IN1000
Introduksjon til objektorientert programmering

Uke 1

Velkommen til Ifi – og IN1000!

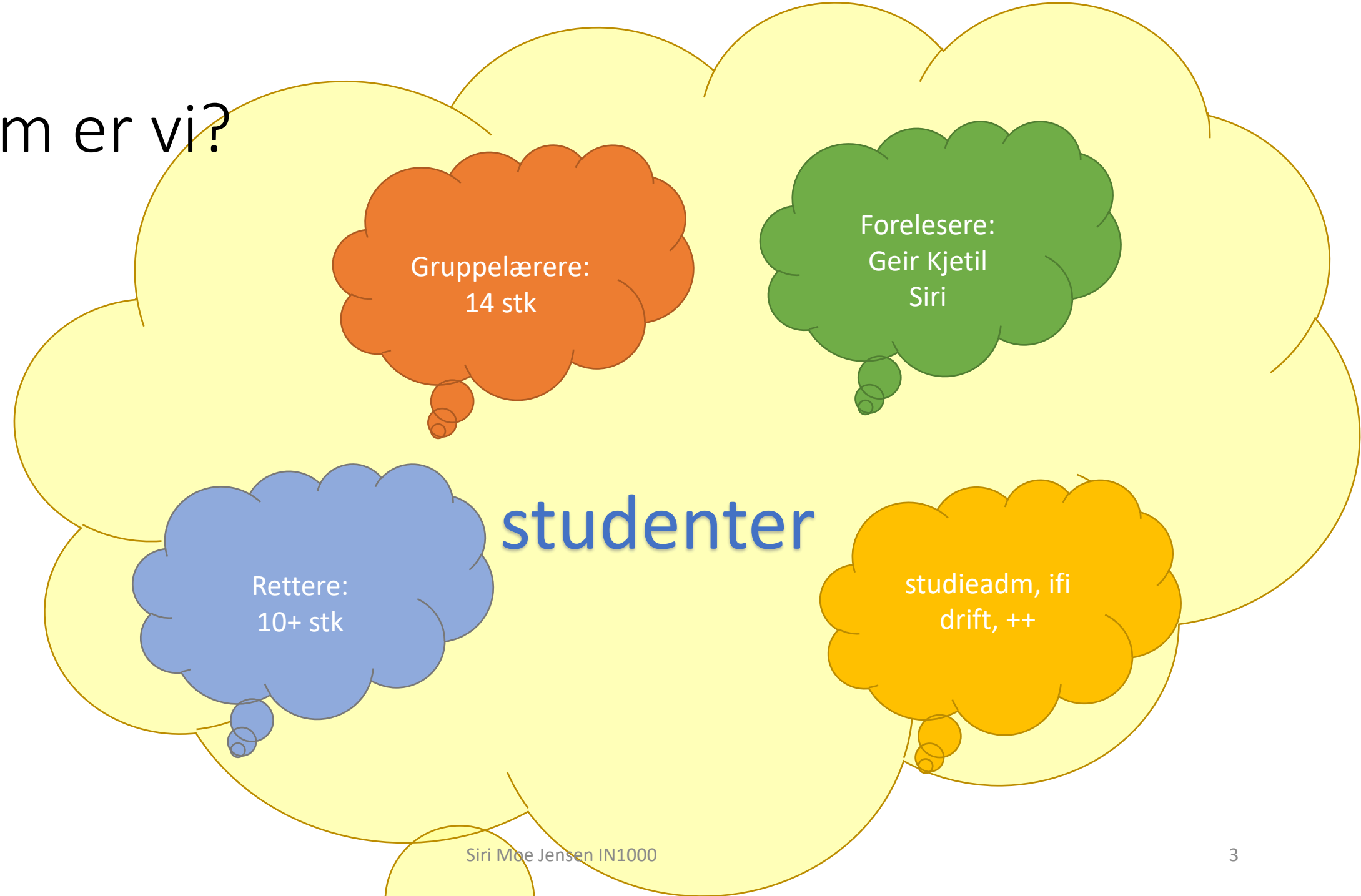
Alle muligheter til en “ny start”

Du jobber nå kun for deg selv –

hvor står du, hva trenger du, hvordan lærer du?



