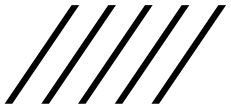
Før vi starter…

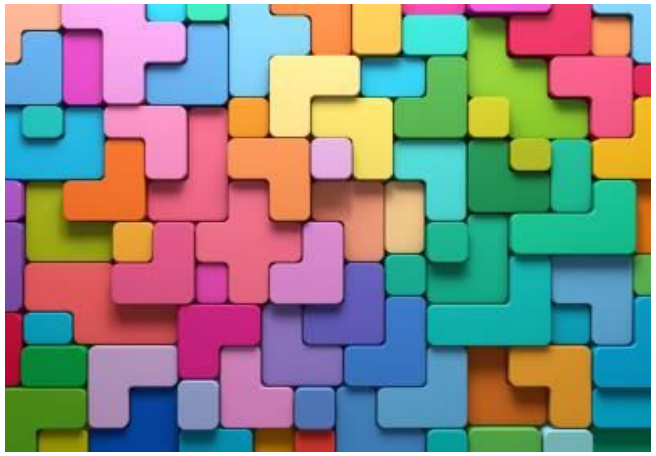
- Logg inn på github.uio.no med UiO-bruker og passord (hvis du ikke har gjort det før, eller er usikker)
- Dette må være på plass før fredag
- Last også ned oppdaterte innstillinger til VS Code (og lagre dem et sted du finner dem når vi kommer litt lenger i forelesningen)

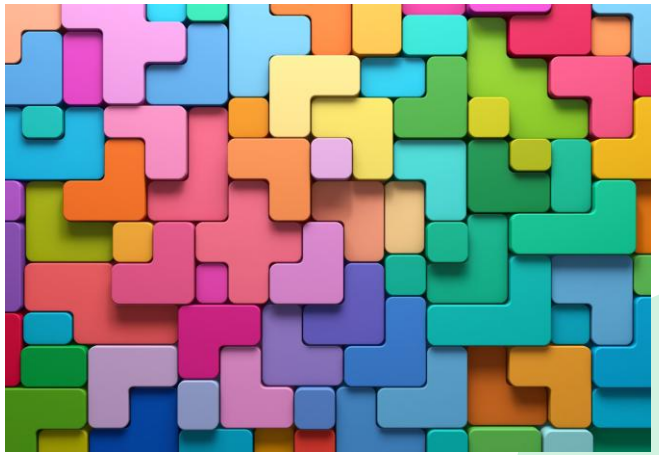


VELKOMMEN TIL IN1910

FORELESNING 1

MANDAG 18/8





Forventninger til dere

- Jeg forventer at dere kommer på forelesning selv om det ikke er obligatorisk
- Jeg forventer at dere deltar i aktivitetene her
- Jeg forventer og håper at dere trives og opplever at det er nyttig at dere kommer hit, både for dere selv og medstudentene
- Jeg forventer at dere kommer på gruppetimene for å jobbe med prosjekter og oppgaver, og spør om hjelp når det trengs
- Jeg forventer at dere leverer 100% deres eget arbeid (og opplyser om det i motsatt fall)

Bruk av KI

- Bruk gjerne KI som en diskusjonspartner eller ekstra gruppelærer, som et verktøy til læring
- Jeg kommer til å gi eksempler på dette i forelesningene

Motivasjon: Hvorfor er vi her?

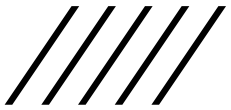
- Det finnes mange eksempler på forskere som er dyktige i faget sitt
- Og som kan lage programmer som gjør det de skal
- Men programmene kan være vanskelige for andre å lese eller bruke
- Eller vanskelige å endre / bygge på (til og med for oss selv!)
- Vi er her for å lære å skrive vitenskapelige programmer på en måte som gjør disse tingene lette for både andre og oss selv!

