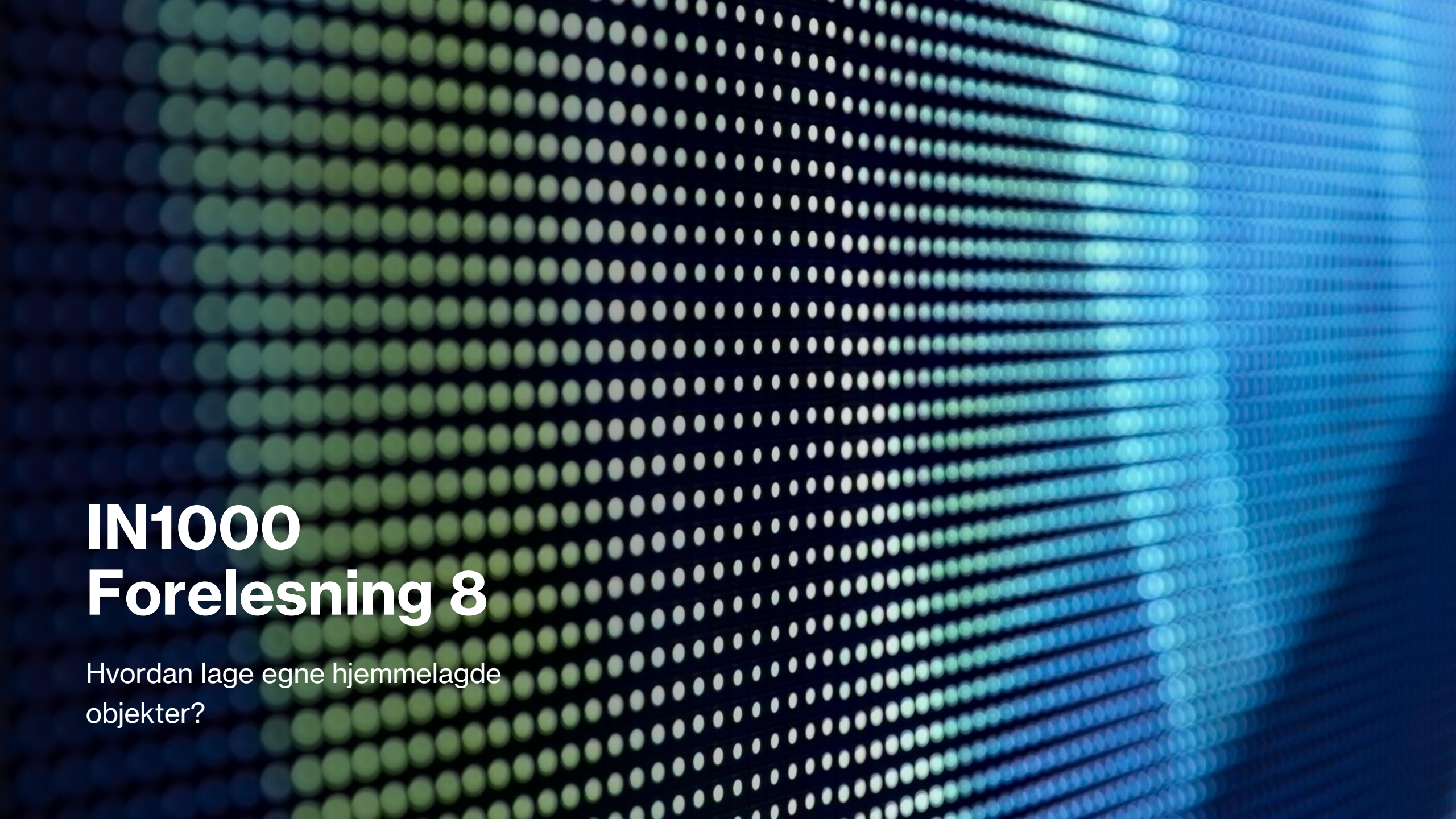


IN1000 Forelesning 8

Hvordan lage egne hjemmelagde objekter?

