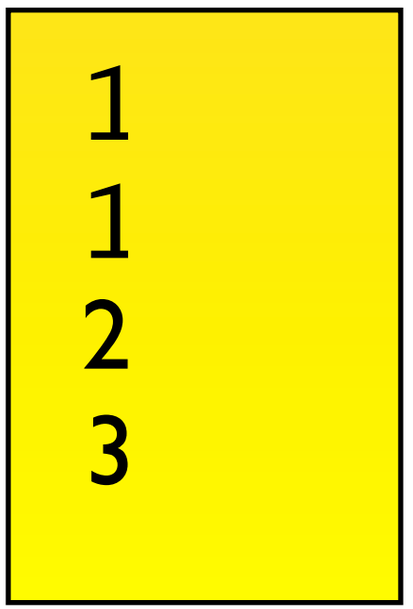
IN1000 - uke 1

Komme i gang med programmering

Et lite oppdrag i bakgrunnen

- Under pultene på bakerste rad er det klistret post-it lapper med tre tall skrevet på
- Regn ut summen av to nederste tall, skriv denne summen under de andre tallene, og send lappen til personen foran deg
- Alle som mottar en lapp (fra personen bak) gjør det samme som beskrevet over
- Jeg samler senere inn lappene fra fremste rad



Plan for forelesingen

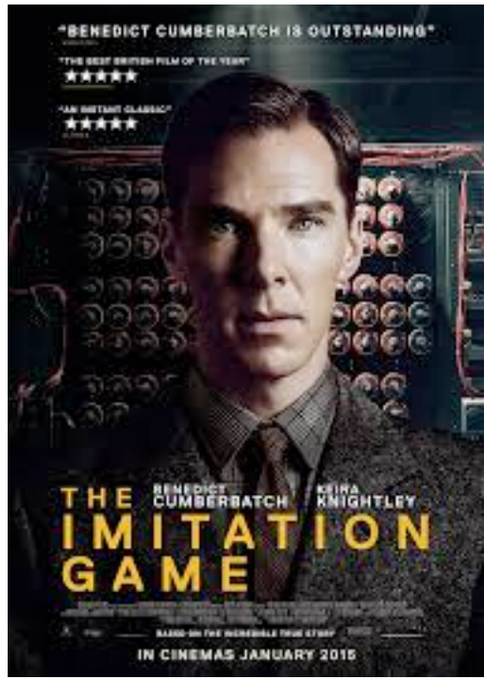
- Hva er programmering?
- Skrive og kjøre våre første program
- Variabler
- Feilmeldinger
- Innlesing fra tastatur
- Beslutninger (if)

Plan for forelesingen

- **Hva er programmering?**
- Skrive og kjøre våre første program
- Variabler
- Feilmeldinger
- Innlesing fra tastatur
- Beslutninger (if)

Hva er en “Computer”?

- "The idea behind digital computers may be explained by saying that these machines are intended to carry out any operations which could be done by a human computer" (Alan Turing, 1950)



- Tradisjonell definisjon av “computer”:
  En ansatt som gjør utregninger ved å blindt følge en liste instruksjoner med penn og papir, så lenge som det trengs, uten at det krever noen form for innsikt (fra 1600-tallet)

- Poeng for oss:
  Dette faget handler ikke om å få en printer til å virke (puh), men om å sette sammen presise oppskrifter!

Dere har i mellomtiden regnet ut Fibonaccis tallrekke!

- 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89
- Merk at resultatet av denne prosedyren er mye mer avansert enn operasjonen i hvert steg (plusse to tall)

Vi har programmert!

- Eller rettere sagt:
  - Dere har vært computere
  - Jeg har programmert dere
- Oppfordring:
  - Forsøk alltid å først løse et problem for hånd, ut fra presis sekvens av enkle operasjoner

Hva er programmering?

- *"Software development happens in your head, not in an editor. Programming is all about problem solving. It requires creativity, ingenuity, and invention" (Andy Hunt)*

  ➡ **Programmering er utfordrende, men ikke minst kreativt og gøy!**

- *"Computer science education cannot make anybody an expert programmer any more than studying brushes and pigment can make somebody an expert painter"* (Eric Raymond)

  ➡ **Forelesningene kan hjelpe dere i gang, men det er gjennom mange timer med egen programming dere virkelig lærer det**

Plan for forelesingen

- Hva er programmering?
- **Skrive og kjøre våre første program**
- Variabler
- Feilmeldinger
- Innlesing fra tastatur
- Beslutninger (if)

Et første Python-program

- Be datamaskinen om å:
  - Skrive "Hei IN1000" på skjermen

Flere Python-program

- Be datamaskinen om å:
  - Skrive 9+9 på skjermen
  - Skrive hva 9+9 blir på skjermen
  - Skrive hva 1+2+3+4 blir på skjermen
  - Skrive på skjermen om det stemmer at 5 er større enn 4
  - Skrive på skjermen om det stemmer at 4 er større enn 5

Ett Python-program som går over flere linjer

- Skriv på skjermen:

  Alle linjer
  kommer
  samtidig!

