# Time 2

- Pensum i objektorientert programmering
- Vurdering og digital hjemme-eksamen.
- IN1000 tilbud frem til eksamen
  - prøveeksamen
  - frivillig oppgave (Datasenter, lenke fra uke12-siden). Video-gjennomgang legges ut
  - gruppetimer (til og med 17.11)
  - repetisjonskurs (fra 18.11)
- Andre ressurser
  - eksamensoppgaver med løsningsforslag (NB: prøveeksamen ~fjorårets eksamen)
  - ukesider med lysark, opptak, Trix-oppgaver, gruppelærer-videoer mm
  - obliger, evt med tilbakemeldinger

# Hva skal vurderes? Fra kurssidene

## Etter å ha tatt IN1000

- forstår du prinsippene for objektorientert programmering og kan benytte disse til å skrive enklere objektorienterte programmer
- kan du programmere i programmeringsspråket Python og kan bruke dette til å løse mindre problemer ved hjelp av valg, løkker, funksjoner, lister, klasser og objekter
- kan du skrive oversiktlige og lesbare programmer
- er du i stand til å sette deg inn i andres programmer, finne eventuelle feil i dem og modifisere dem

## Overordnet pensum

- Kapittel 1-9 i *Python for Everyone 2/e* av Cay Horstmann og Rance Necaise (2. utgave, Wiley 2016)
- Innleveringer og det som er forelest (lysark fra forelesningene)

NB: **Introduksjon i objektorientert programmering (Python er verktøyet)**

Forelesninger, prøveeksamen og obliger viser:

- hvilket stoff vi prioriterer fra pensum
- hvordan vi forventer at du benytter det.

På eksamen kan du få:

- Variasjoner av programmer du har sett før
- Oppgaver der du må kombinere stoff på måter du ikke har sett før

# Kapittel 9 Objekter og klasser

## 9.1 Objektorientert programmering

Forelesning++: Uke 7

Samle data og metoder som behandler dem i samarbeidende *objekter*. Et objekt tilbyr et bestemt sett av tjenester i form av *metoder*.

Hvilke tjenester et objekt tilbyr defineres av *klassen* til objektet.

# Kapittel 9 Objekter og klasser

9.2 Implementasjon

= Hvordan skriver vi klassen!

Forelesning++: Uke 7

- Velge *instansvariabler* som representerer informasjonen hvert objekt skal ta vare på og jobbe med. Instansvariablene initialiseres i *konstruktøren.*

- programmere innholdet i metodene, inkl eventuelle hjelpemetoder

# Kapittel 9 Objekter og klasser

## 9.3 Grensesnittet til en klasse

Forelesning++: Uke 7

*Innkapsling*: Objektene kan brukes når man kjenner deres *offentlige grensesnitt*, altså de *metodene* klassen tilbyr:

- Hva gjør de, hvilke parametere trenger de, og hva returnerer de (om noe).
- Man trenger (bør) ikke kjenne til klassens *implementasjon* for å bruke objekter av klassen

# Kapittel 9 Objekter og klasser

## 9.4 Design av datarepresentasjonen

Forelesning++: Uke 7-12

Hva avgjør hvilke instansvariabler (datastruktur) vi trenger?

- Hvilke data bør være tilgjengelig (kunne hentes ut) i grensesnittet?
- Hvilke oppgaver skal objektene løse for oss som krever tilgang til data av noe slag – over tid, dvs gjennom flere metodekall?
- Trenger vi tjenester/ data fra andre objekter, som vi må ha referanser til?

# Kapittel 9 Objekter og klasser

## 9.5 Konstruktøren

Forelesning++: Uke 7+8

- Metoden __init__ kaller vi klassens *konstruktør*
- Konstruktøren kalles automatisk når vi oppretter et nytt objekt av klassen ved hjelp av klassenavnet – kalles aldri som en vanlig metode
- I konstruktøren definerer og initaliserer vi instansvariablene (datastrukturen) som hvert objekt får sin egen utgave av
- De må få en initialverdi – selv om vi noen ganger ikke vet før senere hvilken verdi de skal ha
- Konstruktøren har alltid en formell parameter *self*, og alle instansvariabler refereres til med self. etterfulgt av instansvariabelnavnet

# Kapittel 9 Objekter og klasser

## 9.5 Konstruktøren (forts.)

Forelesning++: Uke 7+8

- *Initialverdien* til en instansvariabel kan bestemmes på en av flere måter:
  - Når vi skriver klassen:
    Samme verdi for alle nye objekter (f.eks. teller = 0, eller en tom liste)
  - Når vi oppretter nye objekter av klassen:
    Parameter til konstruktøren, bestemmes for hvert objekt (f.eks. navn på student)
  - Konstruktøren finner selv verdien (for eksempel ved å lese fra fil)
- Senere kan verdien til en instansvariabel endres av andre metoder

