in1000 uke 8 dypdykk

viktig forståelse fra forrige ukes pensum

læremål

0 grensesnitt
og innkapsling

1 referanser

0

grensesnitt

og innkapsling

er det verdt å lære seg funksjoner og klasser?

- “det virker så mye enklere å skrive programmer uten dette, heller enn å måtte lære seg det”

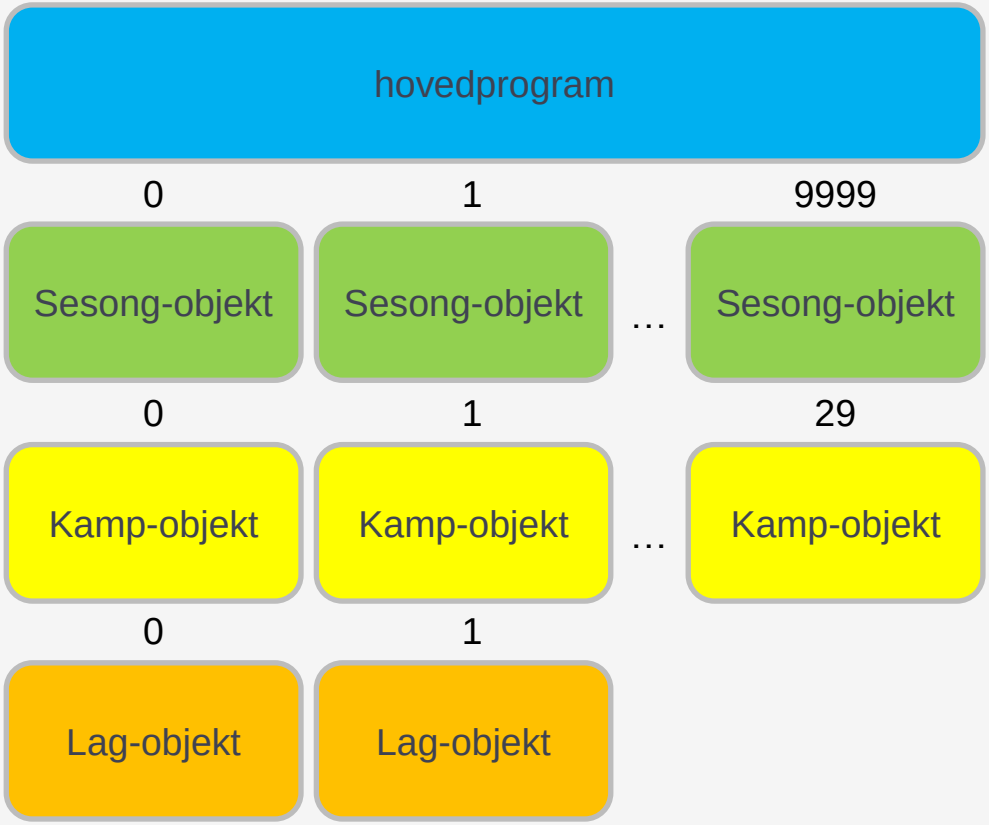
- utfordring: vi lærer det **før** programmene blir så store at vi virkelig **trenger** funksjoner og klasser

men så blir programmet stort og komplisert…

- “hjelp, jeg ser ikke skogen for bare trær… ☹”
- da trenger vi **flere nivå** (hovedprogram, klasser, metoder) for å finne fram og få oversikt

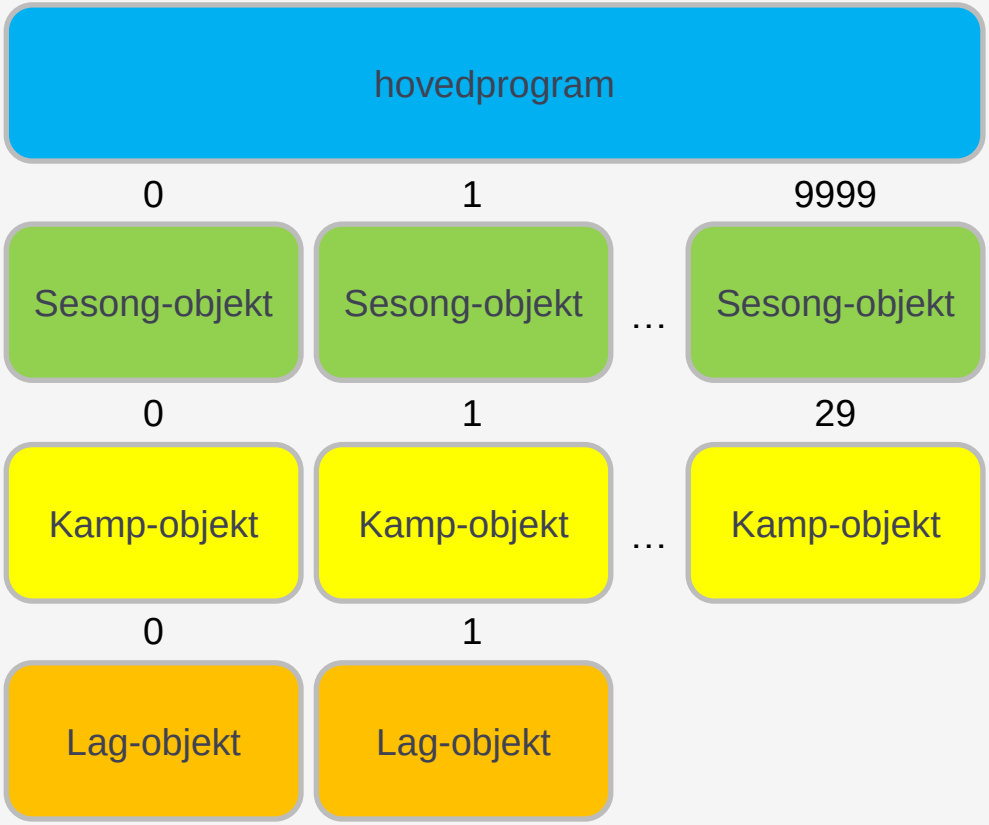
eksempel: simulere 10 000 fotballsesonger