Best Practices for Scientific Computing

1. Skrive programmer så mennesker forstår dem, ikke bare datamaskinen
2. La datamaskinen gjøre det datamaskinen er god til (repetere, automatisere)
3. Gjøre endringer i små steg av gangen (og kunne gå tilbake hvis det oppstår problemer)
4. Unngå å gjenta det du selv (eller andre) har gjort før

Best Practices for Scientific Computing

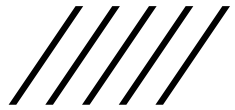
5. Planlegge for at det oppstår feil (testing, debugging)
6. Gjøre programmet raskt først når det virker som det skal
7. Dokumentere design og formål, ikke tekniske finurligheter (*hva* koden gjør, ikke *hvordan* den gjør det)
8. Samarbeide for å finne feil og holde orden på dem



- Læremål

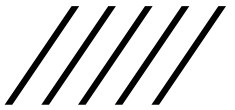
Forkunnskaper

- Vi antar du har brukt Python før, men ikke nødvendigvis på en stund
- Eventuelt at du har programmert i et annet programmeringsspråk
- For dere som ikke har brukt Python før eller på en stund, anbefaler vi å bruke ekstra tid på emnet, slik at du får repetert det grunnleggende i Python
- (Hvis du *aldri* har programmert før, er IN1900 eller IN1000 et mye, mye bedre sted å starte!)



Hva er pensum?

- Forelesningene: se timeplanen på emnesiden
- Alle de fire prosjektene (unntatt frivillige oppgaver)



Emnesiden

- All informasjon blir lagt ut her, og vi forventer at dere til enhver tid følger med på det som blir publisert (og eventuelt endret) gjennom hele semesteret
- Timeplan (med forelesningsnotater og opptak)
- Beskjeder (med info om prosjektene og eksamen)
- Ukesoppgaver og andre ressurser

Prosjektene

- Prosjekt 0 kun individuelt
- Prosjekt 1 kun i grupper på 2 (kan spørre om å få jobbe sammen med en dere kjenner, men ingen garanti)
- Prosjekt 2 og 3: valgfritt individuelt / grupper
- Forelesninger på norsk, prosjektene på engelsk (det finnes heldigvis ordlister med oversettelser)
- Du får ett forsøk på å levere hvert prosjekt
- Eventuelle utsettelser på mer enn 3 dager må søkes om hos studieadministrasjonen (søk i så fall så tidlig som mulig)

Grupper (prosjekt 1 og utover)

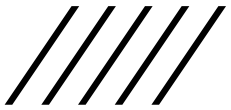
- Det er lov å få hjelp/tips av andre grupper, men hver gruppe må skrive og levere sitt eget arbeid
- Muntlig eksamen er individuell – dvs. alle på en gruppe må kunne forklare alt dere har gjort
- Det går an å jobbe individuelt og gjøre alt selv også i en gruppe, men man må fortsatt levere samlet og sammen velge hva dere vil levere
- Begge i en gruppe får samme poengsum på prosjektet

Støttelitteratur

- Ikke pensum, men kan være til hjelp for å forstå ting bedre
- Foreslått støttelitteratur finnes her
- OBS: Annen info på de sidene kan være utdatert og gjelder ikke nødvendigvis for dette semesteret
- Hvis du er i tvil så er det alltid forelesningsnotater og beskjeder fra emnesiden som gjelder

Når er innleveringsfrister / eksamen?

- Prosjekt 0: fredag 5. september kl. 23:59
- Prosjekt 1: fredag 26. september kl. 23:59
- Prosjekt 2: fredag 24. oktober kl. 23:59
- Prosjekt 3: fredag 14. november kl. 23:59
- Muntlig eksamen: i perioden 1. – 15. desember
  (vi unngår kollisjoner med andre eksamener du har)
- Følg med på emnesiden for publisering av prosjekter og frister!