Syntaks for objekter, oppsummert

- Hvordan finner vi typen til et objekt (når *x* er en variabel)?
  - *print(“x har typen”, type(x))*
- Hvordan lager vi et nytt objekt av typen (eller klassen) *HjemmelagetObjekt*?
  - *x = HjemmelagetObjekt()*
- Noen typer objekter må i tillegg ha **argumenter** for å lages, akkurat som noen funksjoner må ha det (det går an å tenke på *HjemmelagetHeltall* som en slags funksjon som returnerer et nytt objekt av typen *HjemmelagetHeltall*)
  - *y = HjemmelagetHeltall(100)*
- Objekter kan ha variabler (**instansvariabler**): *z = y.verdi*
- Objekter kan ha egne funksjoner (**metoder**) som lar objektet gjøre noe: *y.doble_verdi()*

Viktig om oblig 7

- Siden dagens forelesning ble flyttet en uke fram, har vi ikke gjennomgått alt til oblig 7 når den publiseres senere denne uken
- Alt som mangler vil bli gjennomgått neste mandag (23. mars), i god tid før fristen (onsdag 8. april).
- Dere vil likevel kunne gjøre mye av obligen med det vi har gjennomgått, men ikke alt. Så gjør det dere får til imens og hopp over resten foreløpig, så får dere det dere trenger I neste forelesning.

Relevante læremål

Hva lærer du?

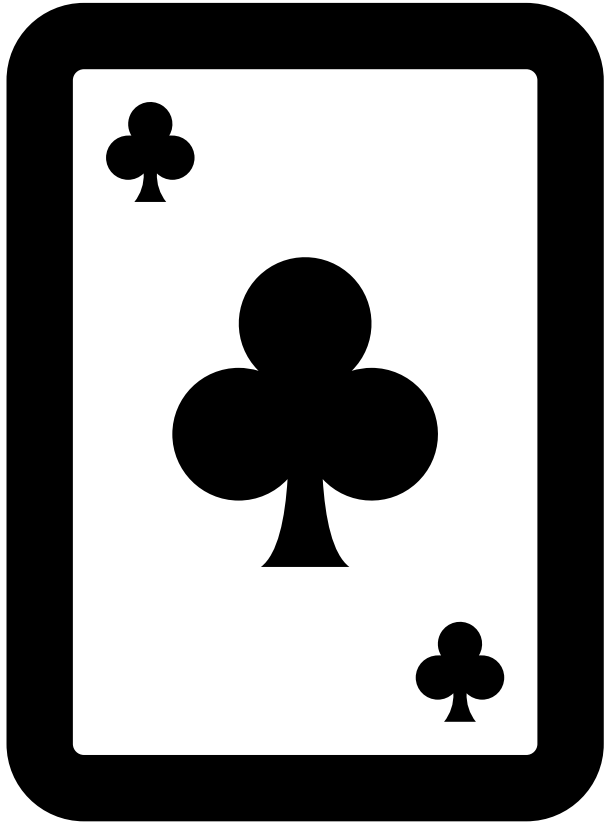
Etter å ha tatt IN1000:

- 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

To typer objekter vi møter ofte

- Enkle **dataobjekter** som bare har instansvariabler med data, og ingen metoder
  - Eksempel: Et *Spillkort*-objekt, som bare har *.type* (hjerter, ruter, kløver, spar) og *.verdi* (2-9)
- En **samling** av flere objekter i en bestemt **rekkefølge**
  - Eksempel: Et *Kortstokk*-objekt, som inneholder mange *Spillkort*-objekter
  - Når vi skal trekke et kort skal vi ta det første (øverste) i bunken
- Det viktigste her er hvordan vi får disse objektene til å fungere riktig *sammen*
  - Mindre viktig, men også relevant: Hvordan vi lager en ny type objekt i Python