Hvem er vi?


Gruppelærere:
14 stk
Forelesere:
Geir Kjetil
Siri
Rettere:
10+ stk
studenter
studieadm, ifi
drift, ++

Mentimeter

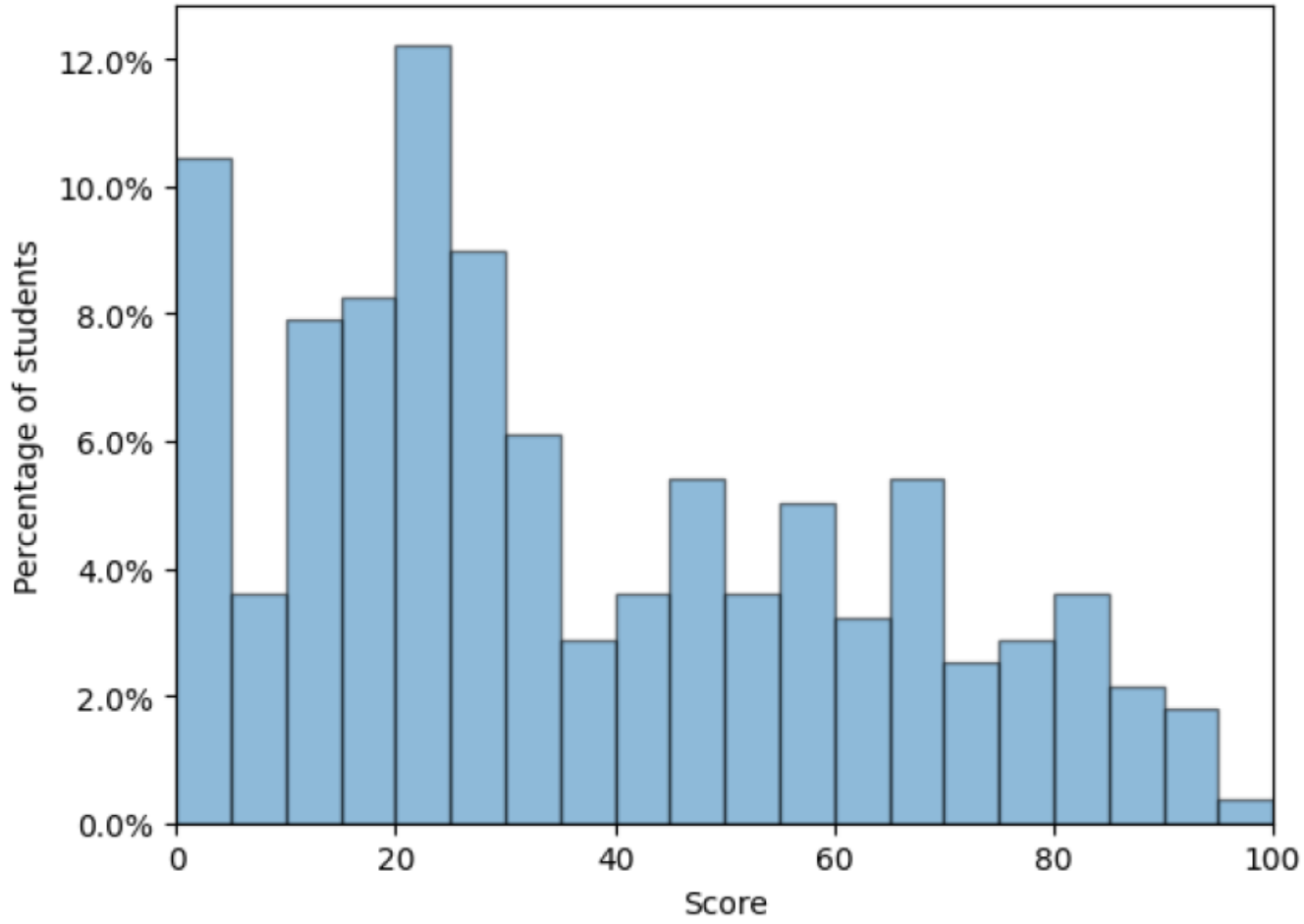
Hvor enig er du i følgende?



