in1000
uke 6
dypdykk

viktig forståelse fra forrige ukes pensum

læremål

0

range

0

range

en snarvei til å lage lister med tall

utforsk i python tutor:

```
r = range(6)
l = list(r)
print("range(6) -->", l)

r = range(1, 6)
l = list(r)
print("range(1, 6) -->", l)

r = range(1, 6, 2)
l = list(r)
print("range(1, 6, 2) -->", l)
```

in1000
uke 7
nytt lærestoff

oversikt og forståelse i kontekst

læremål

0

mer om objekter

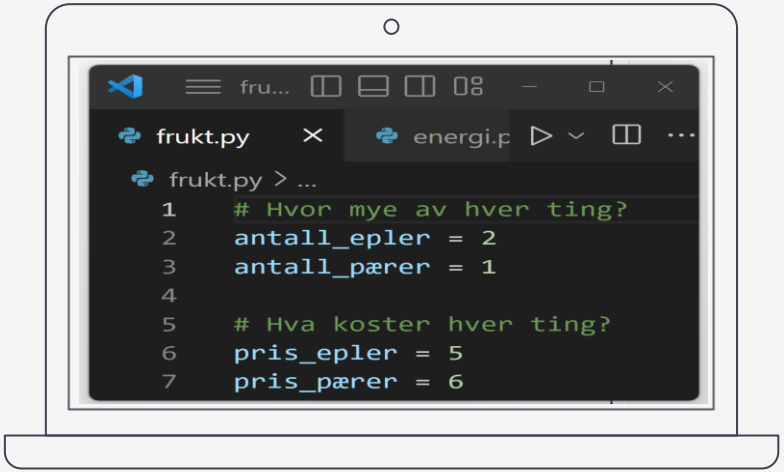
1

klasser

0

objekter

klumper med data som kan
gjøre ting selv



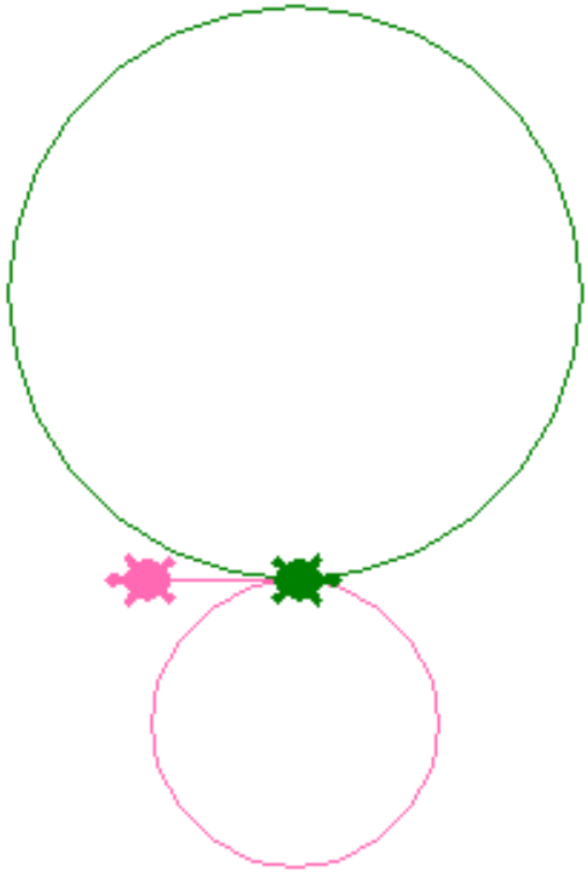
live-koding

motiverende eksempel om objekter

objekter.py
(legges ut i timeplanen på emnesiden etter forelesningen)

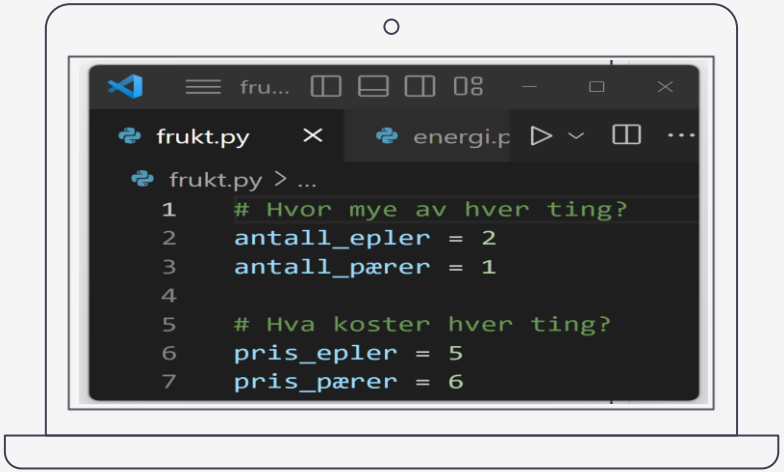
skilpadde-objektene hadde forskjellige **egenskaper** (farge, hastighet…)

men de hadde de samme **metodene** (circle, forward, right…)



