



# NTNU

Det skapende universitet

## **TDT4109 Informasjonsteknologi grunnkurs:**

Python: Bruk av funksjoner, variabler  
og input/output

Førstelektor Børge Haugset

[www.ntnu.no](http://www.ntnu.no)

Start med Kahoot! med repetisjon fra sist:

**Starting out with Python (1): Introduction to Computers and Programming!**

## IDLE eller Thonny eller PyCharm? Einn VS Code?

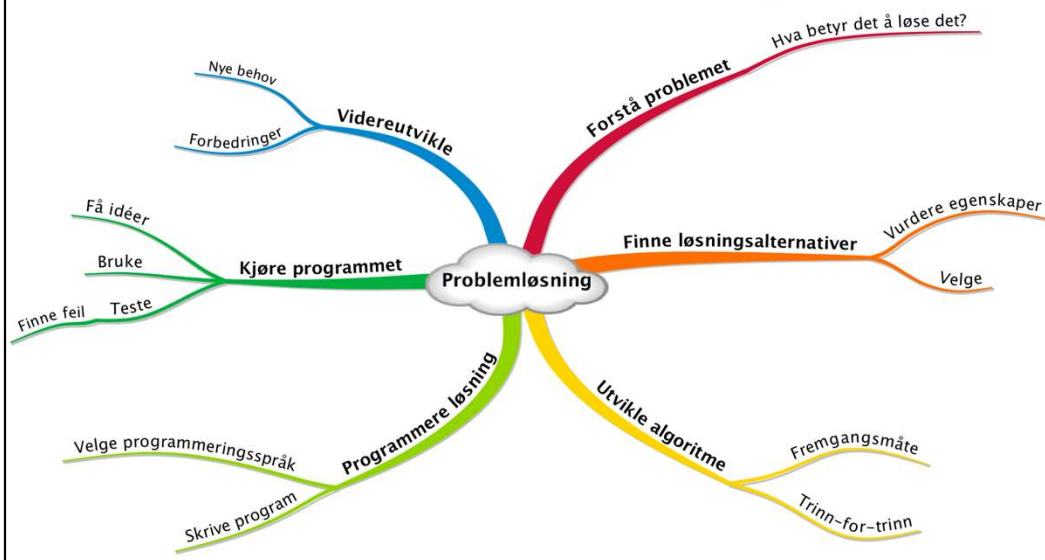
- For å nevne det – noen bruker litt mer avanserte greier
- <https://www.jetbrains.com/pycharm-edu/download/>
- Ganske likt IDLE, men må opprette prosjekt før man oppretter Python filer!
- Min mening: Bestille drosje til butikken over gata.
  - MEN: hvis du skal fortsette med programmering, da kan det være en tanke å ta overgangen. Senere vil du stort sett bruke slike verktøy.
  - Jeg skrev alle eksamensoppgavene mine i Thonny.

# Læringsmål og pensum

- Mål
  - Lære om å designe et program
  - Lære om skrive ut til skjermen (print)
  - Lære om variabler
  - Lære om å lese fra tastatur
- Pensum
  - Starting out with Python: Chapter 2  
Input, Processing, and Output

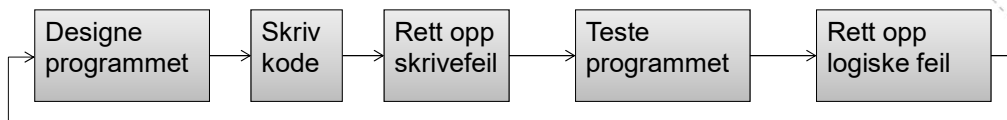
# Design et program

Kapittel 2.1-2.2



# Programutviklingssyklus

- Programutviklingssyklusen er prosessen man følger når man lager og utvikler programmer:



# Programutviklingssyklus

1. Designe programmet:
  - Forstå oppgaven som programmet skal utføre
  - Avgjøre stegene som må tas for å utføre oppgaven
2. Skrive kode:
  - Må følge regler definert av programmeringsspråket
3. Rette opp skrivefeil:
  - Hvis det er skrivefeil i programmet vil tolkeren/kompilator si ifra om feil som må rettes opp før programmet skal kjøres (syntax error)
4. Teste programmet:
  - Når programmet kan kjøres må det testes for logiske feil i koden, for eksempel feil i beregninger, produserer feil resultat, osv.
5. Rette opp logiske feil (debugging)
  - Finner og retter opp logiske feil

# Hjelp til å designe programmer: Pseudokode

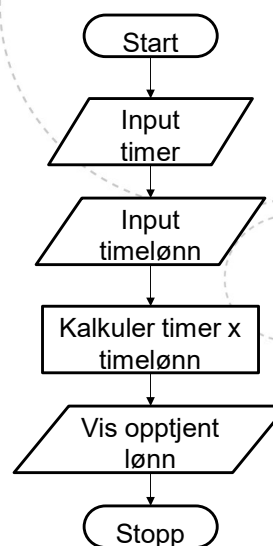
- Beskriver programmet med naturlig språk
- Pseudokode kan ikke forstås av en datamaskin, men kan oversettes av mennesker til ulike programmeringsspråk

```
Hent inn (input) antall timer jobbet  
Hent inn (input) timelønn  
Kalkuler antall timer x timelønn  
Vis (display) opptjent lønn
```



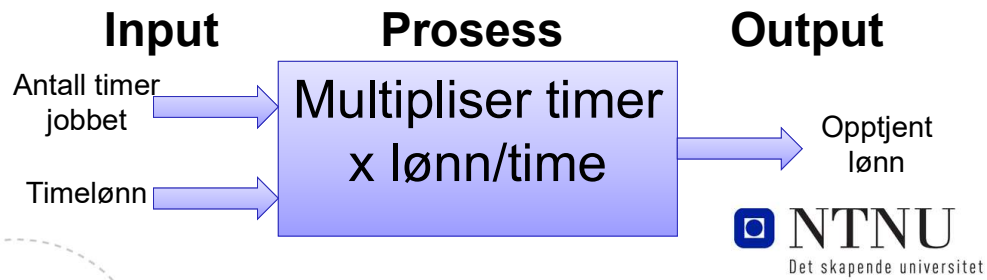
# Hjelp til å designe programmer: Flytskjema

- Beskriver programmer på en grafisk måte:
  - **Ovaler:** *Terminalsymboler* som viser start og stopp
  - **Parallellogram:** *Input- og output-symboler* (hente inn fra bruker og vise til skjerm)
  - **Rektangel:** *Prosesseringssymboler* der man utfører noe på data (for eksempel beregninger)



# Hjelp til å designe programmer: Input – prosessering - output

- Dataprogrammer utføres ofte som en trestegsprosess:
  1. Mottar input
  2. Utfører prosessering av input (gjøre noe med input)
  3. Produserer output



# Vise output med `print-` funksjonen

Kapittel 2.3

# Skrive informasjon til skjerm

- For å skrive noe til konsollet (utskriftsskjermen), brukes funksjonen `print(uttrykk)`  
`print('Hello world')`
  - Kommandoen vil skrive ut *Hello world* til konsollet.
  - 'Hello world' er en tekststreng (streng), dvs. en rekke tegn.
  - Strengen starter med en apostrof (') og avsluttes med en apostrof (')
- Uttrykket kan være et tall, utregning av tall, logiske uttrykk, tekst og/eller variabler.