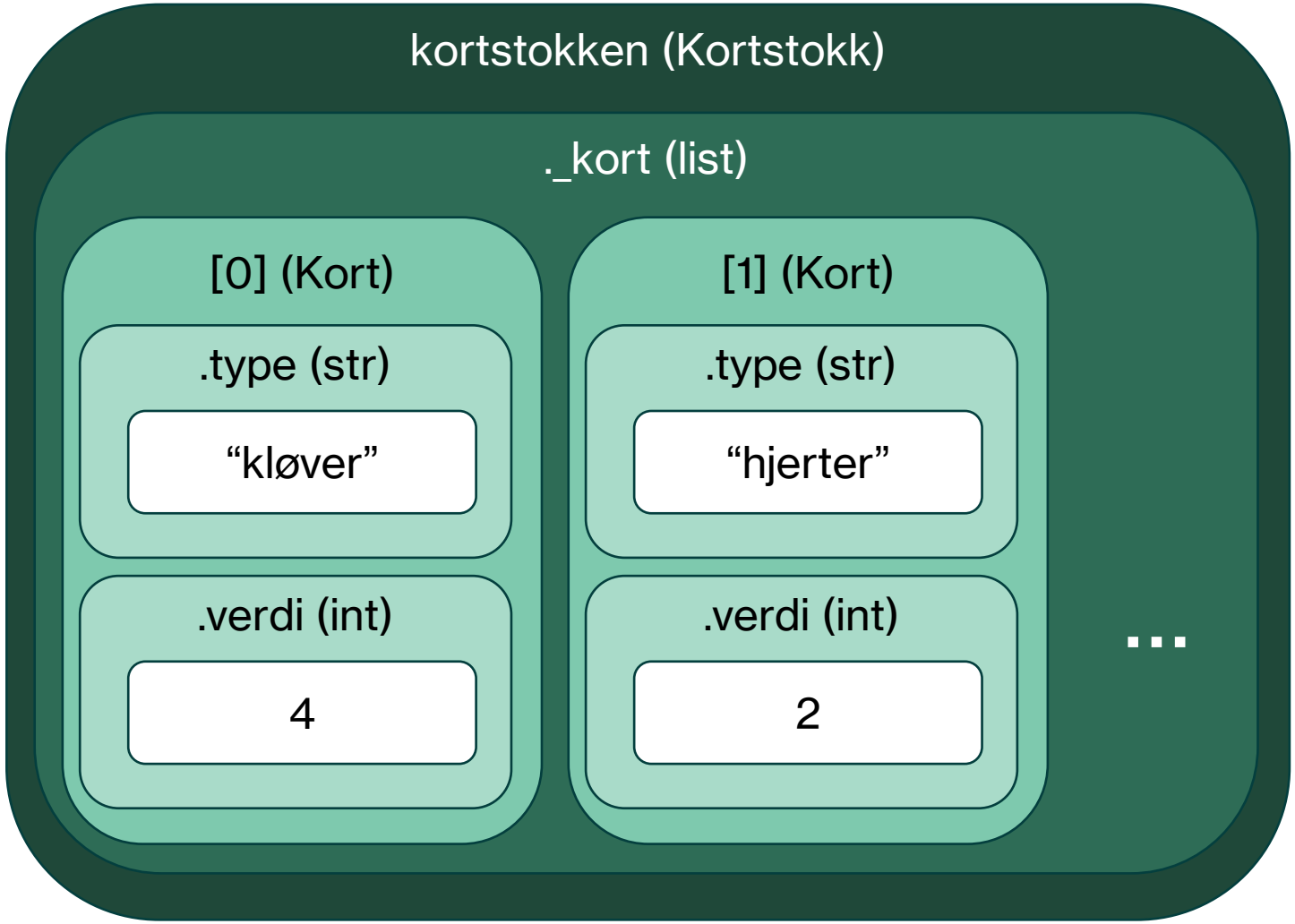
Livekoding: Hjemmelagde objekter



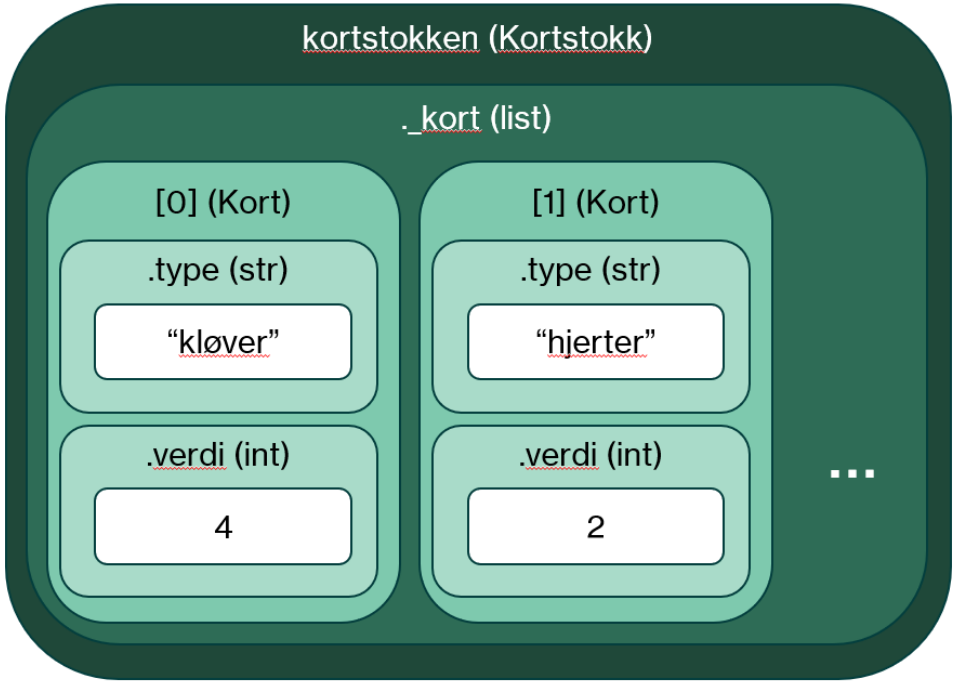
En klasse → mange objekter

- En **klasse** er oppskriften vi bruker på å lage et objekt, som definerer alle metodene objekter av denne typen har
- Det vil si at **klasse** og **type** nesten betyr det samme: alle objekter av samme type er laget med samme oppskrift (klasse)
- Vi kan lage mange kaker av samme oppskrift, og mange objekter av samme klasse
- Disse objektene kan likevel være i forskjellige **tilstander** med ulike **data**: Selv om både *a* og *b* er av klassen **str**, så kan verdien til *a være “Hei”* og verdien til *b* være *“Slartibartfast spiller digeridoo”*, for eksempel
- Litt som at vi kan ha en kake som er halvspist og pyntet med nonstop og en pyntet med seigmenn som det bare er tatt en liten bit av (men begge er sjokoladekaker)

Implementasjon (menneskelig perspektiv)

