En hvilerett er en liten rett som bryter opp måltidet, som frisker opp munnhulen for eksempel etter en rett med fløte. Den skal også skjerpe appetitten på neste rett.

Uke 6: "Hvilerett"

IN1000

Høst 2020

Siri Moe Jensen

Plan for forelesningen

Dagens gode nyhet: Ingen nye Python nøkkelord denne uken!

- Planer for undervisningen videre i IN1000 inkl litt læringsteori...
- Noen typiske feil & eksperimenter med Mentimeter og gruppediskusjoner
- Hva er et godt program?
- Gjennomgang av Mentimeter oppgavene – og svarene
- Tilbakemeldinger (i Mentimeter)

Semesteret i IN1000

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

Obligatoriske innleveringer

- Alt leveres i Devilry.ifi.uio.no (demo i gruppetime)
- Harde frister, stenges ved fristen
- Alle frister tirsdag kl 23:59
- Lever gjerne flere ganger, siste gjelder

- 1-6: ukes-obliger, disse gir poeng
  - I år ikke obligatoriske – men like nødvendige for å holde (og sjekke) progresjon!

- 7-8: større obliger, begge må godkjennes
  - Utsettelse ved egenmelding til gruppelærer/ retter
  - Nytt forsøk ved grensetilfeller

- NB: Krav til selvstendig arbeid! Se info og oblig-reglement lenket fra obligsiden

Tips for programmeringsstudier

1

- Programmering handler om å definere og løse problemer/ finne fremgangsmåter, dvs
  - Mye av læringen skjer mens du klør deg i hodet
  - Lær å like det ☺

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

2

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

- Du kan ikke hoppe over pensum i programmering

Learning that requires understanding

- Learning and teaching is NOT like plugging a pendrive into a learner's head.
- Learning is defined by BOTH the intended learning outcome of the teacher AND the learner's prior knowledge, experiences, and perhaps misconceptions.
- This is considerable work that the learner must do, though teachers may assist by various prompts.

on constructivism, adapted from UK-ICER20 keynote by Steve Draper, 3.9.20

Why and how is peer interaction so important in learning?

It does not rely on a peer telling you the right answer, nor on the soundness of their judgement.

It works by making the learner question themselves under the stimulus of a peer suggesting something different. A peer is often more effective at this than a teacher (or someone you judge ignorant): because with a peer, if you disagree, there is roughly a 50% chance of them being right or wrong, which forces you to think about issue rather than relying on the status of the person.