Fem ulike måter å forstå på

- Å kunne **forklare** til noen andre hvordan noe virker (dokumentasjon i prosjektene, muntlig eksamen)
- Å kunne **tolke** seg frem til hva som er relevant akkurat her (muntlig eksamen)
- Å kunne **bruke** det du har lært (prosjektene)
- Å kunne se ting fra flere **perspektiv**: utvikler, tester, bruker (muntlig eksamen)
- Å forstå hva du forstår og ikke forstår (**metakognisjon**) (muntlig eksamen)

Vurdering i IN1910

Mappeevaluering samt muntlig eksamen. Studentene utvikler og leverer et større programsystem med tilhørende dokumentasjon, og alle vil bli kalt inn til en muntlig høring hvor de må demonstrere at programmet virker og forklare sentrale deler av programmet og den underliggende matematikken.

Alle innleveringene må være godkjente for å få bestått karakter i emnet. Både mappen og den muntlige eksamenen må være bestått for å få bestått karakter.

Bestått/ikkje bestått

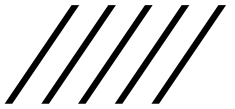
Karakterskalaen med bestått og ikkje bestått er eit sjølvstendig vurderingsuttrykk. Karakterskalaen er utan samanheng med bokstavkarakterane, og karakterane blir ikkje konvertert i samband med godkjenning.

Hvordan bestå prosjektene?

- På hvert prosjekt er det en liste over "døddsynder" som resulterer i at du får ikke bestått på prosjektet (og kurset)
- Ellers får du et visst antall poeng – for å bestå prosjektet må det være mulig for deg å få 50 poeng totalt på alle prosjektene (hvis du har 20 poeng totalt og får 20 på siste prosjekt, har du ikke bestått det siste prosjektet, for da kan du ikke oppnå 50 poeng)
- Det gis ett forsøk på hvert prosjekt

Hvordan bestå prosjektene?

- Du får ikke poeng for, eller tilbakemelding på, AI-generert kode. Du må opplyse om alt du har brukt AI til.
- Bruker du AI uten å opplyse om det, vil du få 0 poeng på prosjektet, uavhengig av hva du ellers har gjort
- I tvilstilfeller vil du bli kalt inn til en kort samtale med faglærer
- TLDR: Gjør mest mulig selv og opplys om alt dere ikke gjør selv (det lønner seg også stort i forhold til eksamen)



Kommunikasjon

- Vi bruker Mattermost til kommunikasjon utenom forelesning og gruppetimer (logg inn med UiO-brukernavn og passord) OBS: Krever at du setter opp tofaktor-autentisering
- Du skal automatisk bli lagt til IN1910-gruppen
- Her er emneansvarlig, gruppelærere og andre studenter
- Her kan man ønske seg ting forklart til kommende gruppetimer og forelesninger

Tilbakemeldinger

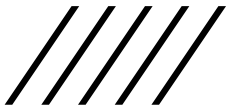
- Vi vil gjerne høre fra dere hvis dere har tanker eller forslag
- Du kan kontakte emneansvarlig direkte på Mattermost eller be gruppelærere melde videre
- For alvorlige ting, anbefaler vi at du går til studieadministrasjonen eller kontakter et av fagutvalgene (FUI, Matematisk Fagutvalg)

Støttelitteratur

- Ikke pensum, men kan være til hjelp for å forstå ting bedre
- Foreslått støttelitteratur finnes her
- OBS: Annen info på de sidene kan være utdatert og gjelder ikke nødvendigvis for dette semesteret
- Hvis du er i tvil så er det alltid forelesningsnotater og beskjeder fra emnesiden som gjelder

Hva slags programvare trenger jeg?