# Kapittel 9 Objekter og klasser

## 9.6 Implementering av metoder

Forelesning++: Uke 8 +

- Alle metoder må ha med self-parameteren (hvilket objekt skal jeg arbeide med denne gangen?)
- Alle instansvariabler aksesseres ved hjelp av parameteren self i metodene
- Metoder kan være offentlige (tilhøre grensesnittet) eller non-public ("*hjelpemetoder*", brukes bare av andre metoder i klassen)

# Kapittel 9 Objekter og klasser

## 9.10 Objektreferanser (referanser)

Forelesning++: Uke 8

- Brukes for å "få tak i" et bestemt objekt

- Må i endel sammenhenger passe på forskjellen på referanse og objekt:
  - to referansevariabler kan referere til samme objekt (testes med **r1 is r2**)
  - innholdet i en referansevariabel endres som andre variabler (kan settes til å referere et annet objekt, eks **r1 = r2**)
  - *innholdet* (instansvariablene) i objekter endres med kall på metoder (eks **r1.endre(5, 7)** )

- self, none

# Kapittel 9 Objekter og klasser

## 9.11 Eksempel: En klasse for brøker

- Eksempelet er nyttig, men ikke pensum
- Pensum: Seksjon 9.11.3 som handler om spesielle metoder
- Vi har brukt
  - `__init__`
  - `__str__`
  - `__eq__`

*Forelesning++: Uke 9*

# Kapittel 9: Objects and classes
# "Faktastoff"

Pensum: Kapittel 9.1 – 9.10, 9.11.3. 40 sider *totalt*

9.1-9.6
9.10
9.11.3:

- "alt" du trenger å vite om klasser og objekter i Python

Oppsummeringer, faktabokser:

- Programming Tip 9.1 og 9.2
- Syntax 9.1 og 9.2
- Common error 9.1

Hopp over Special Topics (9.1,9.2 og 9.3)!!

# Kapittel 9: Objects and classes
# Tips om fremgangsmåter, eksempler

9.7-9.9

- nyttig om testing og hvordan komme fra ide til ferdig objektorientert program
- How to 9.1, Worked example 9.2: Oppskrift og eksempler; Klassen Meny og klassen Bankkonto

*Vi forventer ikke at du designer et større system med flere klasser. Men du bør kunne forstå en slik struktur og kunne implementere den*

9.11

- stort eksempel med en del stoff utenfor pensum,
  - NB: 9.11.3 (Special Methods) er pensum, og du bør kunne skrive og bruke metodene __str__ og __eq__

# Utenom læreboken:
# Flere klasser, mange objekter

- Typisk stoff for "stor oppgave" på eksamen

Forelesning: Uke 9 og 10

- Nyttig å jobbe med flere eksempler, ulike strukturer
  - Oblig 7 og 8, frivillig Datasenter-oppgave, prøveeksamen
  - T-bane og DNA-eksempler fra forelesning

- Merk forskjellen på å se en ferdig løsning, eller noen som livekoder – og det å jobbe selv med oppgaven.

=> ikke gi opp for raskt under trening, aksepter at du må gjennom flere runder med endringer og forbedringer!

Forelesning++: Uke 12

# UML klassediagrammer

- Klassediagrammer er statiske (viser klassedefinisjoner, ikke hvilke verdier som finnes i hukommelsen under en bestemt kjøring)

- Kan ha ulik detaljeringsgrad
  - Bare klassenavn og relasjoner
  - evt også instansvariabler med type
  - evt også metoder med typer på parametere og returverdi
  - angivelse av hva som er public (inngår i grensesnittet) eller non-public (brukes kun internt i klassen)

- Relasjoner kan merkes med antall og navn

# Utfordringer i oppgave med flere klasser

- Sjekk mot oppgavetekst / koden din hva metoder du bruker fra andre klassen skal ha som parametere og hva de leverer ut som returverdi

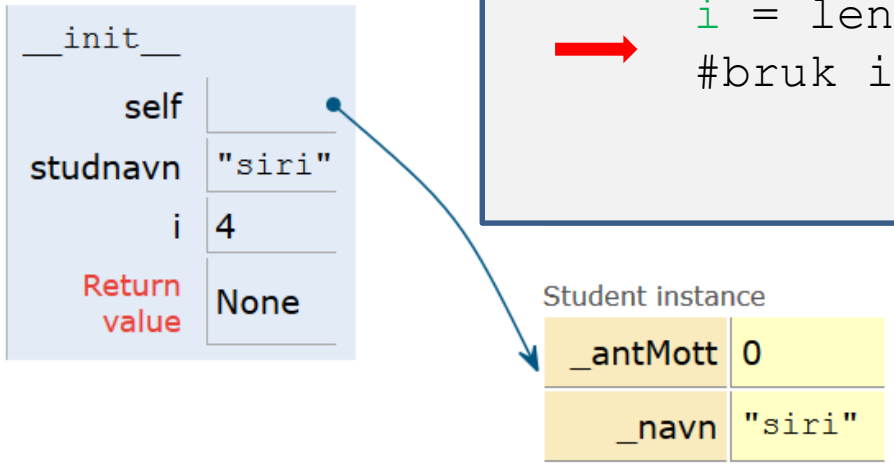
- Bruk gjerne variabel- og parameternavn som indikerer om
  - innholdet er en liste/ ordbok (flertall) eller en enkelt verdi (student eller studenter)
  - streng/int/... eller referanse (eks obligId eller oblig)

