Forkurs IN1900 (m.fl.)

del 2: programmering

Innhold i del 2:

- Litt studieteknikk
- Programmeringsspråk (installere python)
- Editor (installere VS Code), Terminal og Debugger
- Feilsøking (“debugging”)
- Levere en oppgave i Devilry

Litt studieteknikk

- Mange forelesere legger ut forelesningsnotater (slides) før eller etter forelesning – i så fall trenger du ikke notere alt som vises der underveis i forelesningen
- Hvis du programmerer i forelesningen (for eksempel live-koding), kan du skrive forklarende kommentarer direkte i programmet ditt (her i grønt):

```
1    # Venstre side av likhetstegn: variabel
2    # Høyre side av likhetstegn: verdi (eller utregning)
3    beløp = 100
```

Litt studieteknikk

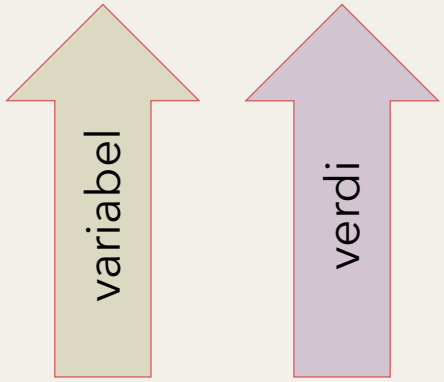
- Programmering handler ikke om å pugge hvordan hver minste ting skrives (syntaks)
- Det viktigste er å forstå hva ting gjør, hva de kan brukes til og ikke brukes til
- På eksamen kan du ha med skrevne notater (inkludert kode-eksempler) og trykte hjelpemidler (inkludert lærebok) slik at du ikke trenger å huske alt utenat
- Men du trenger å forstå hva oppgavene handler om og hva som skjer i maskinen når koden kjøres

Bruk og misbruk av kunstig intelligens (AI)

- Skal du bruke AI, bruk det som et hjelpemiddel til å lære, ikke som en snarvei til å gjøre oppgaver uten å lære noe
- Du får ikke brukt AI på eksamen, da må du stole på det *du* har lært – har du brukt AI som en krykke gjennom semesteret kommer du til å få det vanskelig
- Verdien til AI bestemmes av hva *du* gjør med informasjonen etterpå – hvis du ikke lærer noe av det du får er informasjonen verdiløs
- AI er ofte upålitelig og tar feil – men likevel svært overbevisende. Dette er en skummel kombinasjon…

Å lese programkode – tre nivåer

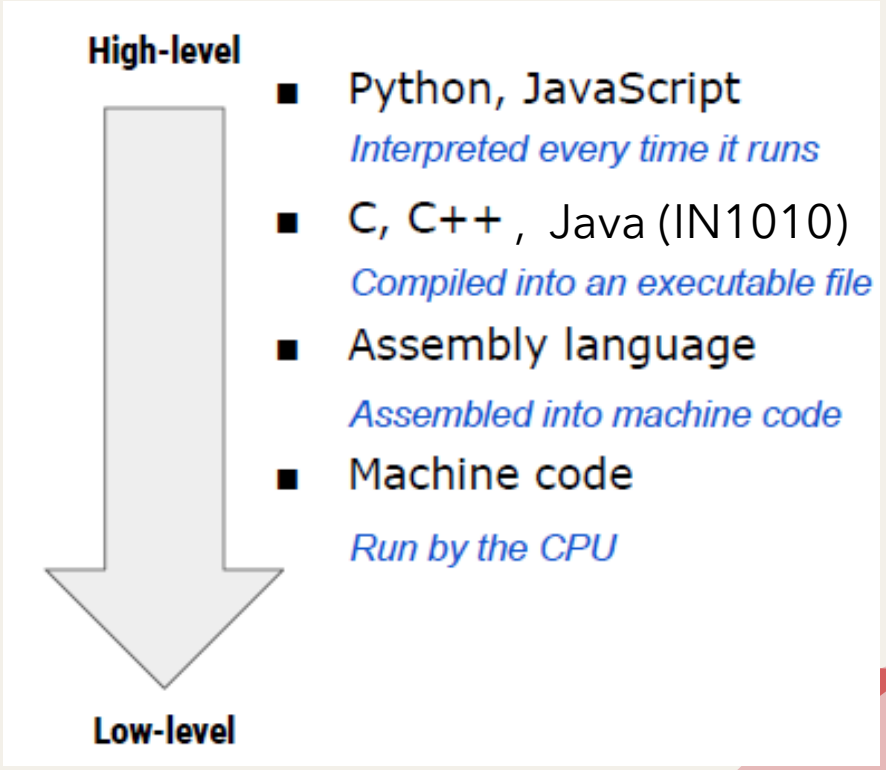
```
1    beløp = 100
2    renter = (5*beløp/100)
3    beløp = beløp + renter
```



1. **Problem-nivået**
   - Hva slags problem er det dette programmet forsøker å løse?
   - Navn på variabler ("beløp", "renter") er hjelpsomme her
2. **Syntaks-nivået: 'gramatikken' i Python**
   - Tallene til venstre er linjenummer, ikke en del av programmet
   - 3 *tilordninger* på formen: *variabel = verdi*
3. **Maskin-nivået: hva skjer når vi kjører koden?**
   - En linje kjøres av gangen
   - Når vi skriver *renter* senere, vil maskinen ha lagret i minnet at dette betyr verdien 5
   - *beløp* får først én verdi (100) og litt senere en ny (105) – da er det den siste verdien som er lagret i minnet, den første dermed er overskrevet ("glemt").