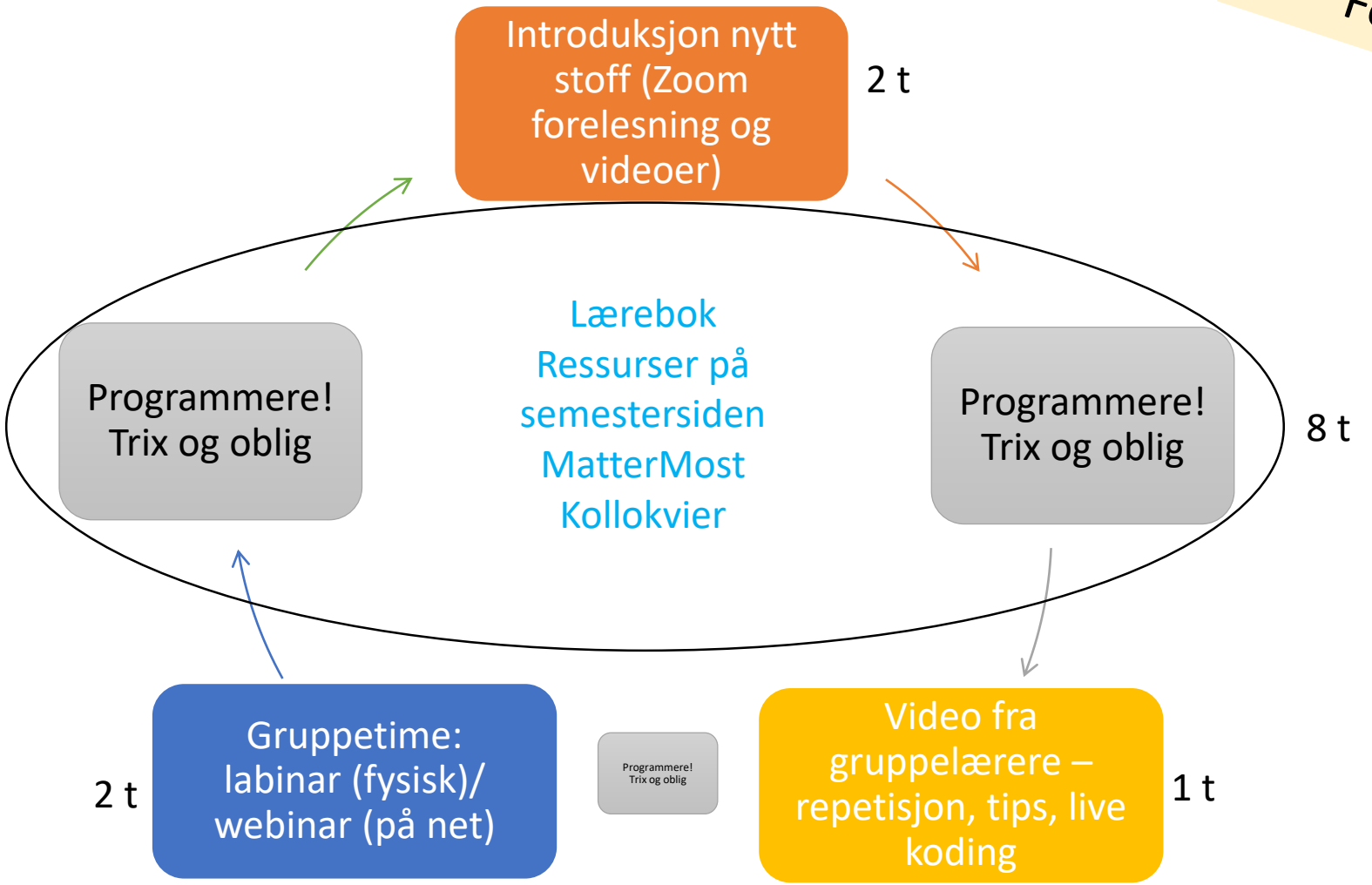
They act as a catalyst to your own thinking, not a source of pre-digested knowledge.

They prompt subsequent learning through a metacognitive mechanism.

UK-ICER20 keynote by Steve Draper, 3.9.20

Arbeidet i IN1000 – ca 13 t/ uke

Repetisjon:
- Innhenting
- Fordypning



Nå: Mer om pensum fra uke 1-5

- Noen vanskeligheter fra obliger og eksamen

- Vi ønsker at dere skal bruke tid på å tenke over
  - hva som faktisk står i koden
  - hva som faktisk skjer på detaljnivå
  - hvilke feil det er lett å gjøre

- Bruker Mentimeter for å gi oppgaver som besvares individuelt og anonymt
  - gå til menti.com i et eget browser-vindu på PC eller mobil

- Prøver ut dette dels i plenum, dels i grupper (breakout rooms)

"Code tracing" – å lese programmer

Vårt mål med undervisningen: Effektiv læring!

- Du lærer mens du jobber aktivt med stoffet (mindre av å bare høre på)
- Å lese kode og "se" hva som skjer er helt nødvendig for å kunne skrive kode
- Å måtte formulere seg om kode bidrar til fagkunnskap og forståelse
- Det er ikke flaut å ta feil her – målet er å oppdage misforståelser og bli bevisst på detaljer nå, og ikke etter å ha jobbet med en feil i timevis (eller et helt semester)
- Å kunne jobbe med andre også når man ikke kan alt er viktig i studier og jobb!