## print – to fnutt or not to dobbelfnutt (apostrofer)

- I Python kan man skrive strenger både med *apostrof*/fnutter ('... ') og med *anførselstegn*/dobbelfnutter ("...")
- Dette kan brukes til å skrive tekst som har fnutter i seg:

```
print("Don't be afraid!")  
print('Han var "veldig" kul!')
```

- I Python kan man bruke trippelfnutter for å skrive tekst som inneholder både ' og ", samt linjeskift:

```
print("""I'm a green "frog",  
linje to  
linje tre""")
```

## Oppgave



- Skriv koden for å skrive ut følgende til konsollet:

```
Programmering er veldig "kult"
```

```
'To be or not to be' er et sitat
```

```
Han sa: "Her blir det mye 'fnutter' "!"
```

Kode:

```
print('Programmering er veldig "kult"')
```

```
print("'To be or not to be' er et sitat")
```

```
print("""Han sa: "Her blir det mye 'fnutter' "!" """)
```

Eller

```
print("""Programmering er veldig "kult"'
```

```
'To be or not to be' er et sitat
```

```
Han sa: "Her blir det mye 'fnutter' "!" """)
```

# Kommentarer

## Kapittel 2.4

# Kommentarer i programmer

- En god programmeringsskikk er å skrive kommentarer i koden som forklarer hva som blir gjort.
- Kommentarer i kode blir ignorert av tolker eller kompilator og blir ikke skrevet ut til skjerm.
- I Python brukes tegnet # for å indikere kommentarer.

- Alt som kommer etter # på ei linje blir ignorert av tolkeren

```
# Programmet skriver ut navn og adresse
```

```
# til en person
```

```
print('Donald Duck')    # Skriver ut navn
```

```
print('Moseveien 23B')  # Skriver ut veiadresse
```

```
print('Andreby')        # Skriver ut by
```



# Kommentere ut store seksjoner

- De samme trippelapostrofene kan brukes til å kommentere ut områder med kode:

```
x=1920
y=354
svar=x*y
navn=input("Skriv inn navnet ditt: ")
'''
# print ("Hallo", navn)
print(f"Hallo {navn}")
print(f"Navnet ditt er {len(navn)} bokstaver langt.")
'''
```

# Læringsmål og pensum

- Mål
  - ~~Lære om å designe et program~~
  - ~~Lære om skrive ut til skjermen (print)~~
  - **Lære om variabler**
  - **Lære om å lese fra tastatur**
- Pensum
  - Starting out with Python: Chapter 2  
Input, Processing, and Output

# Variabler og datatyper

## Kapittel 2.5

# Variabler

- Definisjon: En variabel er et navn som representerer en verdi lagret i datamaskinens minne.
- En *variabel* fungerer som minne på en kalkulator:
  - **M+** Lagre et tall i variabelen M
  - **Mr** Hente fram verdien fra variabelen M
  - **Mc** Slette verdien på variabelen
- I all programmering trenger man variable i forbindelse med utregning, sammenlikning, lagre og gjøre endringer på informasjon
- Det kalles en variabel fordi innholdet kan variere i løpet av programmet.

# Opprette variabler med tilordning

- Man bruker et tilordningsuttrykk for å opprette en variable og lage en referanse til en dataverdi:

```
alder = 23
```

–Variabelen alder refererer til verdien 23 i datamaskinens minne:



- En tilordning skrives på formen:

```
variabel = uttrykk
```

–Tegnet "=" (er lik) er en tilordningsoperator.

–*variabel* er navnet på variabelen

–uttrykk representerer en verdi

–små og store bokstaver?

Variabel er som en navn eller en huskelapp for en verdi lagret i minnet.

Tester ut med å opprette variabler og skrive disse ut ved hjelp av print!

Test også `23 = alder` og forklar hvorfor dette ikke fungerer!