- *(flere_linjer.py)*

Kommentarer

```
#Disse to linjene
#påvirker ingen ting
print("Hei")
```

Plan for forelesingen

- Hva er programmering?
- Skrive og kjøre våre første program
- **Variabler**
- Feilmeldinger
- Innlesing fra tastatur
- Beslutninger (if)

Variabler

- Variabler er noe av det mest fundamentale i programmering!
- En variabel er et bestemt navn som representerer en verdi
  - tall = 13
- Med en variabel kan man holde på en verdi og bruke den igjen senere i et program
  - tall = 13
  - print("Her kommer det:")
  - print(tall)

Variabler:
eksempel på å holde mellomverdier

- [areal.py]

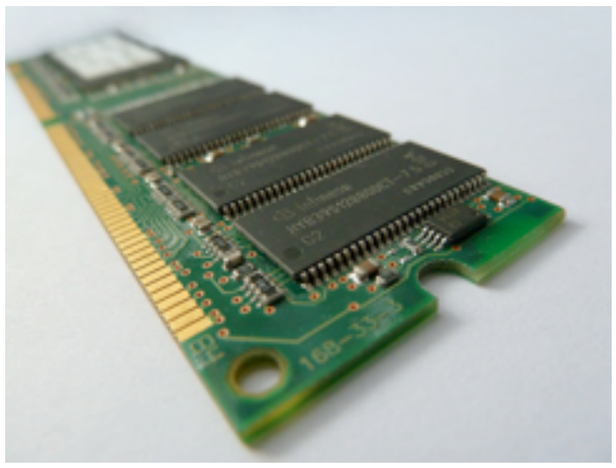
Variabler:
eksempel på å bruke flere ganger

- [Epler.py]

Tre måter å tenke på variable på (ulike metaforer)

Tre måter å tenke på variable på (ulike metaforer)

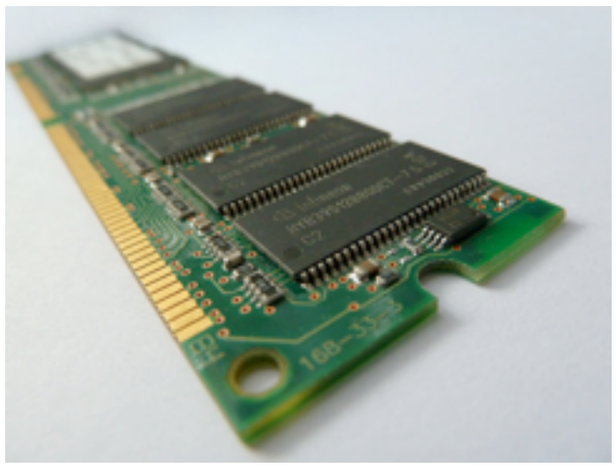
En navngitt lokasjon i datamaskinens minne hvor en verdi kan lagres



Mest korrekt, men minst nyttig..

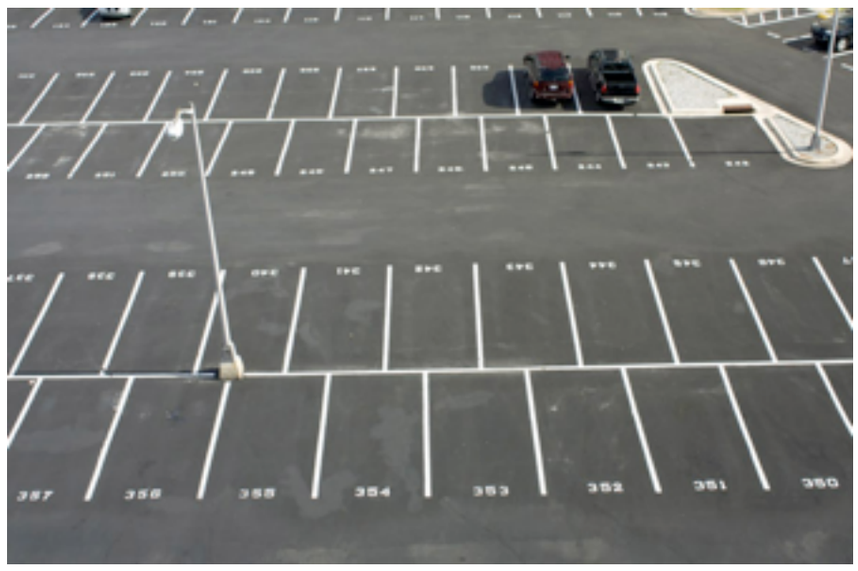
Tre måter å tenke på variable på (ulike metaforer)

En navngitt lokasjon i datamaskinens minne hvor en verdi kan lagres



Mest korrekt, men minst nyttig..