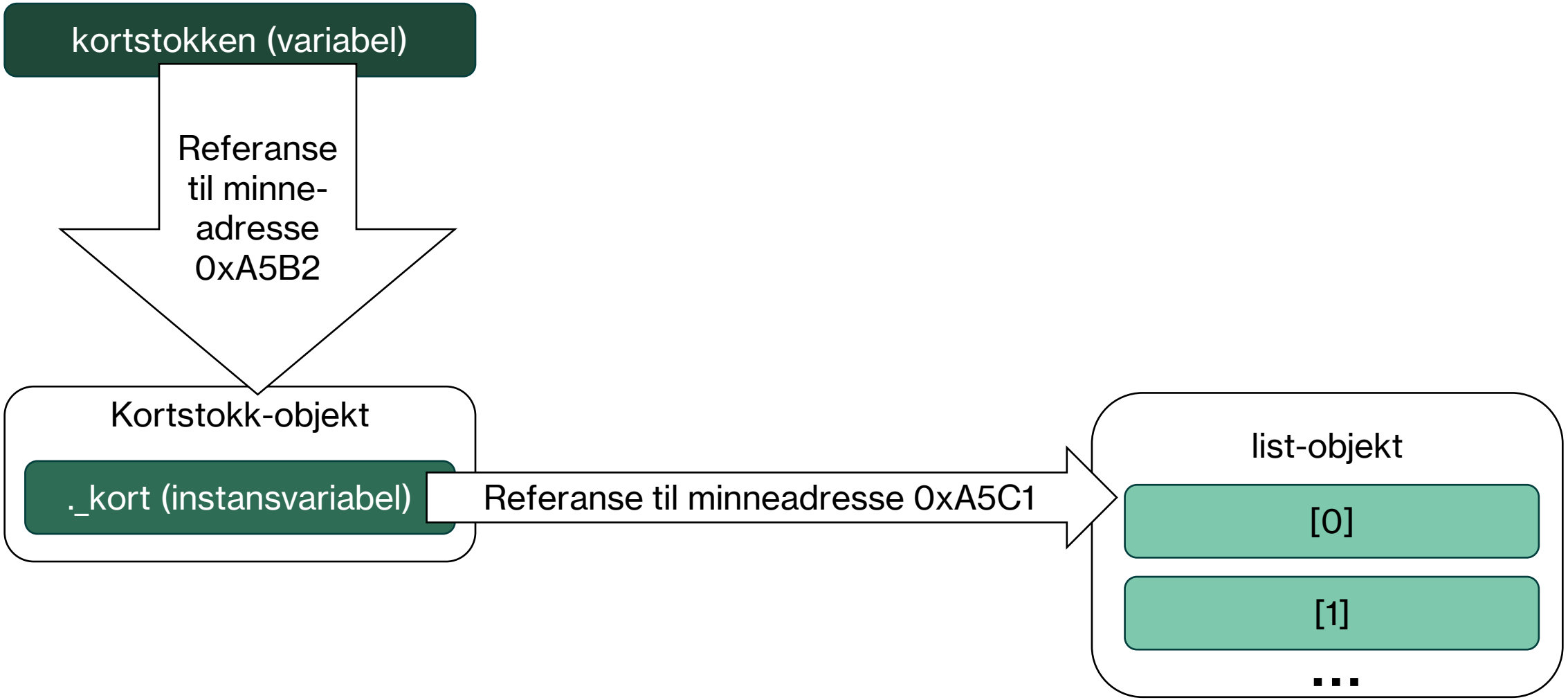


Implementasjon (menneskelig perspektiv)