Vis problemer hvis man bruker print og skriver variabel navnet feil

Vis problemer hvis man bruker print og skriver med storebokstaver

# En liten pausevideo...



# Lagring av verdier

- OBS! Variabler kan ikke brukes før de er opprettet!
- Hvordan lagres verdier?
- Hvor mange verdier er det plass til egentlig?
- Verdier lagres sammen med andre program og andre data i minnet (RAM på datamaskinen)
  - ligner en stor kommode med veldig mange nummererte skuffer
  - i hver skuff kan du lagre en *byte* eller 8 *biter* (bits)
  - en byte er et tall i området 0-255 (8 sifre i 2-tallssystemet)
- Python finner ut selv hvor mye plass som trengs for å lagre en verdi

# Navneregler for variabler

- Skal ikke bruke nøkkelord i Python som variabelnavn (dvs. ord som betyr noe i språket, som f.eks. print, if)
- Hvis du overskriver nøkkelord kan du bruke funksjonen `del(variabel)` for å slette variabelen.
- Variabelnavn kan ikke inneholde mellomrom (space)
- Første bokstav i navnet må være bokstav eller underscore (" \_")
- Etter første bokstav kan man bruke bokstaver, tall og underscore.
- Stor og liten bokstav tolkes forskjellig!
- Du kan bruke æ, ø, og å i variabelnavn i Python, men det anbefales ikke!

Man kan bruke nøkkelord til variabler, men da overskriver man de som finnes der fra før.



## Tips til variabelnavn

- Bruk variabelnavn som sier noe om hva variabelen skal brukes til:

```
pris = 29
```

- For variabelnavn som består av flere ord, bruk underscore mellom hvert ord (brukes i læreboka):

```
antall_elever = 300
```

```
sum_utgifter = 950
```

- Kan også bruke stor bokstav for hvert nytt ord i en variabel (kamelpukkel-stil):

```
antallElever = 300
```

```
sumUtgifter = 950
```

Personlig syns jeg underscore (understrek) gjør et enklere å lese

# Vise flere elementer med `print` funksjonen

- Funksjonen `print` gjør det mulig å vise flere elementer i samme setning ved bruk av komma mellom hvert element.

– Eksempel på å vise fire elementer til konsoll ved hjelp av `print`:

```
elever = 250
ansatte = 13
print('Skolen har', elever, 'elever og', ansatte, 'ansatte')
```

Diagram labels:

- `elever` and `ansatte` are labeled as **variabel** (variable).
- `'Skolen har'` is labeled as **tekststreng** (string).
- `'elever og'` and `'ansatte'` are labeled as **tekststrenger** (strings).

Resultat: Skolen har 250 elever og 13 ansatte

- Merk at variabler ikke har frutter rundt seg!
- Merk at `print` legger på et mellomrom mellom hvert element!

## To viktige ting på mandag:

- Innleveringsfrist øving 1
- For seint å stemme?
- Piazza
- Neste uke er det ordinær teori – uten meg!
- python – teori – øvingsforelesning – gruppe...  
HÆH?!

## En smartere måte å skrive ut strenger

- F-strings bruker et litt annet format, men det er kanskje mer logisk på mange måter. Ny metode:
- `print(f' tekst {variabel} tekst {variabel2}')`
- Merk: det er strengen som lages på en ny måte, en trenger ikke gjøre det inni en print.

```
elever = 250
ansatte = 13
print(f'Skolen har {elever} elever og {ansatte} ansatte')
```

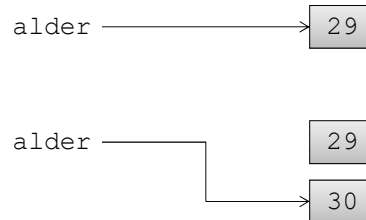
The diagram illustrates the components of the f-string `f'Skolen har {elever} elever og {ansatte} ansatte'`. Arrows point from the labels 'variabel-referanse' to the curly braces `{elever}` and `{ansatte}`. Arrows point from the label 'tekstdel' to the text `Skolen har`. Arrows point from the label 'tekstdeler' to the text `elever og` and `ansatte`.

Kom hit etter andre time.