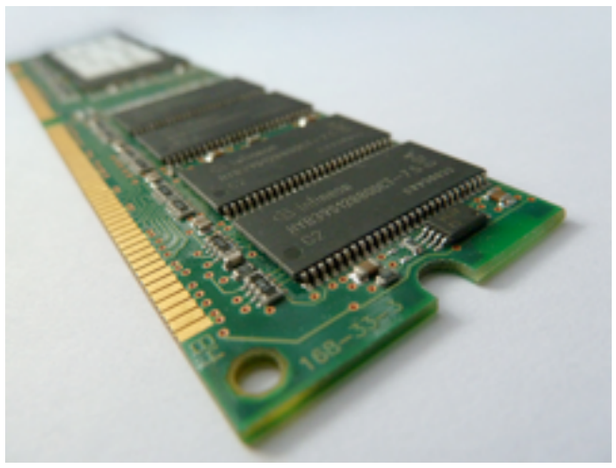
En navngitt parkeringsplass hvor en verdi kan parkeres



Nyttig metafor, brukt i lærebok

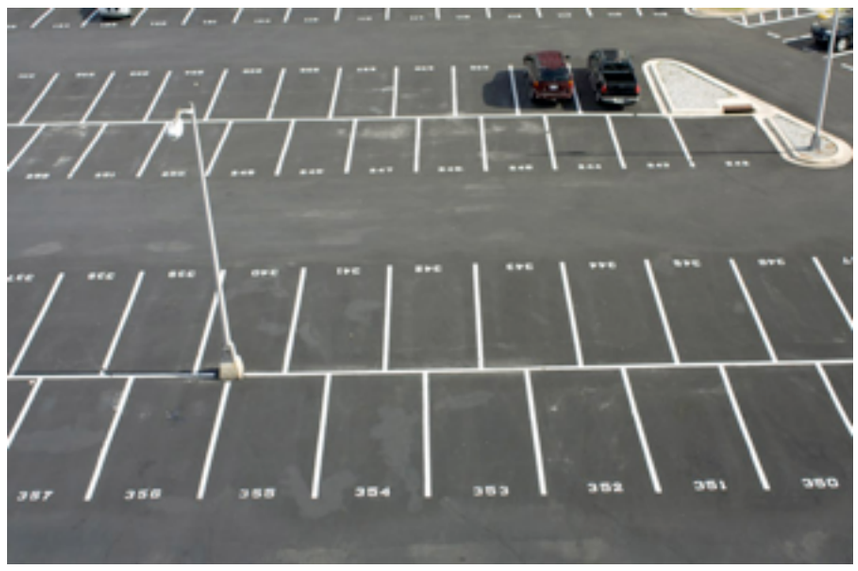
Tre måter å tenke på variable på (ulike metaforer)

En navngitt lokasjon i datamaskinens minne hvor en verdi kan lagres



Mest korrekt, men minst nyttig..