Det finnes forskjellige språk å lage programmer i

- Tolket kode / skript (høynivå): Et program som oversettes til maskinkode hver gang det skal kjøres. Treigest, men også enklest for mennesker.
- Kompilert kode (middels): Et program som oversettes til maskinkode på forhånd av et spesielt program (kompilator). Treigere, men enklere.
- Assembly-kode (lavnivå): Ekstremt detaljert (men raskt), det laveste nivået som er noenlunde forståelig for mennesker (IN1020)
- Maskinkode (helt på bunn): Kjøres direkte på den fysiske datamaskinen (hardwaren) : 00101001…



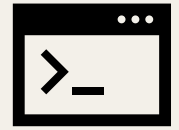
Installere Anaconda (IN1900)

- Python er et høynivå-språk: Ikke så raskt, men lettere å lese og lære seg
- Et av de mest brukte språkene – det finnes mye hjelp og mange programbiblioteker
- Men **python** er også et program som tolker og kjører programmer skrevet i Python
- **anaconda** holder styr på både **python** og programbiblioteker for oss
- Last ned og installer på din lokale maskin her: https://docs.anaconda.com/free/anaconda/install/index.html

Tre grunnleggende verktøy:

- **Editor:** Programmet som vi bruker til å skrive programmer
- **Terminal:** Del av OSet (som vi så på i del 1) hvor vi kjører Python-programmer

  (programmene våre kan også åpne vinduer, men det er mer avansert)
- **Debugger:** Verktøy som lar oss se hvordan programmet fungerer i maskinen, og som hjelper oss med å finne feil (“bugs”)



Editor: Visual Studio Code (VS Code)

- Program som vi bruker til å skrive programmer
- VS Code er den eneste editoren vi anbefaler og gir hjelp med
- (hvis du foretrekker en annen editor kan du bruke den på eget ansvar)
- https://code.visualstudio.com/Download

Oppsett av VS Code

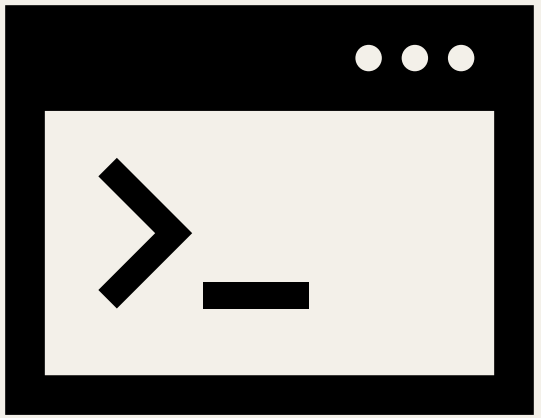
- Enklere, mindre forvirrende
- Bedre tilpasset læring av Python
- Last ned denne filen: https://tinyurl.com/in1900-vscode
- File → Preferences → Settings → 
- Livekoding: Lim inn fra nedlastet fil og lagre innstillingene
- Start VS Code på nytt og sjekk

Live-koding: Lage og kjøre et program

- Bruk din egen maskin
- Lagre programmet ditt som **forkurs.py** et sted du finner det senere
- Åpne denne mappen i **Anaconda Powershell Prompt** og kjør programmet i dette terminalvinduet med
  python forkurs.py
- Hvis du kjører direkte i VS Code, vil VS Code åpne et terminalvindu i mappen hvor filen du kjører befinner seg (Windows: uten PowerShell, så "ls" og "pwd" vil ikke virke)

Hvorfor kjøre programmer fra terminalen?

- Er det slik at alle som skal kjøre programmene våre må ha en editor (VS Code) installert?
- Nei!
- Du eller noen andre kan kjøre dem på en annen helt uavhenging av om det finnes en editor på maskinen
- Men **python** må være installert!



Debugger: Python Tutor

- Hjelper oss å forstå hva som skjer i maskinen når vi kjører programmet
- En og en linje kjøres av gangen
- Vi kan se variabler og verdiene deres for hvert steg
- Hjelpsomt når vi skal oppdage og rette opp feil
- (VS Code har også en innebygd debugger som kan brukes i selve editoren)

Feil ("bugs") er en helt naturlig, uunngåelig del av programmering (selv etter mange år!)

- Det er umulig å skrive feilfri programmer på første forsøk – syklusen er:
- Skriv litt kode (ikke for mye)
- Test at det fungerer
- Ofte er det en bug: Finn → forstå → fiks
  Bruk debugger!
- Test igjen etterpå – det er ofte flere bugs
- Når alle er funnet, kan vi gå videre og skrive mer kode
- Å oppdage feil er en **god ting** ☺
  (Det er mye verre om vi ikke oppdager dem)
- Det gjelder også "bugs i forståelse": Test også forståelsen deres ofte!

Etterpå: Levere oppgave i Devilry

- Bruk din egen maskin og gå til devilry.ifi.uio.no
- Du må ha fylt ut nettskjema fra del 1
- Vi skal nå levere filene vi laget på del 1 og del 2 slik at gruppelærere kan gi tilbakemelding på dem
- Da er det lurt å legge alt som skal leveres i en mappe og levere alle filene fra den mappen
- Husk at koden som leveres skal kunne kjøre på en annen maskin – lever alle filene som trengs for å kjøre den (men ikke mer enn det)