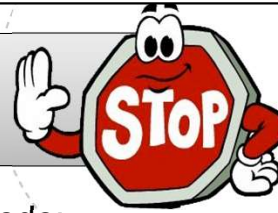
# Endring av verdi i en variabel

- En variabel kan endre verdi i løpet av et program.
- Man endrer verdien av variabel ved å gjøre en ny tilordning.
- Eks:

```
alder = 29  
print(alder)  
alder = 30  
print(alder)
```



## Oppgave: Variabler



- Skriv koden for å utføre følgende pseudokode:
  - Opprett en variabel for antall epler og gi den verdien 23
  - Opprett en variabel for antall bananer og gi den verdien 12
  - Skriv ut teksten "Antall epler:" og verdi av variabel
  - Skriv ut teksten "Antall bananer:" og verdi av variabel
  - Endre verdien av variabel for antall epler til 43
  - Endre verdien av variabel for antall bananer til 17
  - Skriv teksten "Antall epler: " og verdi av variabel, og teksten "Antall bananer:" og verdi av variabel ved hjelp av print()

```
antall_epler = 23
antall_bananer = 12
print('Antall epler:', antall_epler)
print('Antall bananer:', antall_bananer)
antall_epler=43
antall_bananer=17
print('Antall epler:', antall_epler, 'Antall bananer:', antall_bananer)
```

# Datatyper

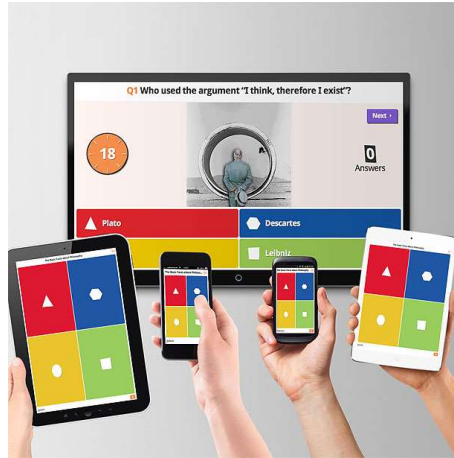
- I Python kan variabelen ta vare på ulike typer data (datatyper) som heltall, desimaltall, tekst og sannhetsverdier (flere senere).
  - Heltall (int): `antall = 7`
  - Desimaltall (float): `penger = 35.5` # Bruker . i stedet for ,
- Tekststreng (str): `navn = 'Petter'` # Bruker apostrof, ikke ``
  - Sannhetsverdi (bool): `rykte=True` eller `rykte=False`
- Datatypen til en verdi eller en variabel kan man finne ved å bruke funksjonen `type(uttrykk)`
- I Python får variabler type ved tilordning
  - Dvs. at en variabel kan bytte datatype

Test ut ulike datatyper

Test ut `type(uttrykk)`

Test ut å bytte datatype

# Oppgave: Datatyper



- <https://play.kahoot.it/#/k/bbc80db2-7562-4bc2-841f-33b090538f90>

A:I  
B:F  
C:s  
D:B  
E:F  
F:s  
G:s



## Mer om variabler og datatyper

- Noen programmeringsspråk har en grense på hvor mange siffer en variabel kan takle.
- Python har ingen grense antall siffer en variabel kan takle.
- Du kan også skrive tall angitt med vitenskapelig notasjon, f.eks:  $5.9e9$  som betyr  $5.9 * 10^9$

# Lese input fra tastatur

## Kapittel 2.6

**Tips:**

Bruk av konstant e i matte:

```
import math  
math.e
```

[www.ntnu.no](http://www.ntnu.no)

HIT!!!!

# Hent input fra tastatur med funksjonen `input`

- I Python brukes funksjonen `input` for å hente input fra tastaturet:

```
variabel = input(prompt)
```

–Her er prompt teksten som vises i konsollet når programmet spør etter input fra brukeren

–variabel er navnet på variabelen som refererer til dataen som blir tastet inn ved hjelp av tastaturet.

–Eks:

```
navn = input('Hva er navnet ditt? ')
```

–Merk at det er et mellomrom på slutten av prompt-teksten. Dette er fordi Python ikke lager mellomrom mellom prompt og det som skal skrives inn automatisk.