En navngitt parkeringsplass hvor en verdi kan parkeres

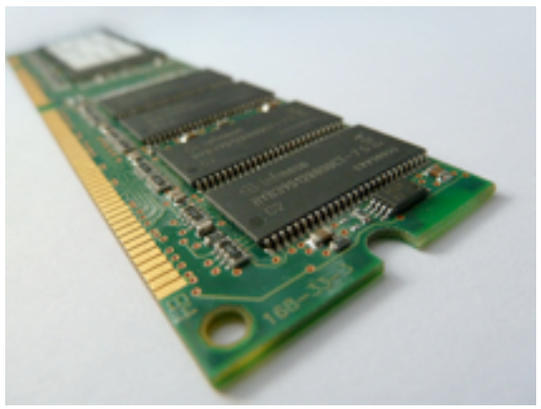


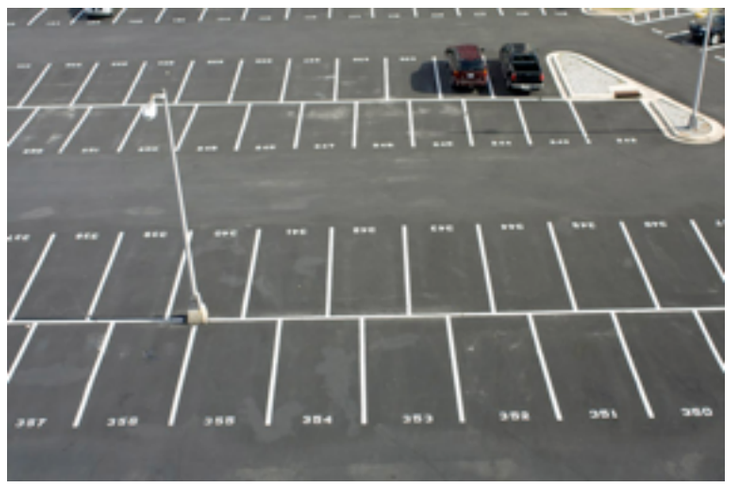
Nyttig metafor, brukt i lærebok

En navnelapp som er satt på en verdi

Nyttig metafor, populær for Python

Tre måter å tenke på variable på (ulike metaforer)





Felles for alle tre metaforer:

- En variabel er et navn som representerer en verdi
- En variabel kan bare holde én verdi av gangen*
- Man kan underveis i et program endre hvilken verdi en variabel representerer

* verdien kan være et sammensatt objekt

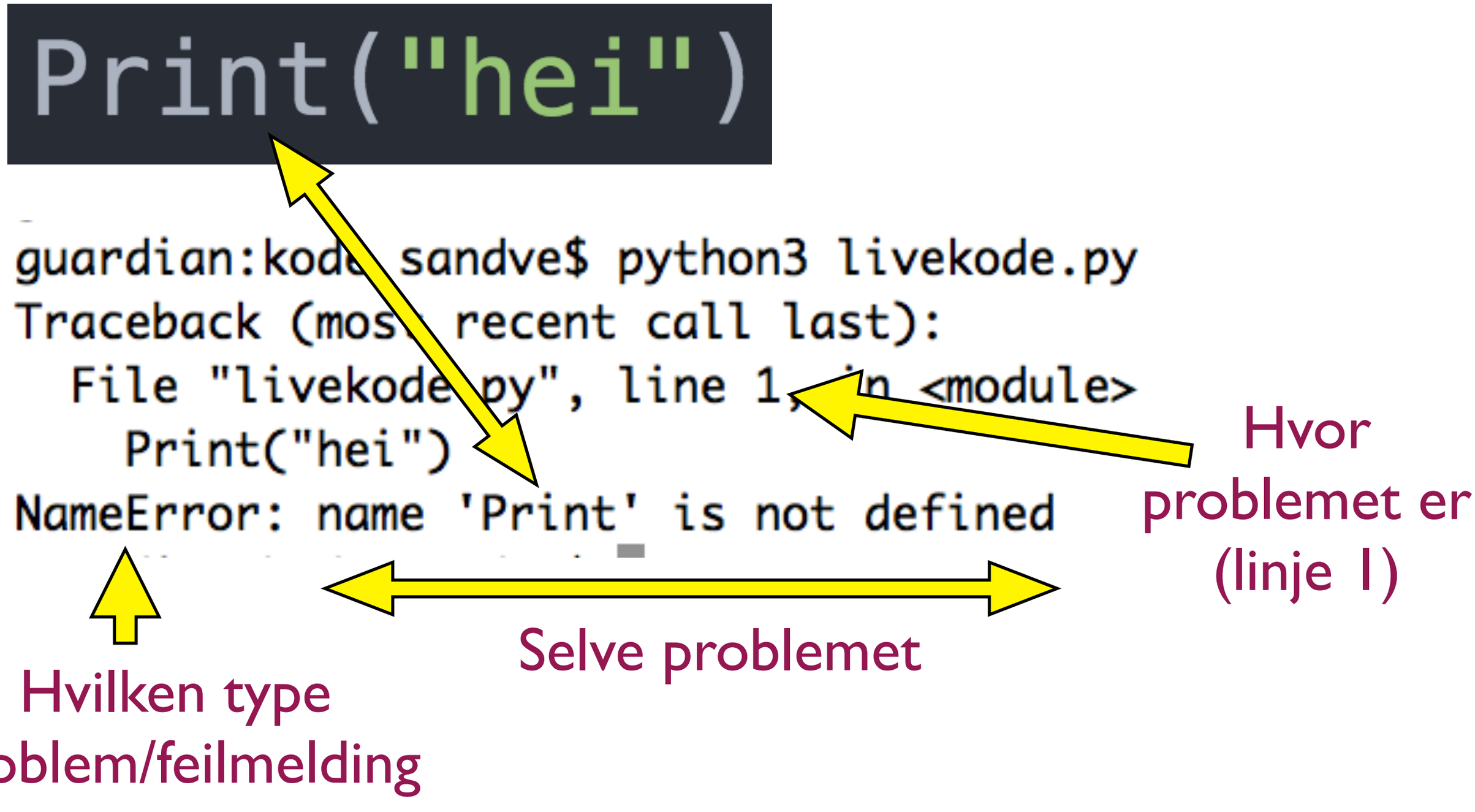
Plan for forelesingen

- Hva er programmering?
- Skrive og kjøre våre første program
- Variabler
- **Feilmeldinger**
- Innlesing fra tastatur
- Beslutninger (if)