1 klasser

oppskrifter på egne objekter
tilpasset problemene **vi** vil løse



live-koding

hvordan lage en klasse og objekter

klasse.py
(legges ut i timeplanen på emnesiden etter forelesningen)

husk å se på koden i [python tutor]

klasse vs objekt

klasse (Student)

oppskrift på hvordan vi lager en bestemt type objekt

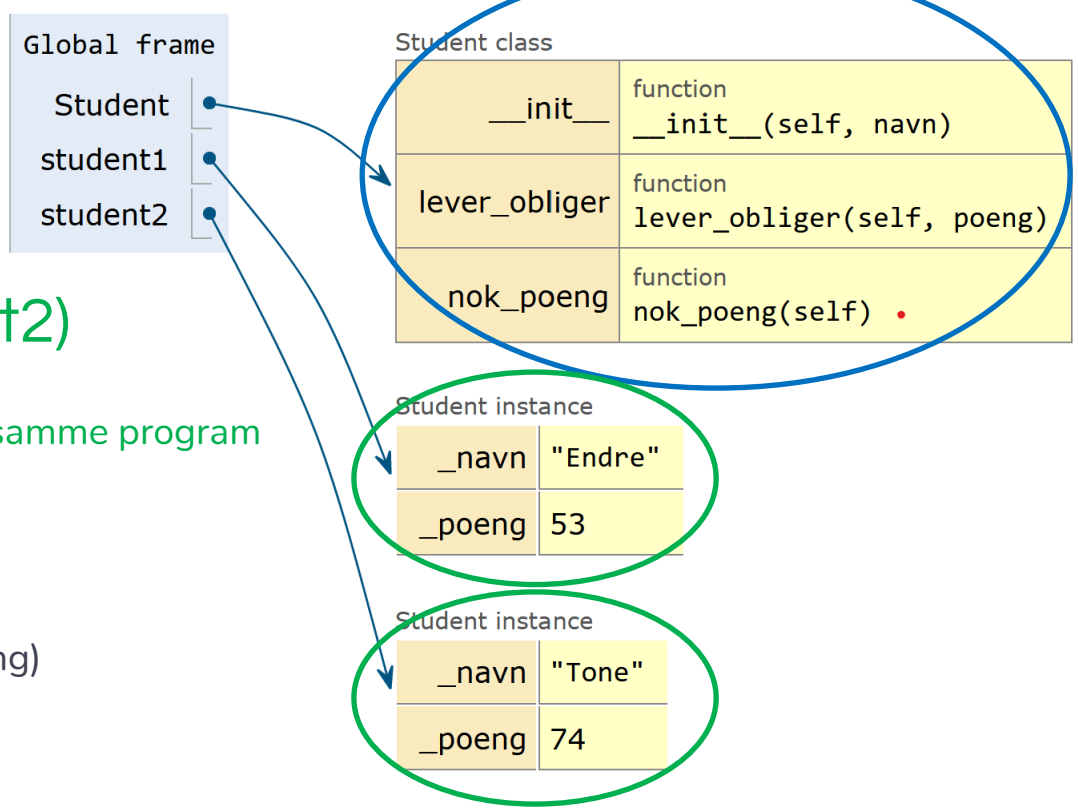
objekter (f.eks. student2)

resultatet av oppskriften
en klasse kan lage mange objekter i samme program

likt og ulikt

objektene har forskjellige verdier på **instansvariablene** sine (_navn, _poeng)

men **metodene** (lever_obliger, nok_poeng) er felles for klassen



en smart ting med objekter: klassens metoder har tilgang til objektenes **instansvariabler**

