Andre time

- IN1000 tilbud frem til eksamen
- Vurdering og digital eksamen
- Pensum i objektorientert programmering
- Typiske utfordringer i stor oppgave, eksempler fra prøveeksamen
- Tips til eksamensgjennomføring og forberedelser

IN1000-seminar

```
1  class IN1000_seminar:
2      def __init__(self, pizza, gruppelærere):
3
4          self._dato = "Fredag 15. november"
5          self._program = (" "
6              "14:15 - Foredrag i Simula"
7              "16:15 - Oppgavejobbing på termstuer"
8              "18:00 - Gratis pizza til alle påmeldte"
9              "20:00 - Arrangementet slutter")
```

HUSK PÅMELDING! Innen 13. november på Facebook-arrangementet

også lenke fra uke 13-siden!

IN1000 frem mot eksamen 6.12

- Prosjektoppgave, programmering i grupper settes i gang på FUI-seminar
  - fredag 15.11 fra 14:15
  - påmeldingsfrist for pizza i morgen
- Ordinære gruppetimer denne uken (13)
- Neste to uker (14-15): Repetisjonskurs og åpne lab'er (tider kommer på semestersiden!)
- 29. november 12:15 -14:00: Avslutning og peer-review av prosjektoppgave, juleservering

Om digital eksamen og Inspera

- Digital eksamen i Silurveien, kun tilgang til Inspera
- Prøveeksamen: Beste måten å teste ut og bli vant til å jobbe i Inspera på, med typiske IN1000-oppgaver

- Når og hvor for deg: Studentweb
- "Alt" om IN1000 eksamen:
- https://www.uio.no/studier/emner/matnat/ifi/IN1000/#exam
- "Alt" om eksamen ved UiO: https://www.uio.no/studier/eksamen/
- "Alt" om Inspera:
- https://www.uio.no/studier/eksamen/inspera/

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

Objektorientert programmering

- Deler opp koden i håndterbare elementer
  - Oversikt, vedlikehold, gjenbruk
- Klassen definerer et mønster for objekter
  - instansvariablene lagrer data som hører sammen
  - metodene implementerer operasjoner på dataene
- *Innkapsling* hjelper oss å utnytte fordelene ved OO
  - Kun metodene i grensesnittet brukes utenfra klassen
  - Instansvariable og interne metoder merkes som "non-public" ved navn som starter med _
  - Disse brukes kun av metoder i samme klasse
  - Gjennomføres i IN1000

Kapittel 9 Objekter og klasser

9.1 Objektorientert programmering

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 kommer an på *klassen* til objektet.

Kapittel 9 Objekter og klasser

9.2 Implementasjon
= Hvordan skriver vi klassen!

- Velge *instansvariable* 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

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

Hva avgjør hvilke instansvariable (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 fra andre objekter, som vi må ha referanser til?

Kapittel 9 Objekter og klasser

9.5 Konstruktøren

- Metoden __init__ kalles 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
- De defineres med self. etterfulgt av instansvariablenavn

Kapittel 9 Objekter og klasser

9.5 Konstruktøren (forts.)

- Initialverdien (startverdien) til instansvariable kan bestemmes på flere måter:
  - Når vi skriver klassen:
    Samme verdi for alle nye objekter (f.eks. teller = 0, 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 verdiene endres av andre metoder

Kapittel 9 Objekter og klasser

9.6 Implementering av metoder

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

Kapittel 9 Objekter og klasser

9.10 Objektreferanser

- Brukes for å holde rede på objekter
- Må vite når vi bruker/ sammenligner referansen og når vi bruker/ sammenligner objektet
- 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__`

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

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

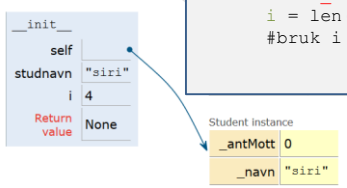
- Typisk stoff for "stor oppgave" på eksamen
- Nyttig å jobbe med flere eksempler, ulike strukturer
  - Oblig 7 og 8, prosjektoppgaven
  - T-bane og DNA-eksempler fra forelesning
- Tenk "ett ledd av gangen"
  - å bruke klasser som dere ikke har skrevet (ennå)
  - å skrive klasser som dere ikke vet hvor skal brukes (ennå)
- Tegn (skisse av) objekter og referanser for et tenkt eksempel for oversikt

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 (rett eller retter)
  - streng eller referanse (eks katNavn eller kat)