Feilmeldinger

- Når noe går galt i et program får man en feilmelding
  - Vi kommer mer tilbake til dette i uke 2..
- Et lite eksempel:
  - Man kjører et Python-program som består av en linje: Print("hei")
  - Dette vil gi en feilmelding bestående av hvor feilen lå (fil og linjenummer), hvilken type feil det var, og akkurat hva som var problemet

Feilmeldinger



Plan for forelesingen

- Hva er programmering?
- Skrive og kjøre våre første program
- Variabler
- Feilmeldinger
- **Innlesing fra tastatur**
- Beslutninger (if)

Slipp brukeren til!

- De fleste program tar en eller annen jevnlig inputt fra brukeren
- Dette gir også mye mer dynamikk i programmene enn det vi har sett på til nå
- Kommunisere med brukeren av et program:
  - **`print`** gir output til bruker
  - **`input`** henter inputt fra bruker
- [innlesing.py]

Gi brukeren beskjed om å skrive inn et svar

- Brukeren må vite at han/hun skal skrive noe:

```
print("Hva heter du? ")
navn = input()
```

- **input** kan skrive beskjed før den venter på svar:

```
navn = input("Hva heter du? ")
```

Plan for forelesingen

- Hva er programmering?
- Skrive og kjøre våre første program
- Variabler
- Feilmeldinger
- Innlesing fra tastatur
- **Beslutninger (if)**

Ikke bare følge strømmen

- For det vi har sett på til nå,
  har programmet alltid fulgt én bestemt sekvens av instruksjoner
  - Vi kan lese ulike startverdier fra tastatur og dermed få ulike resultat, men alltid basert på de samme operasjonene
- Vi trenger mer variasjon!
  - Det blir mye artigere å programmere dersom hvilke operasjoner som utføres også kan avhenge av verdier man regner ut

Et konkret problem som krever beslutninger

- Problemstilling:
  - Vi ønsker å lage et program som kan fortelle hovedstaden til et valgfritt land i nordre Skandinavia (brukeren ber om Norge eller Sverige)
- Vi trenger altså to ulike `print`
- Vi må imidlertid sørge for at kun én `print` blir kjørt, hvor hvilken som blir kjørt avhenger av innlest land

En uferdig løsning (mangler beslutning)

```
land = input("Velg land i nordre Skandinavia: ")

    print("Oslo")

    print("Stockholm")
```

Beslutninger i et program: `if`

- Syntaksen (skrivemåten) er veldig enkel:
  - `if land=="Norge":`
    `print "Oslo"`
- Merk dobbelt likhetstegn
- Merk innrykk
- {hovedstad1.py}

Mange linjer kan styres av samme beslutning (**if**)

- ```
  if land=="Norge"
      print "Oslo"
      print("Kjent som byen ved fjorden")
  ```

- Merk felles innrykk for alle programlinjer som skal styres sammen (kalles en blokk)
- {hovedstad2.py}
  ```

En mer nasjonalistisk variant

- Lage et program som kun ønsker å fortelle hovedstaden til Norge
- {hovedstad3.py}

En praktisk kortform: `if-else`

- En veldig begrenset utviding av syntaks:
  - if land == "Norge":
    print("Oslo")
    else:
    print("Kunne ikke brydd meg mindre!")

- {hovedstad4.py}

Kombinere *ellers* med ny *if*: `elif`

- Kan kombinere if, elif og else:
  - if land == "Norge":
        print("Oslo")
    elif land == "Sverige":
        print("Stockholm")
    elif land == "Danmark":
        print("Kobenhavn")
    else:
        print("Ukjent land")

- *elif* og *else* kjøres kun dersom ingenting før slår til
- {hovedstad5.py}

Man kan nøste sammen flere **if**

```
if alder<6:
    print("Velkommen til bhg")
    if alder<3:
        print("Du er smaabarn")
    else:
        print("Du er storbarn")
```

Man kan nøste sammen flere **if**

Alt dette kun dersom alder<6

