelen kan kun brukes lokalt i funksjonen

```
def hent_navn():
    navn = input('Skriv navnet: ') # Lokal variabel!
    # Variabelen navn kan kun brukes her!!!

hent_navn()
print('Hallo',navn)      # gir feilmelding!!!
```

 NTNU
Det skapende universitet

www.ntnu.no


21

22

Skoping (synlighet) av lokale variabler

- Skopet til en variabel betegner de deler av programmet som variabelen kan aksesseres (nås).
- En variabel er synlig bare til kodelinjene innenfor skopet til en variabel.
- En variabel er heller ikke synlig før variabelen er opprettet:

```
def feil_funksjon():
    print('Verdien er: ',verdi) # gir feilmelding
    verdi = 99
```

 NTNU
Det skapende universitet

www.ntnu.no

22

23

Kall av flere funksjoner og skop

```
farge = 'svart' # Variabel som dekker hele koden: global

def farge_min():
    print('Min favorittfarge er',farge)

def farge_jens():
    farge = 'rød' # farge variabel kun for Jens
    print('Jens sin favorittfarge er',farge)

def farge_frida():
    farge = 'grønn' # farge variabel kun for Frida
    print('Frida sin favorittfarge er',farge)

farge_min()
farge_jens()
farge_frida()

print('Min favorittfarge er fremdeles',farge) # Fremdeles svart!
```

Variablene farge i de to funksjonene er to forskjellige variabler og opphører etter at funksjonene har kjørt ferdig.

 NTNU
Det skapende universitet

www.ntnu.no

23

24

Oppgave: Flere funksjoner

- Skriv koden for følgende:
 - En funksjon `main` som skal kalle (kjøre) funksjonene `c2f` og deretter `f2c`
 - Funksjonen `c2f` skal:
 - Spørre brukeren om å skrive inn temp i Celsius
 - Skrive ut temp i Fahrenheit: $F = (C \times 9/5) + 32$
 - Funksjonen `f2c` skal:
 - Spørre brukeren om å skrive inn temp i Fahrenheit
 - Skrive ut temp i Celsius: $C = (F - 32) \times 5/9$

 NTNU
Det skapende universitet

www.ntnu.no

24

25

Overføring av argumenter til funksjoner

Kapittel 5.4

NTNU
Det skapende universitet

www.ntnu.no

25

26

Argument og parameter

- Det er ofte nyttig å sende med data til funksjoner når de kalles. Dette gjøres ved argumenter og parametere.
- Et **argument** er data som blir matet til en funksjon når en funksjon blir kalt. `int('3')`
- En **parameter** er variabelen som mottar et argument som blir matet en funksjon. Str-funksjonen bruker '3'.
 - Parameteren kan brukes som en vanlig variabel, men kun inne i funksjonen, dvs. skopet (synligheten) til parameteren er inne i funksjonen.
 - Man kan gjøre endringer i verdien til en parameter inne i en funksjon på samme måte som man endrer verdi på en variabel.

NTNU
Det skapende universitet

www.ntnu.no

26

27

Sende over flere argumenter

- Funksjoner kan ha fra ingen til mange parametere.
- Vanlig måte å sende over flere argumenter til en funksjon er å sende inn verdier i riktig rekkefølge:

```
def vis_sum(tall1, tall2):
    resultat = tall1 + tall2
    print(resultat)

vis_sum(12, 45)
```

tall1 → 12
tall2 → 45

NTNU
Det skapende universitet

www.ntnu.no

27

28

Overføring av verdier til funksjoner

```
farger_argumenter.py
1 farge = 'svart' # Variabel som dekker hele koden: global
2
3 def favorittfarge(hvem, farge):
4     print(hvem, 'favorittfarge er', farge)
5
6 favorittfarge('Min', farge)
7 favorittfarge('Jens sin', 'rød')
8 favorittfarge('Frida sin', 'grønn')
9
10 print('Min favorittfarge er frøndeles', farge) # Frøndeles svart!
```

NTNU
Det skapende universitet

www.ntnu.no

28

29

Bruke nøkkelord for argumenter

- Kan også spesifisere parameternavn som argumenter:

```
def skriv_tall(tall1, tall2):
    print('Tall 1:', tall1)
    print('Tall 2:', tall2)

skriv_tall(tall2=100, tall1=50)
```

tall1 → 50
tall2 → 100


NTNU
Det skapende universitet

www.ntnu.no

29

30

Oppgave: Funksjoner med parametere



- Skriv koden til funksjonen `areal_farget_sirkel()`:
 - Tar parametrene `r` (radius) og `farge`
 - Beregner arealet av en sirkel ved:
 - $A = \pi \cdot r^2$
 - `import math` og `math.pi*r**2`
 - Farge og areal skal skrives ut til konsollet som dette:
 - Arealet til grønn sirkel er 1256.6370614359173

NTNU
Det skapende universitet

www.ntnu.no

30

31

Globale variabler og globale konstanter

Kapittel 5.6

NTNU
Det skapende universitet

www.ntnu.no

31

32

Globale variabler og globale konstanter

- En global variabel kan aksesseres av alle funksjoner i en programfil (python-fila der koden er lagret)
- En global variabel må opprettes utenom en funksjon

```
# Opprett global variabel
verdi = 10

def vis_verdi():
    print(verdi) # Fungerer fordi verdi er global

vis_verdi()
```

NTNU
Det skapende universitet

www.ntnu.no

32

33

Endre på global variabel i funksjon

- Global variabel kan endres i funksjon ved å bruke kodeordet `global` inne i funksjonen

```
tall = 5 # Opprett global variabel

def main(): # Kan endre variabel tall her!
    global tall # Kan nå endre på variabelen tall
    tall= eval(input('Skriv et tall: '))
    vis_tall()

def vis_tall(): # Kan ikke endre variabel tall her!
    print('Tallet er:',tall)

main()
```

 NTNU
Det skapende universitet

www.ntnu.no

33

34

Bruk av globale variabler og konstanter

- I store programmer gjør bruk av globale variabler det vanskelig å finne feil (debugge) og bør derfor unngås.
- Man bør spesielt være forsiktig med å endre på globale variabler inne i funksjoner, som gjør koden svært rotete.
- Bruksområde for globale variabler er å definere konstanter som ikke skal endres i koden:
 - Konstanter er variabler som ikke endrer verdi
 - Konstanter skrives gjerne kun med store bokstaver:

```
MOMS = 0.25 # Endres sjeldent
MAT_MOMS = 0.125 # Endres sjeldent
```

 NTNU
Det skapende universitet

www.ntnu.no

34

35

Oppsummering

- Funksjoner er en måte å dele opp programmer i mindre deler som gir mange fordeler.
- En funksjon må defineres og består av hode og kodeblokk:


```
def funksjonsnavn():
    kode...
```
- En funksjon kalles (kjøres) med funksjonsnavnet:


```
funksjonsnavn()
```
- Funksjoner kan ha lokale variabler som kun kan brukes og lever inne i funksjonen.

 NTNU
Det skapende universitet

www.ntnu.no

35

36

Oppsummering

- Funksjoner kan ta imot verdier ved hjelp av parametere.
 - En parameter er en variabel som tar imot en verdi når den blir kalt:


```
funksjon(parameter1, parameter2): # Variabler
    kode...
```
- Verdier kan overføres til funksjoner ved hjelp av argumenter:


```
funksjon(argument1, argument2) # Verdier
```
- Globale variabler kan defineres utenfor funksjoner som kan brukes og nås av alle funksjoner.

 NTNU
Det skapende universitet

www.ntnu.no

36