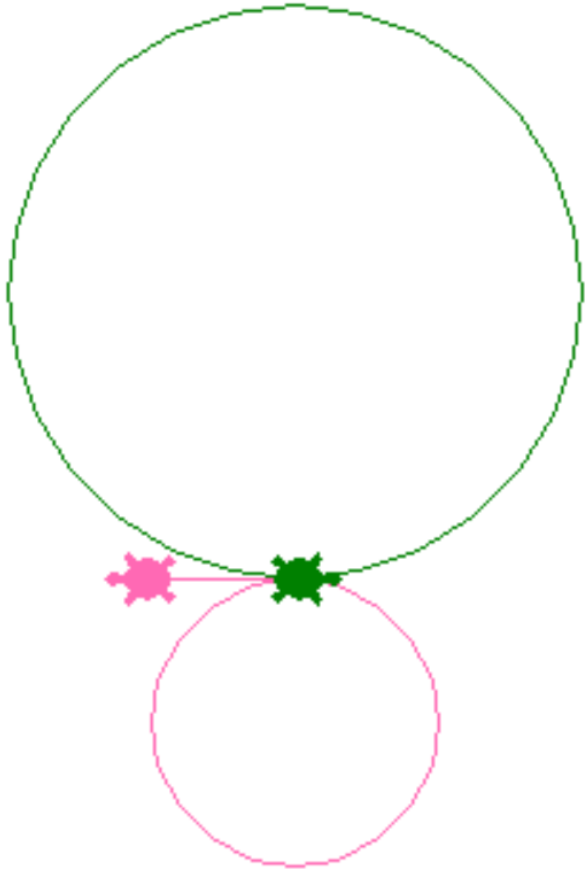
disse har vi tilgang til gjennom **self:**

```
def nok_poeng(self):
    if self._poeng >= 57:
        print(self._navn, "har nok poeng!")
    else:
        print(self._navn, "har IKKE nok poeng!")
```

instansvariablene kan ha forskjellig verdi for forskjellige objekter!

```
def color(self, color_name):
    self._color = color_name
```

```
shelly.color("green")
raphael.color("hotpink")
```



objekter tilbyr tjenester

et objekt er en klædd med data som selv “vet” hvordan det skal gjøre en del ting (ved hjelp av **metoder** definert i klassen)

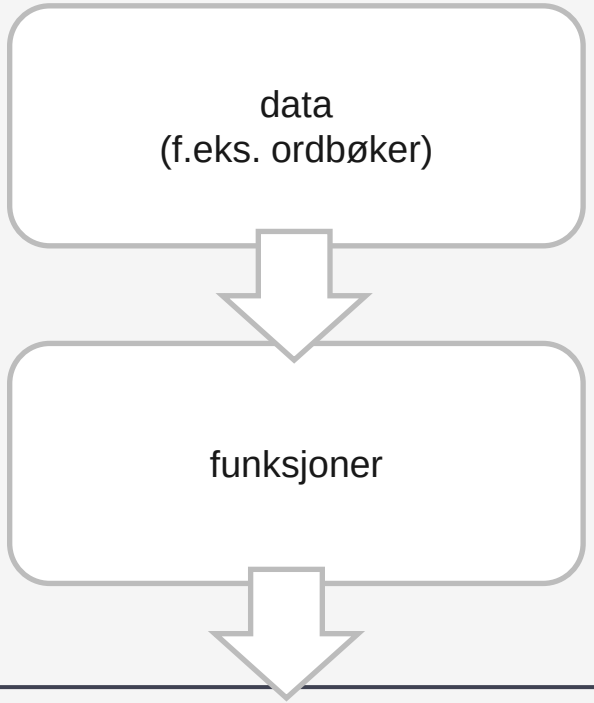
vi kan også representere data uten klasser – men da kan de ikke gjøre stort på egen hånd (de blir ganske **passive**)

```
student3 = {
    "navn": "Trelaina",
    "poeng": 5
}
```

med klasser kan objekter **aktivt** gjøre ting selv når vi har gitt dem en **metode** for det (som å gi beskjed om de har nok poeng når vi spør)

```
student1.nok_poeng()
student2.nok_poeng()
```

uten klasser:
hver for seg



med klasser:
alt i ett

klasser og objekter er en **måte å tenke på** som kan gi oss mer **oversiktelige** og **forståelige** programmer