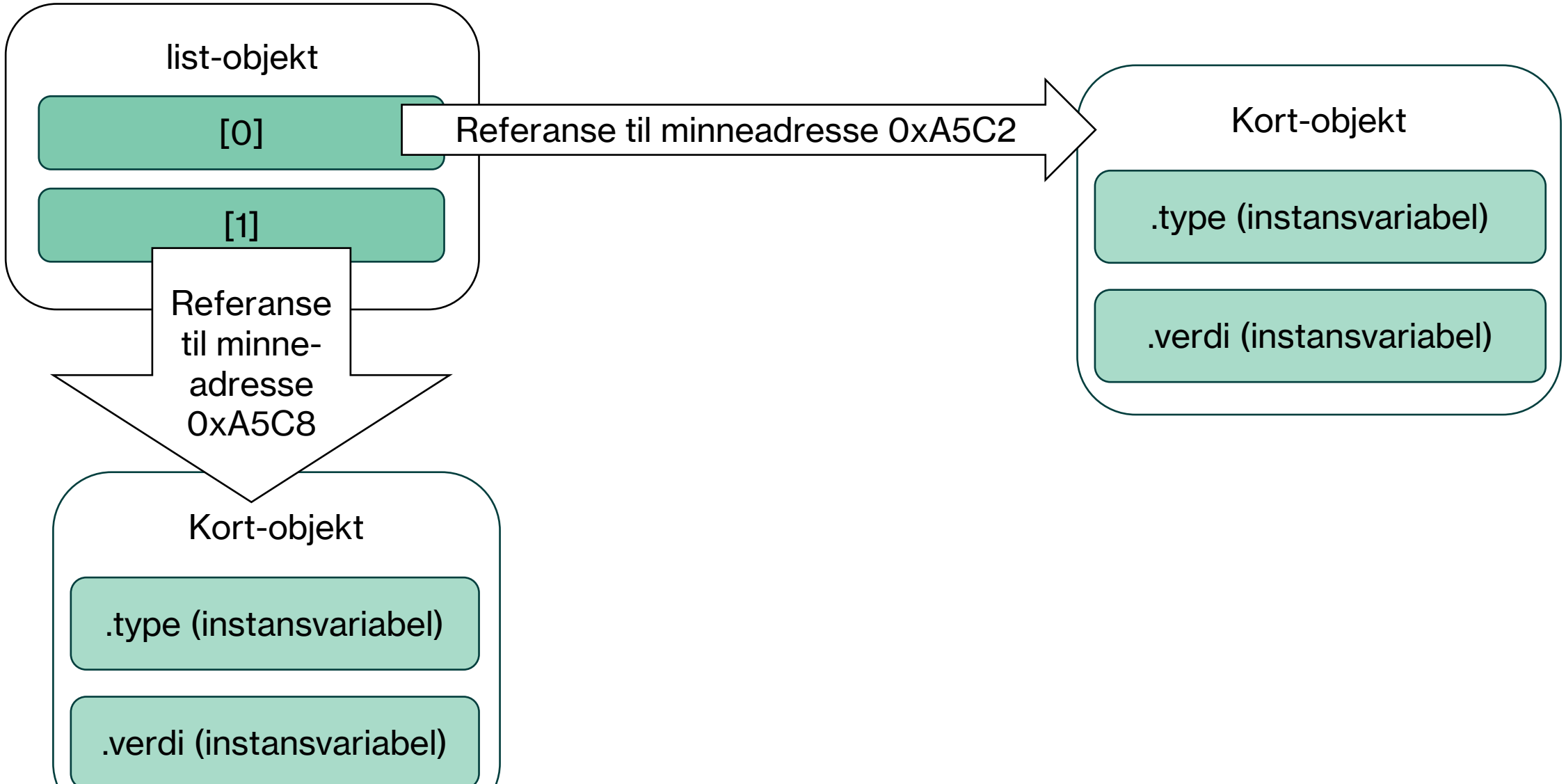


- *kortstokken* er et objekt av type **Kortstokk**
- *kortstokken._kort* er et objekt av type **list**
- *kortstokken._kort[1]* er et objekt av type **Kort**