Resultater fra tidligere IN1000-forelesning

14 325

Årets forkunnskapstest (IN1000 forkurs, n=300)



Mer frihet, mer ansvar

IN1000 er et introduksjonsemne, mange uten studieerfaring. Du får:

- Oversikt over pensum og forventet progresjon
- Vurdering med regelmessige påminnelser og noen milepæler som må passeres
- Muligheter for å teste deg selv og få tilbakemeldinger gjennom hele semesteret
- Fleksible undervisningstilbud og ressurser

Men:

- Du må selv *sette deg inn i* hva som kreves og når
- .. *følge med* på om du har nødvendig progresjon og hvordan du lærer best
- .. *tilpasse* innsatsen og bruk av tilbudene så du oppnår det du vil

Frivillig betyr ikke unødvendig!

I dag

- Informatikk-faget og noen sentrale begreper
- Praktisk informasjon og studietips for IN1000
- Første innføring i programmering

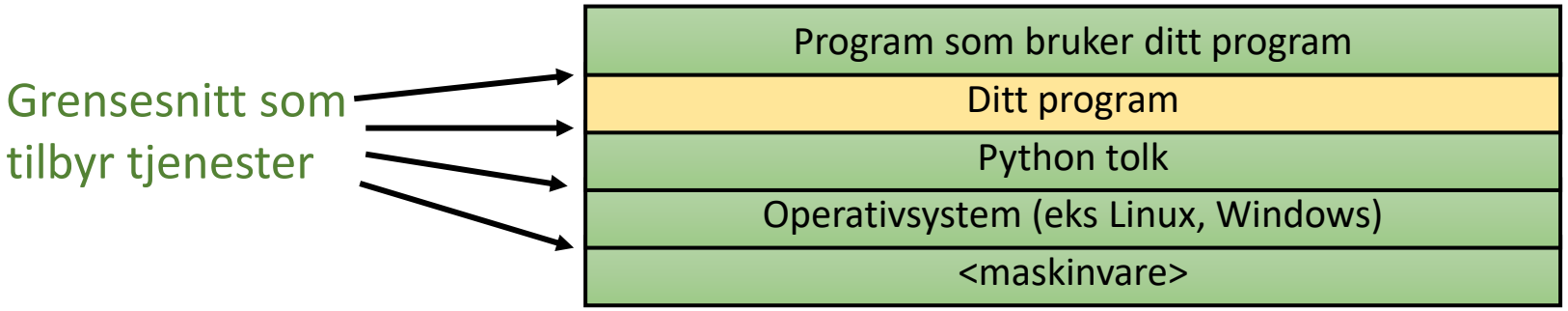
Informatikk

Informatikk er læren om hvordan datasystemer konstrueres og brukes.

Et datasystem består av en eller flere datamaskiner som kjører programvare og kan være knyttet til et eller flere nettverk for overføring av data

Datamaskin: Lagdeling og grensesnitt

Din laptop



- Hvert lag har sine arbeidsoppgaver, og tilbyr disse som tjenester gjennom et grensesnitt til laget over

- Grensesnittet gjør det mulig å abstrahere bort (ikke kjenne til) *hvordan* laget "under" løser oppgavene sine

Abstrahere/ abstraksjon

Wikipedia:

- "..tenkemåter der en skiller ut enkelte elementer, for å betrakte resten.."
- "..Det er en måte å komme fram til fellestrekk eller felles egenskaper og klassifiseringer ..."

Store Norske Leksikon:

- "å sammenfatte enkeltfenomener under ett begrep idet man ser bort fra avvikende detaljer"

Modell

SNL:

<...> brukes ordet om en skjematisering eller en matematisk modell

Slik brukt innebærer en modell alltid en forenkling basert på et utvalg av de viktigste trekk.