- Mer i gjennomgang av prøveeksamen neste uke, se ellers neste slide og generelt om pensum i oop.

# Instansvariabler, lokale variabler, parametere

```
class Student :

  def __init__(self, studnavn):
    self._antMott = 0
    self._navn = studnavn
    i = len(studnavn)
    #bruk i til noe...
```



# IN1000-eksamen 2020 – all info

- Digital hjemme-eksamen i eksamenssystemet Inspera, se semestersiden:
  - Tid og sted:
  - https://www.uio.no/studier/emner/matnat/ifi/IN1000/h20/eksamen/index.html
  - Flere IN1000-spesifikke detaljer (oppdateres):
  - https://www.uio.no/studier/emner/matnat/ifi/IN1000/h20/informasjon-om-eksamen-2020/index.html

- Eksamensavvikling ved MN-fakultetet høsten 2020:
  - https://www.mn.uio.no/om/hms/koronavirus/eksamen-2020.html

- "Alt" om Inspera:
  - https://www.uio.no/studier/eksamen/inspera/

# Eksamen skal vurdere det samme som tidligere skoleeksamener

- Oppgaven følger samme mønster som tidligere år (ikke vår 2020)
  - men oppgaver med spørsmål om resultat av kjøring erstattes
  - det legges til rette for individuell/ selvstendig besvarelse
  - det kan bli noe større/ annerledes arbeidsbelastning

- Sensur vil gjennomføres som før
  - det kreves ikke kjørbar kode, opplagte skrivefeil gir ikke trekk
  - eksempler er til illustrasjon, ikke komplette tester

⇒ du må vurdere nytten av digitale hjelpemidler ift tidsbruk
  ⇒ editor?
  ⇒ pythontutor.com?
  ⇒ kjøre kode i interpreter?

# Oppbygging av eksamenssettet

- Oppgave 1 og 2 tester kjennskap til hvordan kode utføres
- Oppgave 3 tester ferdigheter i å bruke ulike Python mekanismer i mindre programmer
- Oppgave 4 tester ferdigheter i å forstå og implementere et større objektorientert program med flere klasser
- Oppgave 5 tester evne til løse en mer algoritmisk krevende oppgave av begrenset omfang

⇒ Det vil oftest lønne seg å prøve oppgavene i denne rekkefølgen, fra enklere til mer avanserte

⇒ Ikke bruk for mye tid på én oppgave om du står fast, gå heller tilbake hvis tid

# Mange opplever knapt med tid

- Eksamens-settet har totalt 100 poeng (prøveeksamen kan inneholde noen ekstra oppgaver)
- Vurder tidsbruk på enkeltoppgaver opp mot antall poeng de gir
- Svar på det det spørres om – kort og presist. Ikke gjenta oppgaven.
- Vi krever ikke kommentarer på eksamen. Kan brukes om du ønsker å klargjøre noe i løsningsvalget (men husk at sensor kjenner oppgaven godt)

# Inspera

- Tilgjengelig i nettleser, påvirker ikke andre programmer
- Se dokumentasjon/ info: https://www.uio.no/studier/eksamen/inspera/
- Les førstesiden ("Informasjon")
- Velg språk (øverst på siden) (ikke på prøveeksamen)
- Du kan gå frem og tilbake i besvarelsen og endre svar så mange ganger du vil inntil levering
- Dersom det spørres om et heltall, skriv et heltall (ikke for eksempel 6.0). Generelt: Ikke legg på ekstra tegn i oppgave 1 og 2

# Prøveeksamen - praktisk

- Tilgjengelig i Inspera ila dagen (beskjed kommer på semestersiden)
- Mest mulig likt eksamen – men ikke de samme oppgavene ☺

Plan:

- Stenges tirsdag kveld (som oblig-fristene), da skal resultat av automatrettede oppgaver være tilgjengelige
- Gjennomgås på forelesning neste uke (opptak gjøres)
- Åpnes igjen etter forelesning (for de som vil jobbe mer i Inspera) til ca 1.12

⇒ Bruk anledningen til å teste ut både det faglige OG arbeidsform

⇒ Gjerne sette av 4 timer og kjenne på tidsbruk