- *kortstokken._kort[1].verdi* er et objekt av typen **int** (med verdien 2)

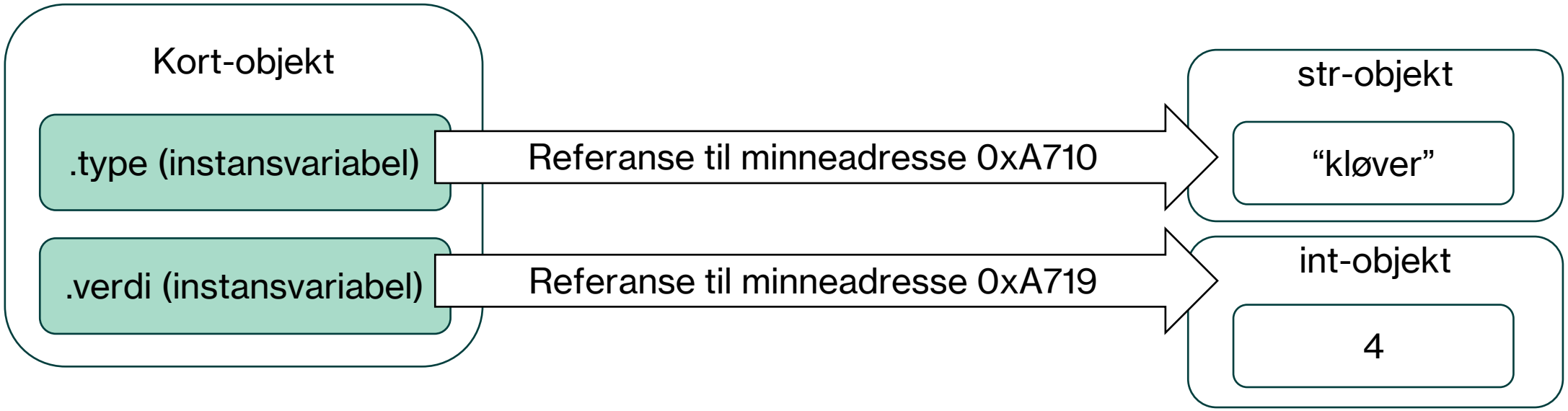
Implementasjon (maskinperspektiv)



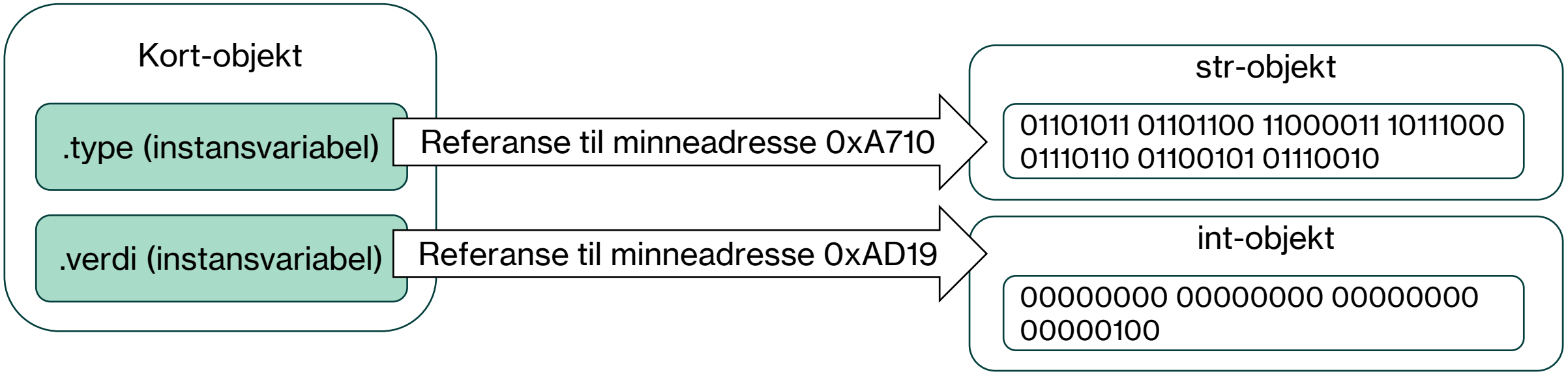
Implementasjon (maskinperspektiv)