```
if alder<6:
    print("Velkommen til bhg")
    if alder<3:
        print("Du er smaabarn")
    else:
        print("Du er storbarn")
```

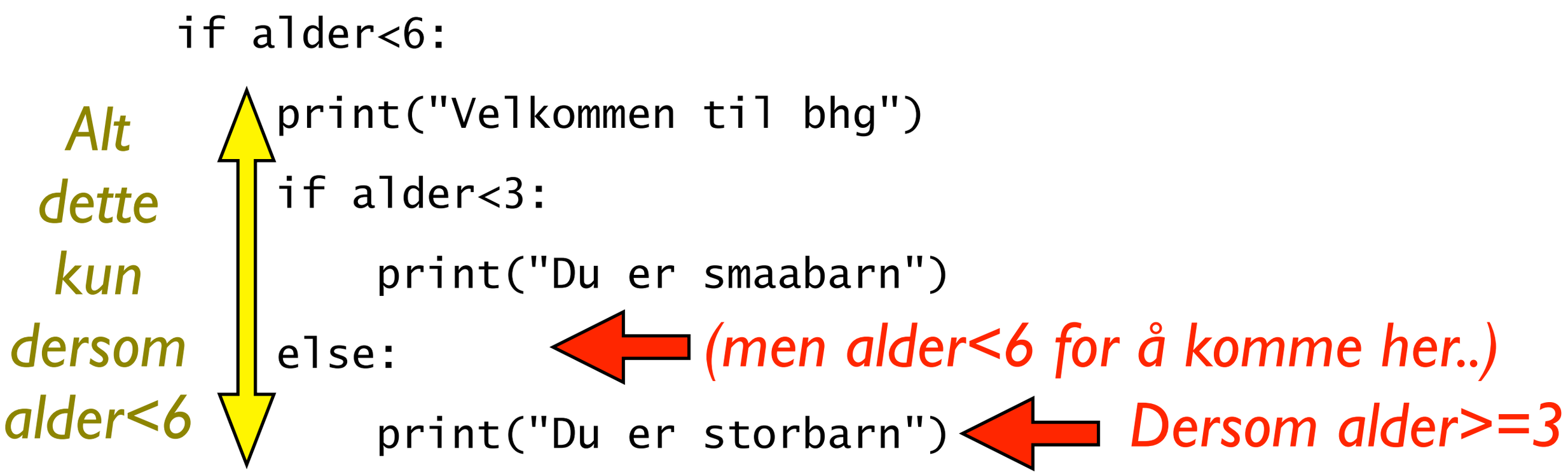
Man kan nøste sammen flere **if**

```
if alder<6:
    print("Velkommen til bhg")
    if alder<3:
        print("Du er smaabarn")
    else:
        print("Du er storbarn")
```

Alt dette kun dersom alder<6

Dersom alder>=3

Man kan nøste sammen flere **if**



Et liten oppgave

- Lag en nøstet beslutning hvor voksne som ikke er gravide får beskjed om å kjøre karusell

```
voksen = input("Er du voksen? (ja/nei)")
gravid = input("Er du gravid? (ja/nei)")