Test ut bruk av input

Test ut også å bruke en variabel med en tekststreng som prompt!

# Lese inn tall med input funksjonen

- input-funksjonen returnerer alltid en tekststreng, dvs. en tekst med fnutter rundt.
- Hvis du ønsker å gjøre beregninger med det som brukeren skriver inn, må du gjøre om tekststrengen til et heltall (int) eller til et flyttall (float) vha. funksjoner:

```
variabel = int(element) # Oversetter element til et heltall  
variabel = float(element) # Oversetter element til desimaltall  
variabel = eval(uttrykk) # Oversetter uttrykk til verdi
```

```
tallstreng = input("Skriv inn et tall: ")  
type(tallstreng)  
tall = eval(tallstreng)  
type(tall)
```

## Bruk av funksjonene `int()` og `float()`

- To måter å hente heltall fra input-funksjon:  

```
streng_verdi = input('Hvor mange kyr har du ?')  
kyr = int(streng_verdi) # Oversetter streng til heltall
```
- Du kan gjøre begge deler i en setning:  

```
kyr = int(input('Hvor mange kyr har du ?'))
```

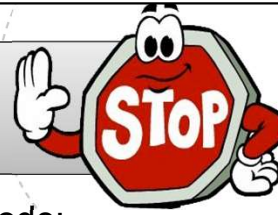
  - Kalles nøstet funksjonkall og fungerer på følgende måte:
    - Først utføres det som er innerst i parentesen (`input('Hvor mange..')`)
    - Verdien til input-funksjonen brukes så i `int`-funksjonen
    - Resultatet fra `int`-funksjonen lagres i variabelen `kyr`
- Hente ut flyttall:  

```
cash = float(input("Mye penger har'ru? "))
```

Test ut!

Vis også hva som skjer om vi prøver å taste inn tekst til `int` for å få exception. Nevn at vi kommer tilbake til dette senere!

## Oppgave: Input



- Skriv koden for å utføre følgende pseudokode:
  - Spør brukeren etter navnet og lagre navnet i en variabel
  - Skriv ut "Hei " og navnet som ble skrevet inn til konsollet.
  - Bruk: <variabel> = input(prompt) og print()
  - Hvis dette er enkelt, kan du også spørre brukeren om alder og skrive ut til skjermen hvor gammel vedkommende vil være om 10år.

```
navn = input('Hva er navnet ditt? ')
# alder = int(input('Hvor gammel er du?'))
print('Hei',navn)
#print('Om 10 år vil du være',alder+10,'år gammel')
```

# Utføre kalkulasjoner

## Kapittel 2.7

## Operator presedens (prioritert rekkefølge)

- Python har følgende matteoperatorer:  $+$   $-$   $*$   $/$   $//$   $\%$   $**$
- Ved utregninger blir disse utført ut ifra en prioritering av hvilke operatorer som utføres først:
  1. Eksponent:  $**$
  2. Multiplikasjon, divisjon og rest av divisjon:  $*$   $/$   $//$   $\%$
  3. Addisjon og subtraksjon:  $+$   $-$
  4. Det vil si at uttrykket:  $12 + 6/3$  blir 14 og ikke 6 ( $6/3$  blir utført først)
- For å sikre korrekte beregninger bruker man parenteser:
  - $(12+6)/3$  blir 6
  - $10/(5-3)$  blir 5 osv...



# Bruk av variabler i utregninger

- Når du bruker flere variabler i ett uttrykk, vil:
  - Variabelen på venstre side av "er-lik" være navnet på referansen til resultatet av utregningen
  - Alle variablene på høyreside av "er-lik" representerer kun verdier som brukes i utregning.