Implementasjon (maskinperspektiv)



Implementasjon (maskinperspektiv)



Variabel → referanse → verdi (objekt)

En **variabel** er et menneskevennlig navn vi gir til noe i programmet vårt

Variabelen inneholder en **referanse** (en minne-adresse) som forteller maskinen hvor den skal slå opp i minnet når vi bruker variabelen senere

På denne minne-adressen finner vi et **objekt** (som vi kaller variabelens **verdi**)

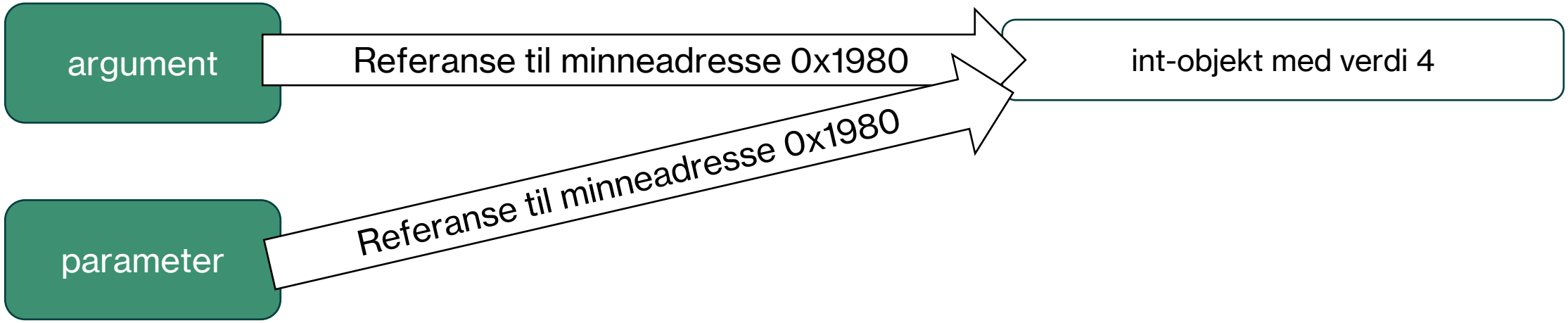
Objektet har en bestemt **type** (eller **klasse**) som bestemmer hvilke **metoder** dette objektet har (ting det selv vet hvordan det skal gjøre)

Objektet kan ha **instansvariabler**, som har referanser til *andre* objekter, osv...