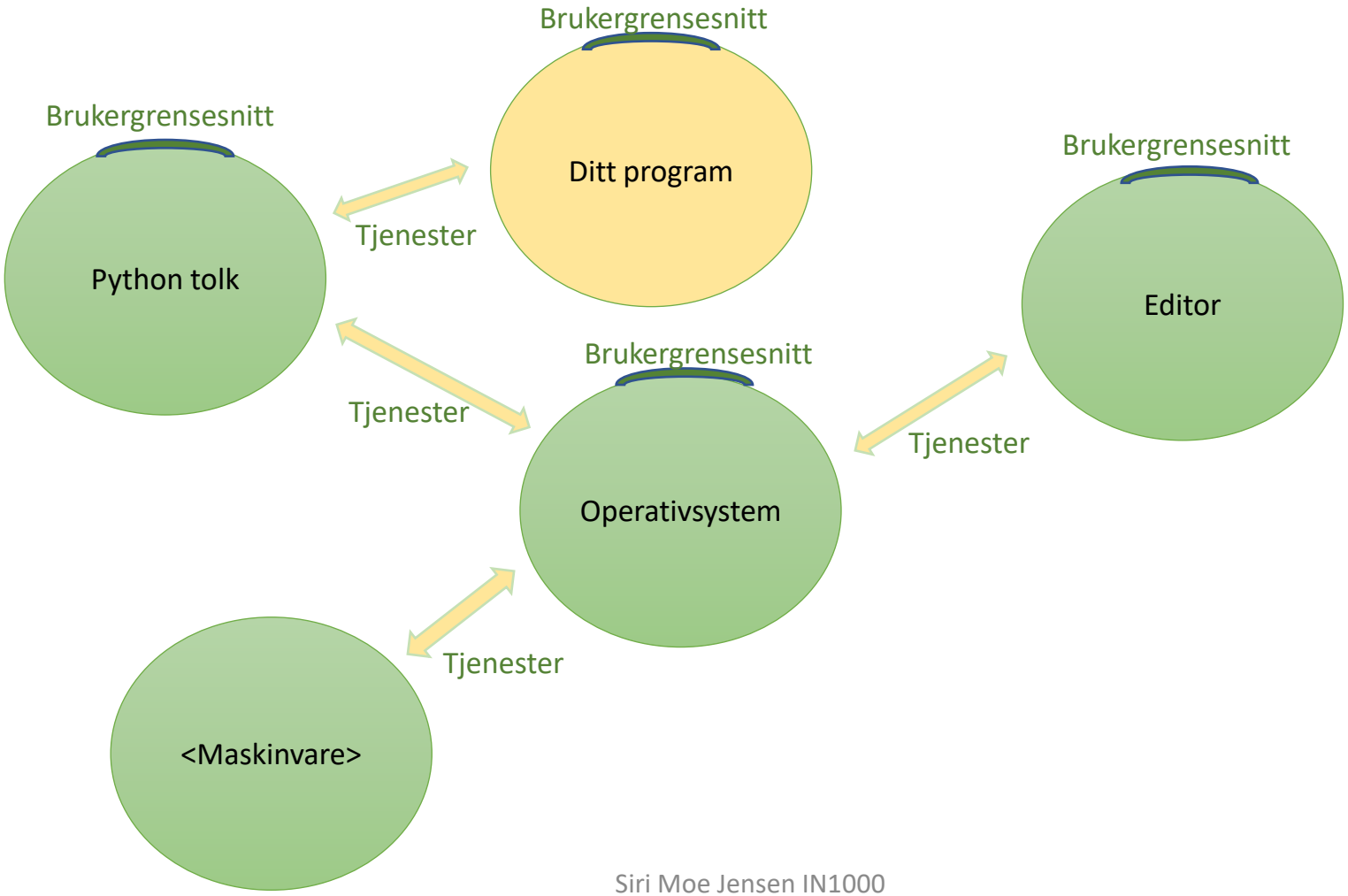
Wikipedia:

Teoretiske modeller gir et sterkt forenklet bilde av virkeligheten.