#Nøstet beslutning med utskrift..
```

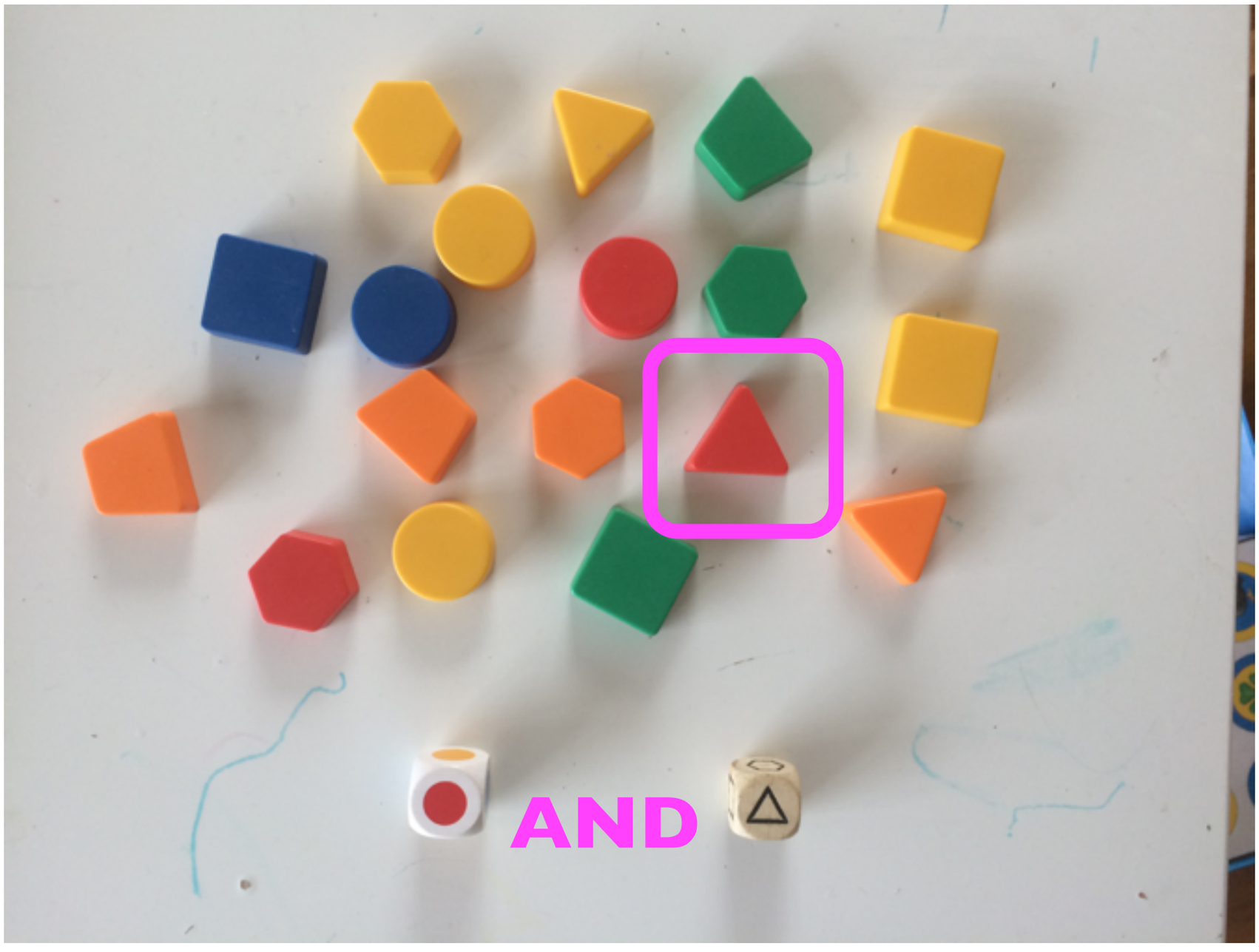
- Hvordan vil du skrive koden?
  - Prøv selv med blyant og papir! (3 minutt)
  - Etterpå diskuter med nabo (3 minutt)

En mulig løsning

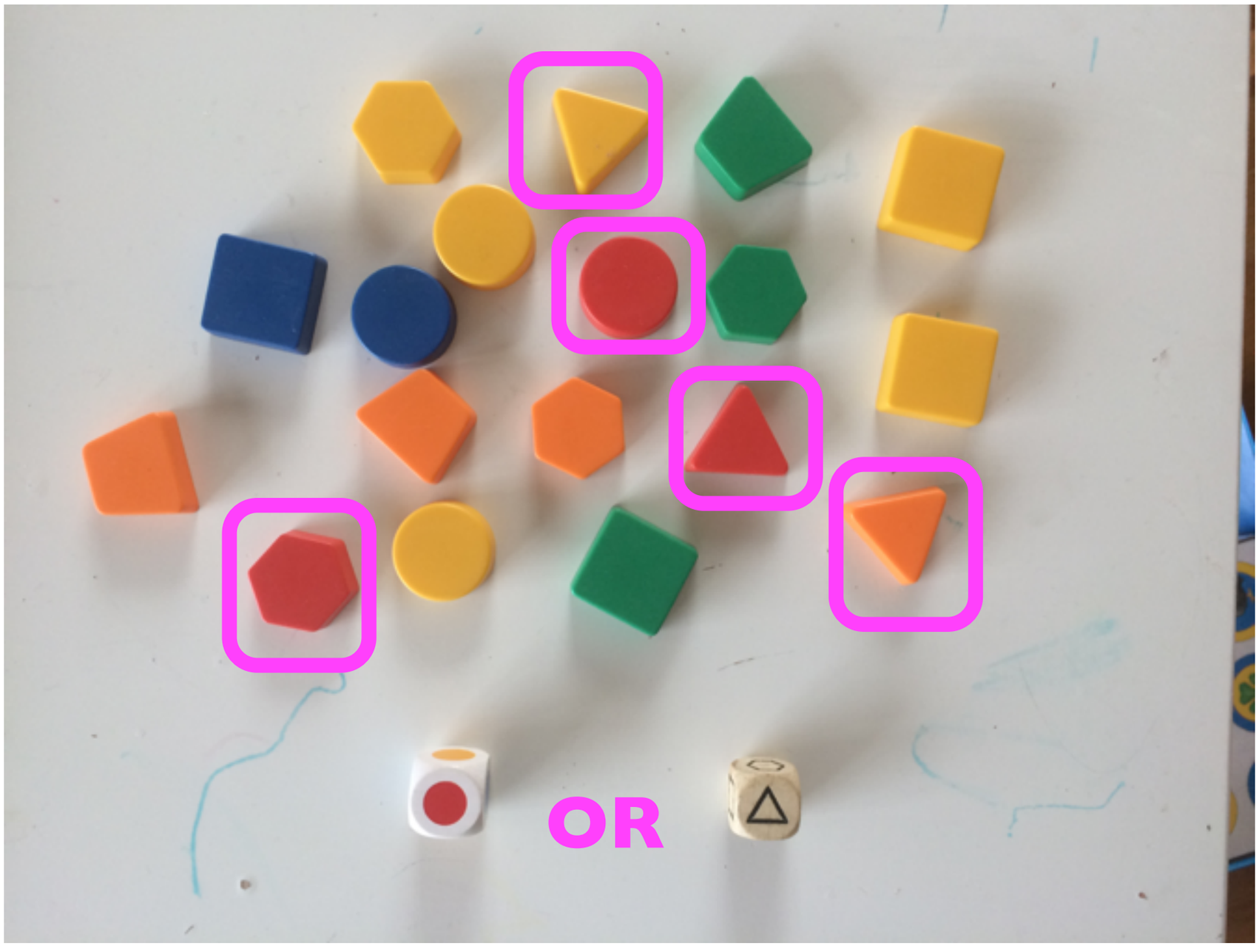
```
voksen = input("Er du voksen? (ja/nei)")
gravid = input("Er du gravid? (ja/nei)")