konstruktør vs metode

konstruktør: __init__

spesiell metode som kalles når vi **lager et nytt objekt**

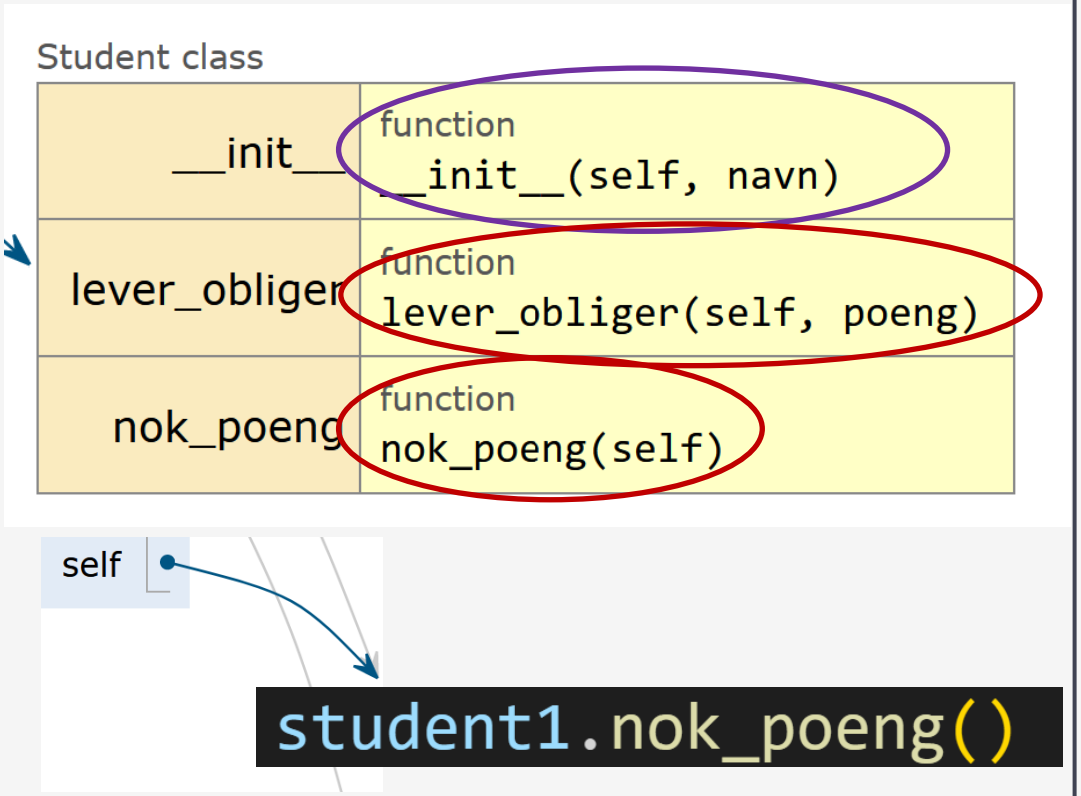
metode

når er objekt **er blitt laget,** kan vi kalle metoder for å la dem gjøre ting

parameteren self

alle metoder har denne parameteren verdien av den er **hele objektet selv**

objektet foran punktum er verdien av **self** når metoden kalles



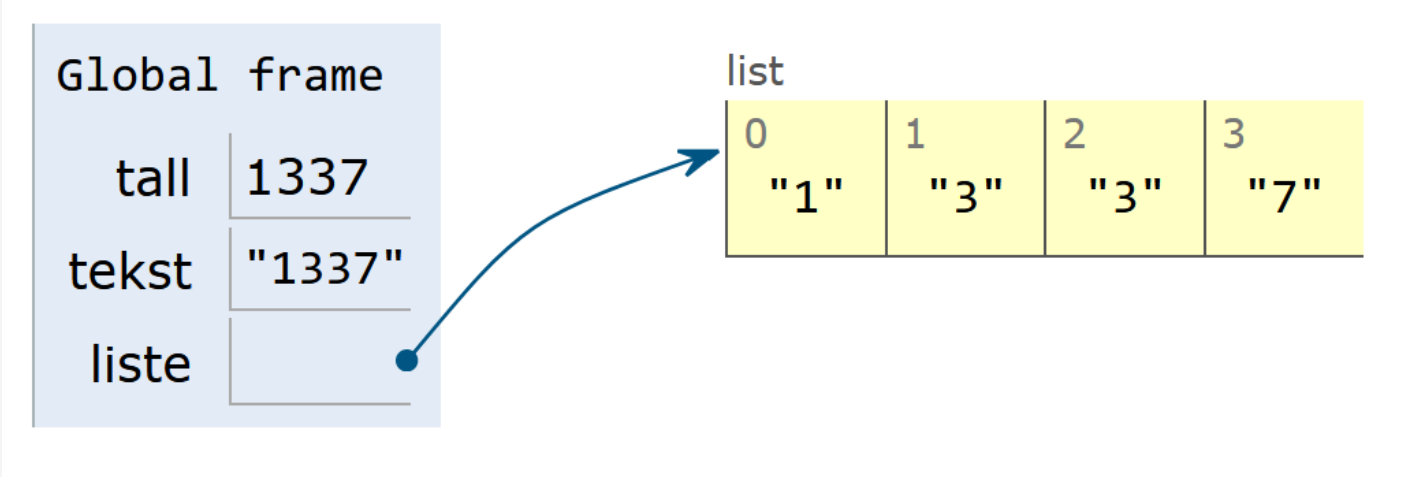
mentimeter:

menti.com

kode: 2649 4712

alle disse verdiene er objekter som her lages med konstruktører:

```
tall = int("1337")
tekst = str(tall)
liste = list(tekst)
```



```
# innebygde objekter kan lages direkte
# (UTEN å bruke konstruktor)
# det klarer ikke våre 'hjemmelagde' objekter
tall = 1337
tekst = "1337"
liste = ["1", "3", "3", "7"]
```

mentimeter:

menti.com

kode: 2862 7664


slidesgo