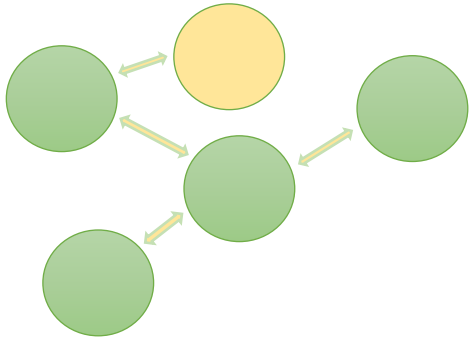
Forenklingen dreier seg om å fokusere på de trekk som anses å være vesentlig for det som skal forklares, samtidig utelukke alt som er uvesentlig



En annen modell av hvordan elementer i et system samarbeider



Objektorientert programmering



- Når programmet ditt utføres, foregår det (bak kulissene) vha objekter som samarbeider.
- Et objekt holder rede på data som hører sammen, og kan utføre handlinger knyttet til disse
- *Hvilke* data og handlinger er definert i objektets klassedefinisjon
- Objektene tilbyr tjenester via metoder til andre deler av programmet gjennom sine grensesnitt
- Objektorientering hjelper oss å organisere programmene våre bedre, ved å modellere og simulere komplekse systemer

I IN1000 lærer du hvordan vi kan opprette og bruke objekter, og hvordan vi kan skrive egne klasser som mønstre for objekter!

Oppsummering

- Informatikk *krever* at vi abstraherer bort forskjeller og kompleksitet vi ikke trenger i det vi jobber med "akkurat nå"
- For eksempel vha modeller og metaforer som beskriver hva som skjer når programmene våre utføres
- Et fenomen kan modelleres på ulike måter avhengig av hvilke egenskaper/ aspekter man er interessert i (trenger å vite noe om)
- Objektorienterte språk har mekanismer som hjelper oss med dette når vi skriver programmer

Oppsummering

- Informatikk *krever* at vi abstraherer bort forskjeller og kompleksitet vi ikke trenger i det vi jobber med "akkurat nå"
- For eksempel vha modeller og metaforer som beskriver hva som skjer når programmene våre utføres
- Et fenomen kan modelleres på ulike måter avhengig av hvilke egenskaper/ aspekter man er interessert i (trenger å vite noe om)
- Objektorienterte språk har mekanismer som hjelper oss med dette når vi skriver programmer

“Hvorfor … når KI?”

Det er vanskelig å spå – særlig om fremtiden!

- .. ta en utdannelse …
- .. studere informatikk …
- .. lære programmering …
- .. løse oppgaver selv …

- Hvor mye av livet / jobben / samfunnsutviklingen ønsker du å delegere til andres algoritmer?
- Hvem skal forstå/ vurdere/ kontrollere/ utvikle teknologien?
- Programmering er et redskap for å behandle og kommunisere komplekse problemer på en strukturert måte. Kan *det* bli nyttig i fremtiden/ i jobb?
- Utdanning er en grunnmur, ikke et ferdig bygg – behovet for læring og utvikling forsvinner ikke
- Hvordan lærte du å snakke, gå, sykle, svømme?

https://www.uio.no/studier/emner/matnat/ifi/IN1000/

Informatikk-studiet og IN1000

- Mål for emnet
  - Kunne skrive fungerende objektorienterte programmer på noen hundre linjer
  - Solid grunnlag for videre studier
  - Vekt på grunnleggende, generelle konsepter og begreper
  - Trening i programmering ved hjelp av disse i Python
  - Ikke et hurtigkurs for utviklere – eller i Python

- Ingen krav til forkunnskaper, men helt nødvendig med jevnt arbeid!
  - Gjenspeiles i innleveringsoppgavene - og eksamensoppgaver mot slutten

https://www.uio.no/studier/emner/matnat/ifi/IN1000/h26/

Semesteret i IN1000

- Uke 1-6: Grunnleggende elementer i programmering
- Uke 7-12: Objektorientert programmering, større programmer, mer kompleksitet
- Ca Uke 13-15: Prøveeksamen, eksamenstips, repetisjon
- Uke 16: Eksamen (4. desember)