- Kanskje blir du også litt bedre kjent med noen medstudenter

Slik gjør vi det

- For hvert Mentimeter spørsmål:
  1. Svar raskt. Ingen ser hva akkurat du svarer ☺
  2. Tenk gjennom: Hva kan gjøre at noen gir andre svar?
  3. Svar en gang til (hver oppgave kommer to ganger)

- Vi går gjennom resultatene – og svarene – senere i forelesningen!

- Du vil bli spurt om tilbakemeldinger helt til slutt i Mentimeter

Pause til 10 på halv

Hva lærer vi når vi lærer programmering?

- Syntaks. Hvilke tegn skal med i hvilken rekkefølge i f eks en while-løkke?
- Semantikk. Hvordan utføres while-løkken? Hva skjer med programflyt og variabler?

- Logikk. Bruker vi den slik at den løser vårt problem?

Her virker programmet!!

- Pragmatikk, God design, Clean code, ...: Hvordan bruke språket på en best mulig måte?
- Var en while-løkke beste valg – og er den skrevet slik at programmet er
  - lett å lese og forstå?
  - lett å endre?
  - (raskt å kjøre med effektiv bruk av prosessor og minne?)

Style guide – PEP 8

- Det er skrevet en egen "style guide" for Python
- Gode tips om bl.a. navngiving og hvordan programmet bør "se ut" (layout):
  - navngiving
  - whitespace
  - kommentarer
  - linjelengde/ -deling

- Viktig: Dette er en guide
- Konsistens og bevisste valg er hovedpoenget – formålet er å skrive lettlest, lettforståelig kode

Navngiving

- Se craftmanship-modul uke 4

Variabler og funksjoner:

- I læreboka (og Java): Påfølgende ord har stor forbokstav (mixedCase)

- PEP 8: lowercase_with_underscores (snake_case)

Klassenavn: Stor første bokstav i hvert ord (CamelCase)

- Er dette gode variabelnavn?

  I (stor 'i') l (liten 'L') O (stor 'o')

Sterke meninger & pragmatisme - PEP 20

The Zen of Python

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

```python
while True:
    kommando = int(input("Kommando (-1 avslutter): "))
    # Mange linjer med kode
    # som gjør forskjellige ting,
    # kanskje tester på input
    # og utfører ulike handlinger
    # avhengig av verdien

    if kommando == -1:
        break

print("Programmet avsluttes")
```

- Tegn på "uklar ide" om løkkens oppgave, uryddig struktur
- Brukes svært ofte feil på eksamen

- Ikke anbefalt i IN1000!

- Når man ser "while" ser man når den skal utføres

- Du er tvunget til å bestemme hva som avgjør at du har oppnådd det du ville
- og kommunisere det på ett riktig sted

- I IN1000 bruker vi løkker "til noe er oppnådd"

```python
kommando = int(input("Kommando (-1 avslutter): "))
while kommando != -1:
    # Mange linjer med kode
    # som gjør forskjellige ting,
    # kanskje tester på input
    # og utfører ulike handlinger
    # avhengig av verdien
    kommando = int(input("Kommando (-1 avslutter): "))

print("Programmet avsluttes")
```

God design

- Viktigere tema jo mer dere lærer
  - flere valg for samme problem
  - større programmer
  - mer samarbeid med andre
  - mer gjenbruk av kode

- Objektorientert programmering og design tilbyr verktøy som muliggjør god design i komplekse programmer med en bestemt måte å tenke på

Neste uke

- Hvorfor objektorientert programmering?
- Å designe, skrive og teste egne klasser
- Å bruke egne klasser i programmer

=> syntaks og overfladisk semantikk

Men først – svar gjerne på resten av Mentimeter-spørsmålene!