Instansvariable, lokale variable, parametere

```
class Student :

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



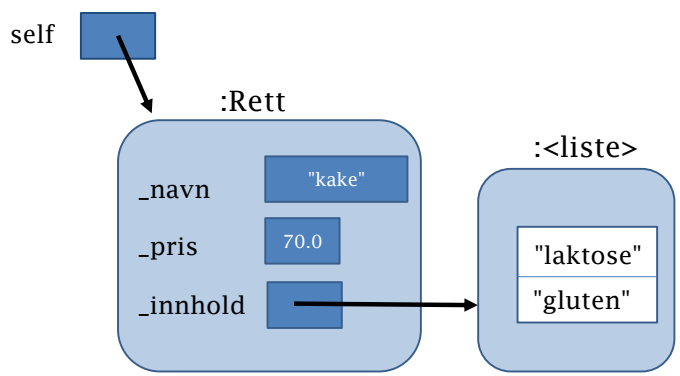
Oppgave 4a)

```
# 4a) class Rett 10 p
class Rett :
    def __init__(self, navn, pris, innhold) :
        self._navn = navn
        self._pris = pris
        self._innhold = innhold

    def sjekkInnholdOk (self, unngaa) :
        for innh in self._innhold :
            if innh in unngaa :
                return False
        return True

    def __str__ (self):
        s = ("Rett: " + self._navn + ". Pris: "
             + str(self._pris) + ". Innhold:")
        for i in self._innhold :
            s +=  " " + i
        return s
```

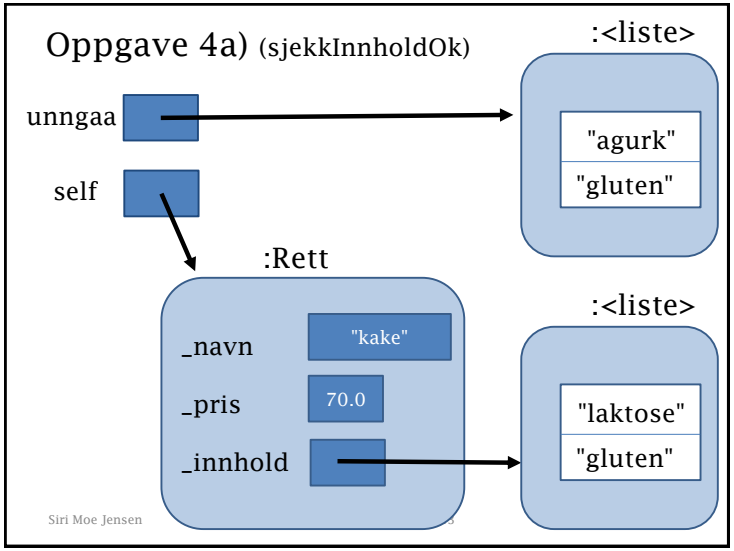

Oppgave; hva er instansvariabler, parametere og lokale variabler her?

```
# 4a) class Rett 10 p
class Rett :
    def __init__(self, navn, pris, innhold) :
        self._navn = navn
        self._pris = pris
        self._innhold = innhold

    def sjekkInnholdOk (self, unngaa) :
        for innh in self._innhold :
            if innh in unngaa :
                return False
        return True

    def __str__ (self):
        s = ("Rett: " + self._navn + ". Pris: "
             + str(self._pris) + ". Innhold:")
        for i in self._innhold :
            s +=  " " + i
        return s
```

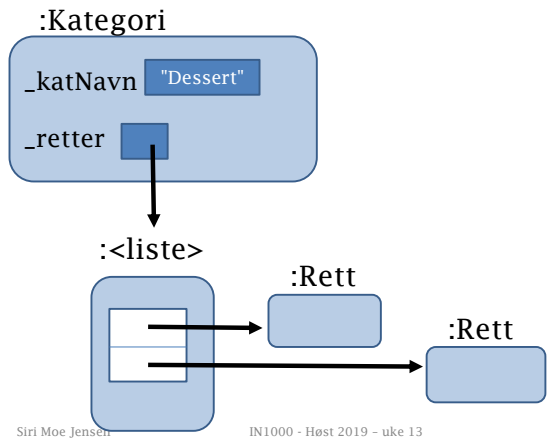

Oppgave 4a) (konstruktør)



Oppgave 4a) (sjekkInnholdOk)



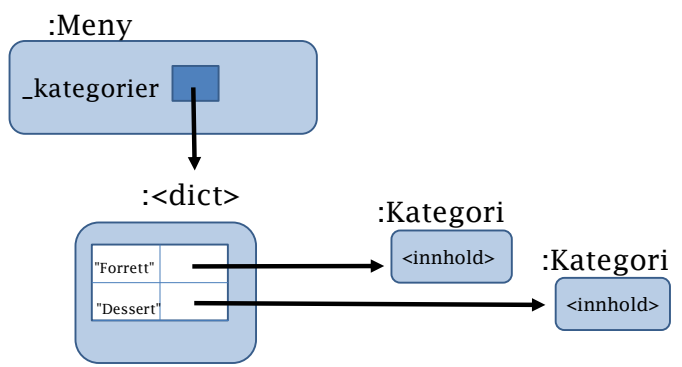
Oppgave 4b)



Oppgave 4b)