Digital, 4 timer,
kun skriftlige/ trykte hjelpemidler

Emneside og semesterside

← Studier ← Emner ← Matematikk og naturvitenskap ← Informatikk

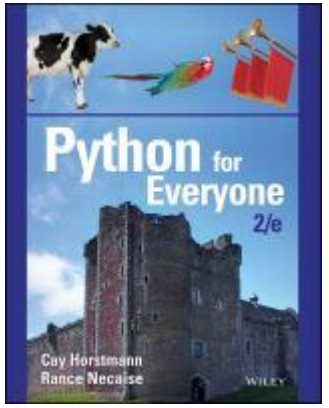
IN1000 – Introduksjon til objektorientert programmering

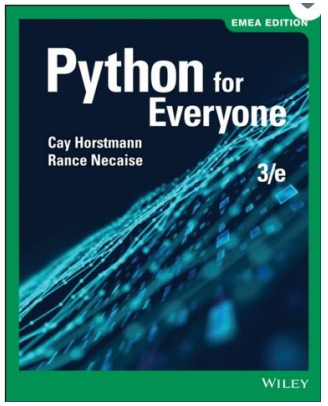
Studiepoeng: 10 **Nivå:** Bachelor

Undervisning: Høst **Eksamen:** Vår og høst **Undervisningsspråk:** Norsk

Beskrivelse av emnet

- → Kort om emnet
- → Hva lærer du?
- → Opptak til emnet
- → Overlappende emner
- → Undervisning
- → Eksamen







Timeplan

IN1000 - Introduksjon til objektorientert programmering

Timeplan

Fellesundervisning

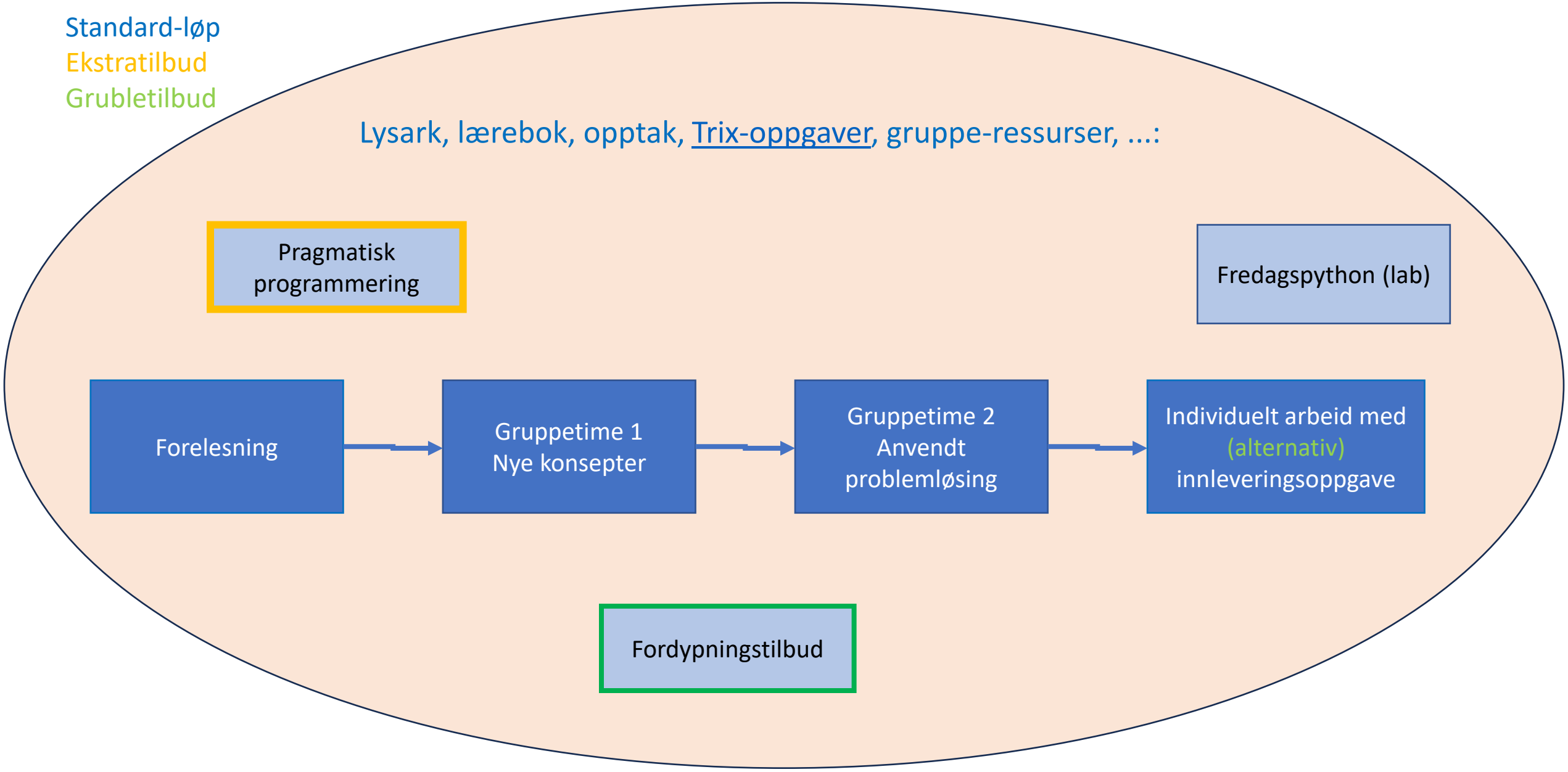
Vis alle aktiviteter (utskriftsversjon)