if voksen=="ja":
    print "Du er stor nok,"
    if gravid=="ja":
        print "men har dessverre ikke lov"
    else:
        print "velkommen ombord!"
```

Og



Eller



Alternativ løsning

```
voksen = input("Er du voksen? (ja/nei)")
gravid = input("Er du gravid? (ja/nei)")

if voksen=="ja" and gravid=="nei":
    print "velkommen ombord!"
```

Skrive et python-program

- Et python-program er bare en helt vanlig tekst-fil
- Man kan skrive koden med en rekke ulike program (editor)
- Det går an i word eller excel, men det er en veldig dårlig idé
- I IN1000 støtter vi primært en editor som heter Atom
- Atom håndtererer innrykk bra (veldig viktig!), gir fargekoding, og er lett å installere
- Det finnes også nettsteder hvor man kan skrive (og kjøre) python-kode uten å måtte installere noe, men det er ofte mer praktisk med lokal editor og kjøring

Kjøre et python-program

- Man kan sette opp kjøring gjennom å dobbelt-klikke i grafisk brukergrensesnitt
- I IN1000 legger vi opp til å i stedet kjøre via kommandolinjen - noe som i lengden er mer praktisk og fleksibelt
- Mange editorer har også mulighet for å kjøre python-filen man jobber på via menyen (og via tastatur-snarvei)

Python-tolkeren (interpreter)

- I stedet for å skrive kode i en fil i en editor, og deretter kjøre filen, kan man direkte kjøre en og en linje i noe som kalles python-tolkeren.
  - Fra kommandolinjen skriver man "python3" for å starte tolkeren
  - Man kan da skrive og utføre en enkelt linje av gangen
  - Man kan også skrive inn et uttrykk (f.eks. regnestykke) og se hva verdien blir

Python-tolkeren (interpreter) (forts)

- Tolkeren er veldig egnet for å detaljert forstå hva som skjer på en enkelt linje - f.eks. se verdier av variabler og uttrykk
- Når man skriver hele programmer (mange linjer) er det derimot mye mer praktisk å skrive og kjøre en fil (gir bedre oversikt og man kan endre en linje og kjøre hele programmet på nytt fra start)

Avluser (debugger)

- En avluser (debugger) lar oss kjøre hele programmer og samtidig ha kontroll på enkeltlinjer
- Svært nyttig for å finne feil i store programsystemer, og potensielt nyttig når man lærer å programmere
  - Men fare for at verktøyet stjeler fokus fra selve kodingen
- Bruk det gjerne, men sørg i tilfelle for at det støtter læring av programmering, ikke distraherer fra det
- Enkel variant å eventuelt prøve ut: `pythontutor.com`

Konklusjon

- Programmering er en form for kreativ problemløsning
- Variabler lar oss ta vare på verdier gjennom et program
- Beslutninger (if) lar programmene utføre ulike operasjoner avhengig av verdien til en variabel