```
25 #4b) class Kategori 5 p
26 class Kategori :
27     def __init__(self, katNavn, retter) :
28         self._kNavn = katNavn
29         self._retter = retter
30 
31     def hentOkRetter (self, unngaa) :
32         okRetter = []
33         for r in self._retter :
34             if r.sjekkInnholdOk(unngaa) :
35                 okRetter.append(r)
36         return okRetter
37 
38     def skrivKategori(self) :
39         print(self._kNavn + ":")
40         for rett in self._retter :
41             print(rett)
```


Oppgave 4c)



Utfordringer i oppgave med flere klasser

- pass på nøkkel versus verdi i ordbøkene!

verdier

nøkler

```
ord = {1:"bok", 2:"film"}
for o in ord.values():
    print(o)

for nr in ord:
    print(nr)
```


Oppgave 4c)

```
43 #4c) class Meny 10 p
44 class Meny:
45     def __init__(self, katNavnListe):
46         self._kategorier = {}
47         for knavn in katNavnListe :
48             nyKat = self._lesKategoriFil (knavn+".txt")
49             self._kategorier[knavn] = nyKat
50 
51     def hentRedusertMeny(self, unngaa):
52         okKategorier = {}
53         for knavn in self._kategorier :
54             retter = self._kategorier[knavn].hentOkRetter(unngaa)
55             if retter  :
56                 nyKat = Kategori(knavn, retter)
57                 okKategorier[knavn] = nyKat
58         return okKategorier
```


Utfordringer i oppgave med flere klasser

- Ordreløkke
  - pass på at du spør om ordre minst en gang
  - alltid endre inne i while-løkken det som testes

Oppgave 4f)

```
#4f)     Hovedprogram: 5 poeng
def hovedprogram():
    mineKategorier = ["Forretter", "Hovedretter", "Desserter"]
    minTakeAway = TakeAway(mineKategorier, "kunder.txt")
    kundeId = input("Oppgi telefonnr: ")
    while kundeId != "" :
        minTakeAway.betjenKunde(kundeId)
        kundeId = input("Oppgi telefonnr: ")

hovedprogram()
```

Før eksamen I: Fag

- Fra problem til program + enkeltdeler av pensum
  - Løs oppgaver fra Trix og evt lærebok
  - Forelesninger, slå opp/ utdyp fra lærebok
  - Lag gjerne egne oppgaver / utvidelser/ variasjoner!
  - Uløste obligoppg? Nyttige tilbakemeldinger på leverte?
  - Praktisk trening OG teoretisk forståelse/ kunnskaper
- Trening i større programmer med komplekse strukturer, overblikk/ helhet:
  - Prosjektoppgave
  - Prøveeksamen 2018, 2017, evt tidligere eksamener
  - Oblig 7 og 8 (kan de utvides eller forbedres?)

Før eksamen II: Praktisk

- Gjennomfør prøveksamen!
- Planlegg hjelpemidler (alt skriftlig, ingenting elektronisk)
- Les eksamensreglementet, sjekk grundig hvor/ når du skal møte (kurssidene og studentweb), beregn god tid
- Obliger må være godkjent for å bli meldt opp til eksamen
- Følg med på beskjeder på semestersidene, og les eller videresend UiO-mail

På eksamen

Hensikten er å vise sensorene hvilke deler av pensum du behersker

- Les førstesiden ("Informasjon")
- Velg språk (øverst på siden)
- Du kan gå frem og tilbake i besvarelsen og endre svar så mange ganger du vil inntil levering
- Foreleser(e) vil gå rundt etter ca 0.5-1.5 time, forbered spørsmål om uklarheter
- Programkoden som skrives vurderes av sensorer = mennesker. Skal ikke kompileres eller kjøres av maskiner.
- Les oppgaven grundig! Hva ber den (ikke) om?

Inspera

- Unngå å bruke " eller ' i oppgaver der det f eks spørres om
  - Hva skrives ut her
  - Hvilken verdi har variabelen xxx
- Dersom det spørres om et heltall, skriv et heltall (ikke for eksempel 6.0)
- Hvis det spørres om deler av et progran i flere deloppgaver skriv hver del der den etterspørres

Mange opplever knapt med tid

- Hovedprinsipp: Skriv direkte inn i Inspera – kan endres. Men tegn/ skisser/ hold rede på data ved gjennomgang av kode ved siden av.
- Oppgavesettet gir totalt maksimalt 100 poeng. 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)

Sensor vil deg vel!

- Hensikten er å se hva du behersker av læringsmålene – ikke å pirke på språk eller skrivefeil
- Vi leter etter hva du kan – men du må vise oss det (og det må svare på det oppgaven spør om)
- Har du ikke tid til å programmere i detalj *kan* pseudokode/ kort beskrivelse være bedre enn ingenting – men den må vise noen relevante tanker om hvordan du ville gått frem. (dvs mer enn å gjenta oppgaven!)

Lykke til!

(og bruk gruppetimer og Piazza om det dukker opp spørsmål fremover)