Forelesninger - tir. 14:15-16:00

Abonner på disse aktivitetene

Dato	Tid	Aktivitet	Sted	Foreleser	Ressurser/pensum
ti. 19. aug.	14:15–16:00	Kom i gang med informatikk, programmering og Python. Variabler og beslutninger	SL, Auditoriet	▪ S. A. M. Jensen ▪ G. K. F. Sandve	Alle ressurser for uke 1

!


Standard-løp
Ekstratilbud
Grubletilbud
Lysark, lærebok, opptak, Trix-oppgaver, gruppe-ressurser, ...:
Pragmatisk programmering
Fredagspython (lab)
Forelesning
Gruppetime 1 Nye konsepter
Gruppetime 2 Anvendt problemløsing
Individuelt arbeid med (alternativ) innleveringsoppgave
Fordypningstilbud

Hvordan øke sannsynligheten for et bra resultat?

- Eksamensresultater fra 2024 (da 6 av 8 innleveringer var frivillige) viser:
  - Av de som leverte alle innleveringer fikk 80% A-C, mens 5% ikke besto
  - Av de som leverte kun to innleveringer fikk 35% A-C, mens 45% ikke besto

=> Man *kan* gjøre det bra med minimum innleveringer

- Først og fremst hvis man har forkunnskaper *og* setter seg inn i pensum
- Forkunnskaper øker sannsynligheten for å gjøre det bra – men er ikke noen garanti – og kan gi feil inntrykk i starten av semesteret

=> Innleveringer øker sjansen for å gjøre det bra dramatisk

- Men er ingen garanti om man ikke jobber *selvstendig* med fokus på *læring*

Obligatoriske innleveringer i 2026

- Alle åtte må leveres innen fristen og godkjennes for eksamensrett

- Består av en kort quiz og programmerings-/ teorioppgaver

- Noen av programmeringsoppgavene er frivillige – men like viktige for å **følge progresjonen**!
  - Gir deg et inntrykk av hvordan du ligger an
  - Nyttige tilbakemeldinger fra retter
  - ..**hvis** du unngår andres løsninger (inkl KI) og for mye hjelp