- Python-distribusjon: Anaconda
- Editor: Visual Studio Code (VS Code)
  (eller en annen en du liker – men da får du ikke sett på forelesning hvordan man gjør ting direkte i din editor)
- Terminal: PowerShell / bash
  Dette har du allerede på maskinen!
- GitHub Desktop (denne installerer vi sammen på fredag)
- (Senere i semesteret: En kompilator til C++)



Live-koding

- Det vi skal gjøre etter pausen
- Vi skriver alle den samme koden, jeg på storskjerm og du på din egen maskin (så du har din egen kopi av koden du kan beholde etterpå)
- Hvis du står fast, får feilmelding eller lignende:Se på opptaket og finn feilen senere – det er OK å gå videre selv om det ikke funker, ikke bruk tid på å finne feilen her og nå
- Rekk opp en hånd hvis du synes det går for fort!
- (Bruk opptaket til å livekode senere hvis du er borte en dag)

Pause

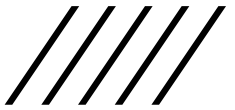
- Etter pausen skal vi repetere litt Python (live-koding)
- For å komme inn i en god vane med å bruke terminalen, anbefales det å legge inn følgende innstillinger i VS Code:

Windows: File → Preferences → Settings → 

Mac: Code → Settings → Settings → 

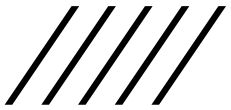
Litt om denne repetisjonen

- Dette er *ikke* en fullverdig introduksjon til Python
- Friske opp det dere har lært som kanskje ikke har vært brukt på en stund



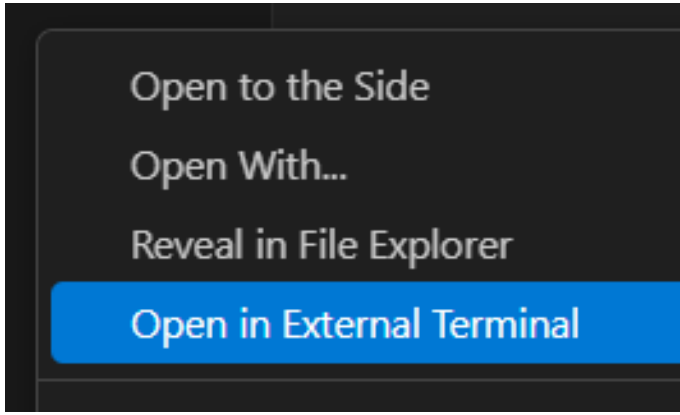
Live-koding

- Hvis du nå har en editor (f.eks. VS Code) og Python (helst Anaconda, som du uansett trenger snart), bli med på live-kodingen
- Hvis du mangler editor / Python anbefales det å gå gjennom opptaket i ettertid så du får prøvd live-kodingen selv
- Og du bør få editor og Python på plass så fort du kan
- Generelt når vi live-koder vil mye av det vi gjør ikke være med på lysarkene fra forelesningen (se heller på opptak i disse tilfellene)



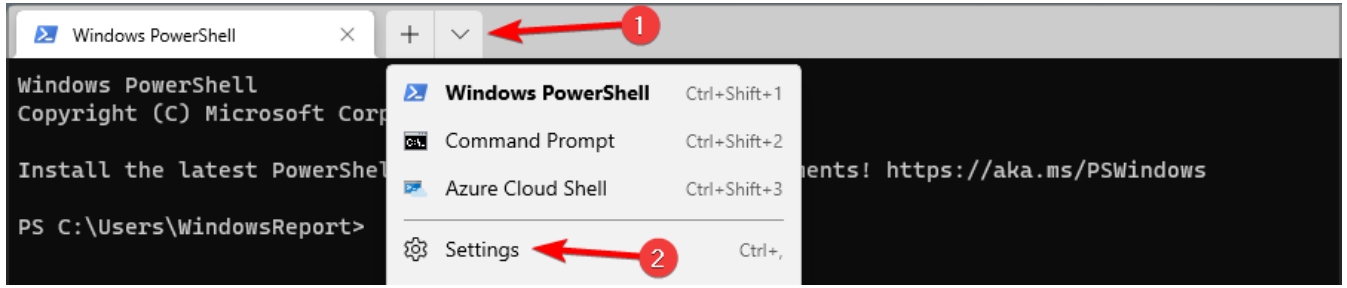
"Hvorfor bruker du ikke play-knappen?"