– Eks:

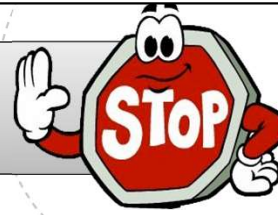
A=5

B=8

$$\begin{array}{ccccccc} C & = & A & + & B & + & 9 \\ \uparrow & & \uparrow & & \uparrow & & \uparrow \\ \text{Resultat} & & 5 & + & 8 & + & 9 \\ \text{lagres her} & & & & & & \end{array}$$

Vi prøver litt...

## Skriv kode for gjennomsnitt av tre målinger



- Pseudokode:
  - Hent inn første måling
  - Hent inn andre måling
  - Hent inn tredje måling
  - Kalkuler gjennomsnitt ved å legge samme tre målinger og dele på 3
  - Vis gjennomsnittet på skjerm
- Prøv å skriv programmet som gjør dette selv!

uke36-01.py

# Kode for gjennomsnitt av tre målinger

```
m1 = int(input('Måling 1: '))
```

```
m2 = int(input('Måling 2: '))
```

```
m3 = int(input('Måling 3: '))
```

```
gsnitt = (m1+m2+m3)/3
```

```
print('Gjennomsnitt:',gsnitt)
```

# Mer om utskrift til skjerm

## Kapittel 2.8

# Endring av oppførsel av funksjonen `print`

- Det er mulig å endre oppførselen på funksjonen `print` for hvordan den separerer elementer:

```
a=5
b=7
print(a,b,sep='_')      # gir 5_7
print(a,b,sep='')       # gir 57
```

- Kan også endre oppførsel på linjeskift:

```
print('Her kommer en setning.',end='')
print('her kommer en setning til.')
-Begge blir skrevet ut på samme linje!
```

Test ut litt!

# Escape-karakterer

- Escape-karakterer er spesialkarakterer som kan skrives inn i tekststrenger som gjør spesielle operasjoner:

<code>\n</code>	Gir linjeskift
<code>\t</code>	Hopper til neste tabulator
<code>\'</code>	Skriver ut tegnet '
<code>\"</code>	Skriver ut tegnet "
<code>\\</code>	Skriver ut tegnet \

–Eks:

```
print('\tInnrykk er kult med\nNy linje')
```

# Formattering av tall ved hjelp av funksjonen `format`

- Funksjonen `format` tar inn et tall og returnerer en tekststreng der du kan bestemme hvordan tallet skal formatteres:

```
format(tall, formattering)
```

–formattering er en tekststreng for ulike valg, f.eks:

<code>format(1/3, '.2f')</code>	# 2 desimaler, f står for float
<code>format(1/3, '10.1f')</code>	# 1 desimal og setter av ti tegn
<code>format(1/3, 'e')</code>	# vitenskapelig notasjon på tallet
<code>format(1/3, '.0%')</code>	# tallet i prosent med 0 desimaler
<code>format(500, '10d')</code>	# heltall der det settes av ti tegn

–Typisk bruk:

```
print(f"En tredjedel er omtrent {format(1/3, '6.5f')}.")
```

↑  
verdi

Prøv ut litt.

## Hmm... skal jeg si noe mer

- Man kan lese enkeltelementene i en streng

```
>>> a = 'Hei på deg'
```

```
>>> a[5]
```

```
'å'
```

- Dette ser litt ut som noe som heter en liste

# Oppsummering

- Utviklingssyklus:
  - Design program, skriv kode, rett opp skrivefeil, test programmet, rett opp logiske feil (debugging)
- Skrive til skjerm: `print(uttrykk)`
- Variabler: Referanser med navn til verdier i minnet
- Tilordning: Gi variabler verdier: `variabel = uttrykk`
- Datatyper: int, float, str, bool
- Lese fra tastatur: `variabel = input(prompt)`
- Operatorpresedens: Rekkefølge av matematiske operatorer
- Ved utrekning er variabler på høyre side av = verdier og resultatet blir lagret i variabel på venstre side!
- Formattering av tall ved hjelp av funksjonen `format(tall, formattering)`

Tema neste uke: Betingelser og logiske ut-trykk

If- else