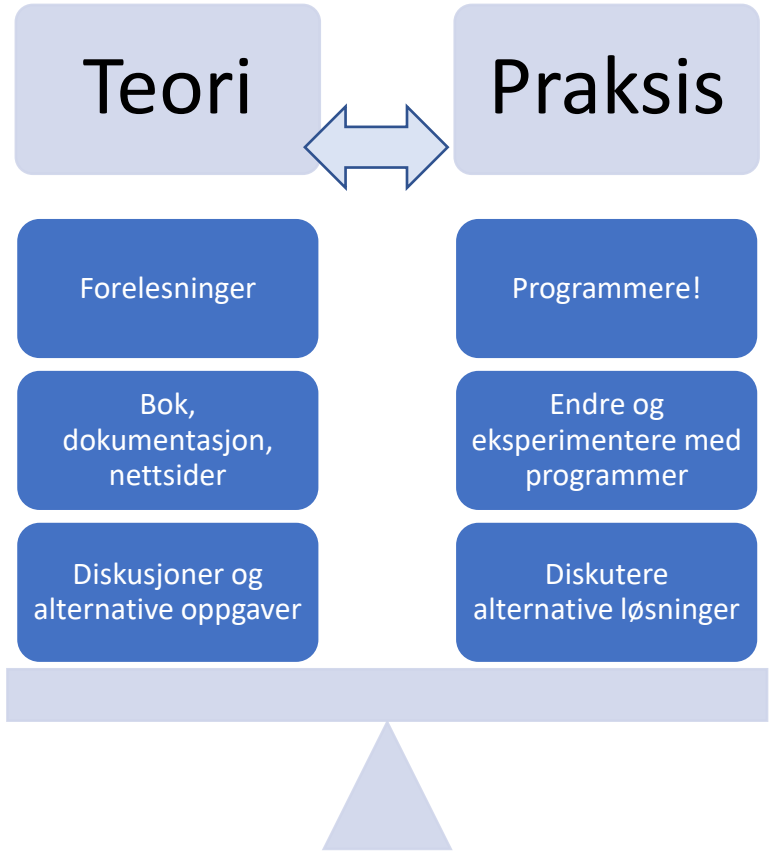
- Alt leveres i Devilry.ifi.uio.no
  - Harde frister, stenges ved fristen (tirsdager kl 23:59)
  - Lever gjerne flere ganger, siste gjelder (NB: Bruk filnavn fra oppgaveteksten)

Alt du trenger å vite finner du under **Obligatoriske innleveringer** på semestersiden

Programmering er et kreativt håndverk

- Teori i IN1000 handler om å lære
  - nye konstruksjoner og nyttige fremgangsmåter
  - prinsipper som øker kvaliteten på programmene du skriver
  - et begrepsapparat som gjør det raskere å lære nye språk og anvendelser

- En god programmerer har skrevet og finpusset *mange* programmer
- For å kunne kommunisere med andre og raskt gjenbruke kunnskap i nye situasjoner *må du bruke tid på teori*



Tips for programmeringsstudier

- Programmering handler om å definere og løse problemer/ finne fremgangsmåter, dvs
  - Du sitter alltid med et «problem» - og får belønning hver gang det er løst!
  - Ikke gi opp – selv om det av og til tar (urimelig mye) tid!



- Start koding selv om du ikke forstår alt, eksperimenter Praksis
- Men finn alltid ut hvorfor en feil "forsvant"! Teori



- Jobb sammen med andre, diskuter løsninger og "hva skjer her" Gruppetimer, Trix
- Jobb alene, svett over problemer ☺ Innlevering!

- Du kan ikke hoppe over pensum i programmering. Husk jevnt arbeid!

Tre råd og en påminnelse til slutt

- Unngå stress: Sett av tid
  - Og vær forberedt på at noen oppgaver og uker krever mer (eller mindre)

- Husk at du lærer!
  - Det er ikke forventet at du kan det du jobber med (ennå).
  - “Hva lærte jeg?”, ikke “Hvor rask var jeg?”

- Programmering krever modning – og det skjer i ulikt tempo
  - men ikke av seg selv!

⇒ Læring/ trening/ utvikling kan ikke delegeres
(gå, snakke, lese, sykle, svømme, styrke- eller kondisjonstrening, …)