- Fordi det er en dårlig vane å teste programmet i editoren du bruker til å skrive koden med
- Brukeren av programmet ditt skal bare trenge en terminal og de riktige bibliotekene
- Det hjelper lite å si "ja, men det virker jo på min maskin" hvis brukeren (eller retteren) ikke klarer å kjøre koden din
- Derfor: Bruk terminalen til kjøring og testing!

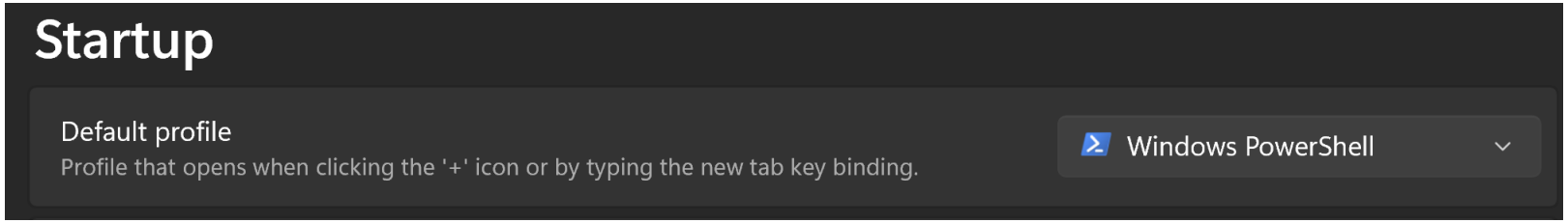


Hvis du bruker Windows…

- Står det Windows PowerShell øverst i vinduet?
- Hvis nei, trykk på:

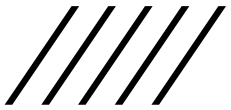


- Velg Windows PowerShell som standard, lukk terminalvinduet og åpne det på nytt



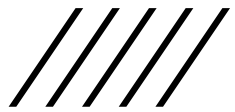
Anaconda i terminalen

- Ser du (base) i terminalvinduet?
- Hvis ikke
- PowerShell (administrator):
  set-executionpolicy remotesigned
- Anaconda Prompt:
  conda init powershell (Windows)
  conda init (Mac/Linux)
- Åpne nytt terminalvindu fra VS Code



Noen begreper

- *Variabler* har *verdier*
- Alle verdier i Python er *objekter* som finnes i minnet til datamaskinen et sted
- Så en variabel er et navn som vi gir til et objekt
- Du har egentlig brukt objekter (verdier) i Python fra dag 1
- Objekter kommer i ulike *typer* (**int**, **str**, **float**, **list**…)
- Programkoden som beskriver hva en type objekt kan gjøre (og hvordan) kaller vi for en *klasse*

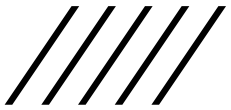


Ikke-muterbare typer

- Objekter av en *ikke-muterbar* type (eller klasse) kan ikke endres
- Når jeg har definert **a = 4** kan jeg ikke endre verdien av 4 etterpå
- Hvis jeg etterpå skriver **a = 5** har jeg ikke endret 4, dette altså et *annet* objekt (som ligger et annet sted i minnet)
- Jeg kan endre hvilket objekt variabelen **a** står for, men selve objektene 4 og 5 endres ikke
- Eksempler: **bool**, **int**, **float**, **str**, **tuple**
  (en **tuple** er en slags liste som ikke kan endres)

Muterbare typer

- Objekter av en *muterbar* type (eller klasse) kan derimot endres underveis i programmet
- Når jeg har definert **a = []** er listen tom til å begynne med
- Hvis jeg etterpå skriver **a.append(1)** er det fortsatt det samme objektet (på samme sted i minnet), men det er endret fra **[]** til **[1]**
- Eksempler: **list**, **dict**, **set**, …
- "Men har alt dette noe å si i praksis, da…?"