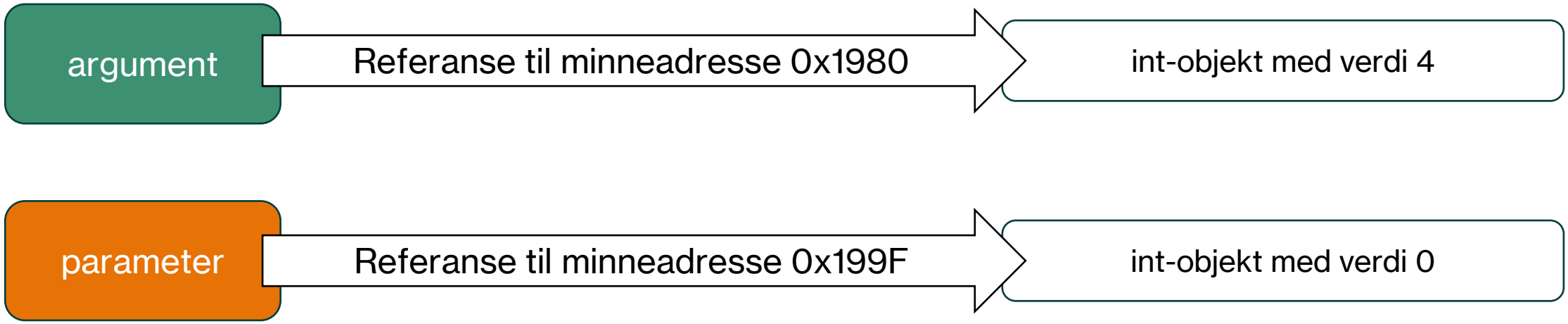
Livekoding: Referanser



Ulike variabler kan ha lik referanse (til samme objekt)



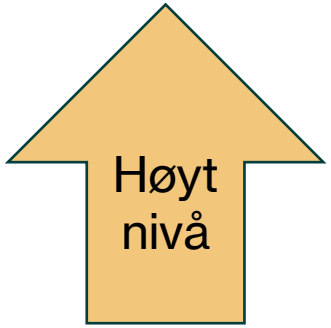
Men *parameter = ...* endrer kun referansen til *parameter*, ikke referansene til andre variabler



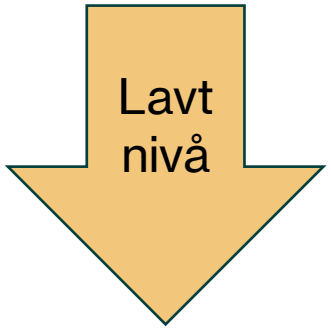
Funksjonskall i Python

- Når vi kaller Python-funksjoner **kopieres ikke** objektene vi sender inn som argument til funksjonen – det lages ikke nye objekter
- I stedet får funksjonen en **referanse** (minneadresse) til objektene
- Hvis vi gir et objekt som kan endre tilstand (f.eks. liste) som argument og **endrer** dette inni funksjonen, så vil også variabelen i hovedprogrammet påvirkes av endringen
- Dette kalles **call by reference**
- I andre programmeringsspråk kan vi støte på noe som heter **call by value**, som vil si at funksjoner noen ganger helt automatisk lager sin egen kopi av objekter
- I slike tilfeller vil endringer på kopien ikke påvirke originalen
- (vi **kan** lage kopier av objekter i Python også, men da må vi gjøre det manuelt)

Tre nivå å lese programkode på



- Vi startet forelesningen på **menneskelig nivå**: Tenker på objektene som klumper med data som kan gjøre enkelte ting selv
  - Opptatt av **hva** objektene gjør (grensenitt), ikke **hvordan** de gjør det (implementasjon)
  - “Hva handler dette programmet om?” (navn på variabler / funksjoner til stor hjelp her!)
- Vi fortsatte på **syntaks-nivå**: Hvordan skriver vi Python-koden for å gjøre dette
  - Forrige forelesning: Skille mellom funksjon, metode og instansvariabel
  - Denne forelesningen: Hvordan skrive en klasse (implementasjon)
- Og avsluttet på **maskin-nivå**: Hvordan er tilstanden til programmer og objekter inne i maskinen akkurat nå (implementasjon)
  - Og hvordan endrer tilstanden seg når en ny kodelinje kjøres
  - Python Tutor er et godt hjelpemiddel for å øve opp dette



Den virkelig store misforståelsen i IN1000

- “Du kommer langt på eksamen med å kunne skrive riktig syntaks for å lage en klasse”
- Nei
- Det har bare vært 5% av disse to forelesningene, og ca. 5% av poengene på eksamen
- **Å skrive en enkel klasse er svært enkelt, og belønningen er tilsvarende enkel**
- “Hvis syntaks-nivået alene er så lite viktig, hva er viktig da?”
- Å kunne **bruke** forskjellige objekter som lages **sammen** og utnytte det hvert objekt kan gjøre **selv** (tjenestene det tilbyr) for å løse problemer effektivt (menneskelig nivå)
- Å kunne forstå hvordan objektene skal **kobles sammen** og hvilke **datastrukturer** (lister, ordbøker etc.) som er fornuftige å bruke internt i en klasse (maskin-nivå)
- Begge disse tingene er fokus videre for de neste forelesningene