Noen flere begreper

```
1   def f(x):
2       return x*2
3
4   y = f(2)
```

Parameter	**x**
Argument (inndata)	**2**
Returverdi (utdata)	**4**

2 → f(x) → 4

while-løkker

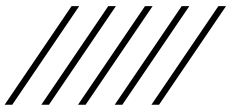
- mest generell, kan brukes i alle situasjoner (selv når vi ikke vet antall repetisjoner på forhånd)
- kan lett gå uendelig om vi ikke passer på

while <en test returnerer True>**:**

    <kode som repeteres>

    <kode som repeteres>

<kode som ikke repeteres (kjører etter at løkken er ferdig)>



for-løkker

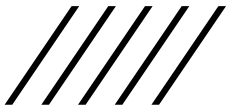
- mer spesialisert løkke, krever en *samling* av objekter (**list**, **tuple**, **dict**, **range**…)
- går gjennom alle disse objektene (*elementene*) i samlingen

for <element> **in** <samling>**:**

    <kode som repeteres>

    <kode som repeteres>

<kode som ikke repeteres (kjører etter at løkken er ferdig)>



Bygge en liste med løkke

[<uttrykk> **for** <element> **in** <samling>**]**

er en snarvei for å bygge en liste på denne måten:

<liste> **= []**
for <element> **in** <samling>**:**
    <liste>**.append(**<uttrykk>**)**

Hente indeks og element samtidig

for <indeks>**,**<element> **in enumerate(**<samling>**):**

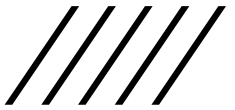
    <gjør noe med indeks og element>

er en snarvei for:

for <indeks> **in range(len(**<samling>**)):**

    <element> = <samling>**[**<indeks>**]**

    <gjør noe med indeks og element>

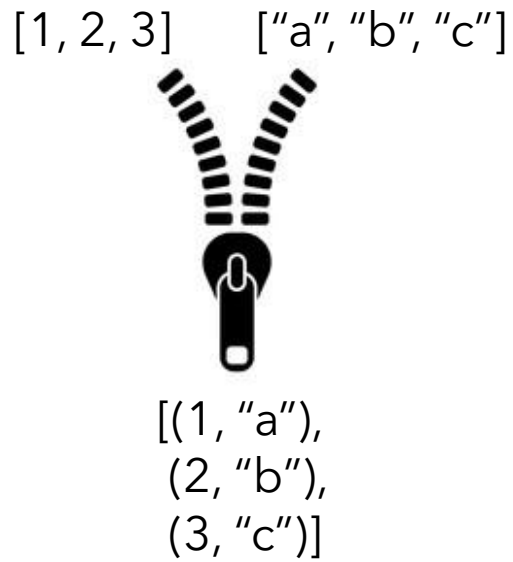


Gå gjennom *flere* samlinger på en gang

for <element1>**,**<element2> **in zip(**<samling1>**,** <samling2>**):**

    <gjør noe med de to elementene>

- Lager en liste med **tuple**r
- Kan også gjøres med 3 eller flere samlinger
- Alle samlingene bør ha like mange elementer



Matematisk eksempel: Forward Euler

$$\frac{du}{dt} = -au(t)$$

$$\frac{\Delta u}{\Delta t} = -au(t)$$

$$\frac{u_{i+1} - u_i}{\Delta t} = -au_i$$

$$u_{i+1} = (1 - a\Delta t)u_i$$

Registrere oppmøte

- https://nettskjema.no/a/537351
- Du kan også svare dersom du har sett forelesningen i opptak

Etter forelesningen

- Det vi ikke rakk å live-kode i dag kan du selv friske opp her
- På fredag begynner vi å se på versjonskontroll med GitHub
- Da starter vi å jobbe med prosjekt 0 her i forelesningen
- Imens, husk å installere det du eventuelt mangler av:
  - Anaconda
  - editor (gjerne VS Code, da får du mye gratis i forelesningene)