- hovedprogrammet inneholder Sesonger
- en Sesong inneholder 30 Kamper
- en Kamp inneholder 2 Lag



innkapsling: hver del får ansvar for hver sine ting

- **hovedprogram:** lage 10 000 sesonger med samme lag og regne ut sannsynligheter
- **Sesong:** når møter lagene hverandre (lest fra fil), hvordan ble tabellen til slutt
- **Kamp:** simulerer resultatet (med litt tilfeldigheter)
- **Lag:** hvor gode er et lag i angrep, i forsvar, har de stor hjemmefordel…



hovedprogrammets perspektiv

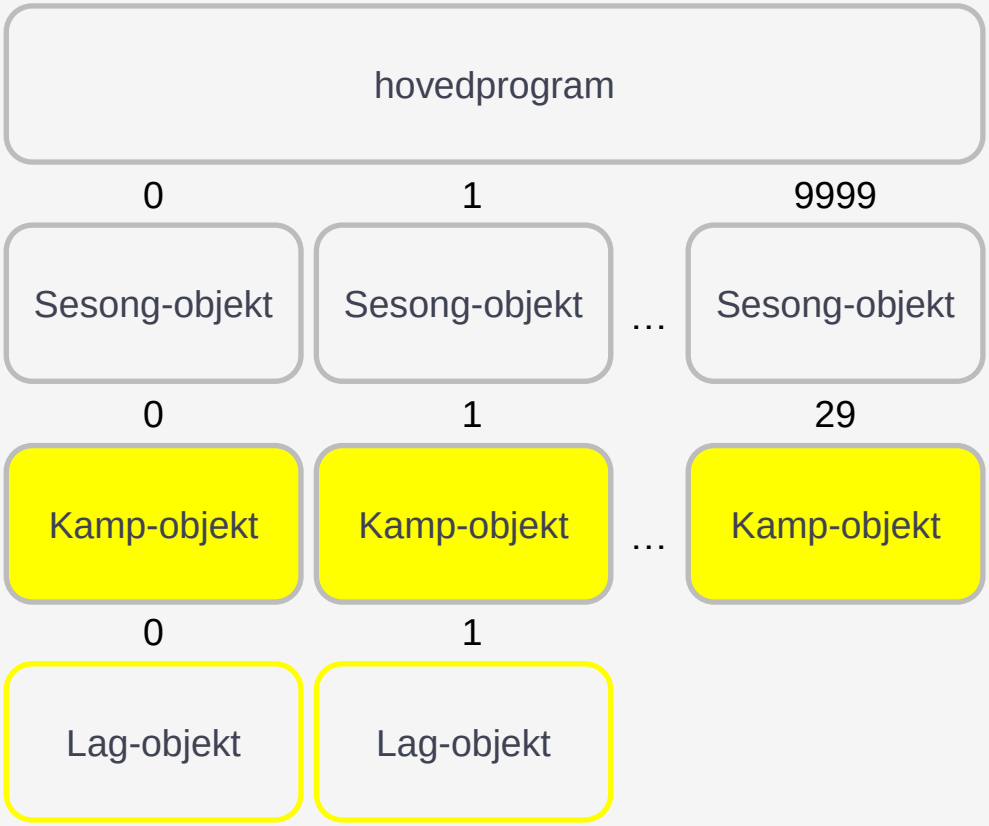
- lager 10 000 sesonger med samme lag
- **grensesnitt:** ber Sesong-objektene om å spille en sesong
- **grensesnitt:** henter resultater (tabell) fra hver Sesong etterpå
- regner ut hvor sannsynlig hver enkelt plassering er for hvert enkelt lag

Sesong-objektenes perspektiv

- leser terminliste fra fil og setter opp Kamp-objekter for hver kamp
- **grensesnitt:** ber Kamp-objektene om å simulere kamp
- **grensesnitt:** henter resultatet fra hver Kamp etterpå
- regner ut hvor mange poeng og hvilken plassering hvert lag får i tabellen

Kamp-objektenes perspektiv

- **grensesnitt:** henter informasjon om hvor gode lagene er fra Lag-objektene
- simulerer en fotballkamp med litt tilfeldigheter og produserer et sluttresultat
- (samme Kamp i forskjellige Sesonger får ikke samme utfall, fordi det er tilfeldig hver gang Kampen spilles)



Lag-objektenes perspektiv

- inneholder informasjon om lagets navn og hvor gode de er i angrep, forsvar og hjemmebanefordel
- har metoder for å gi denne informasjonen til klasser på høyere nivå

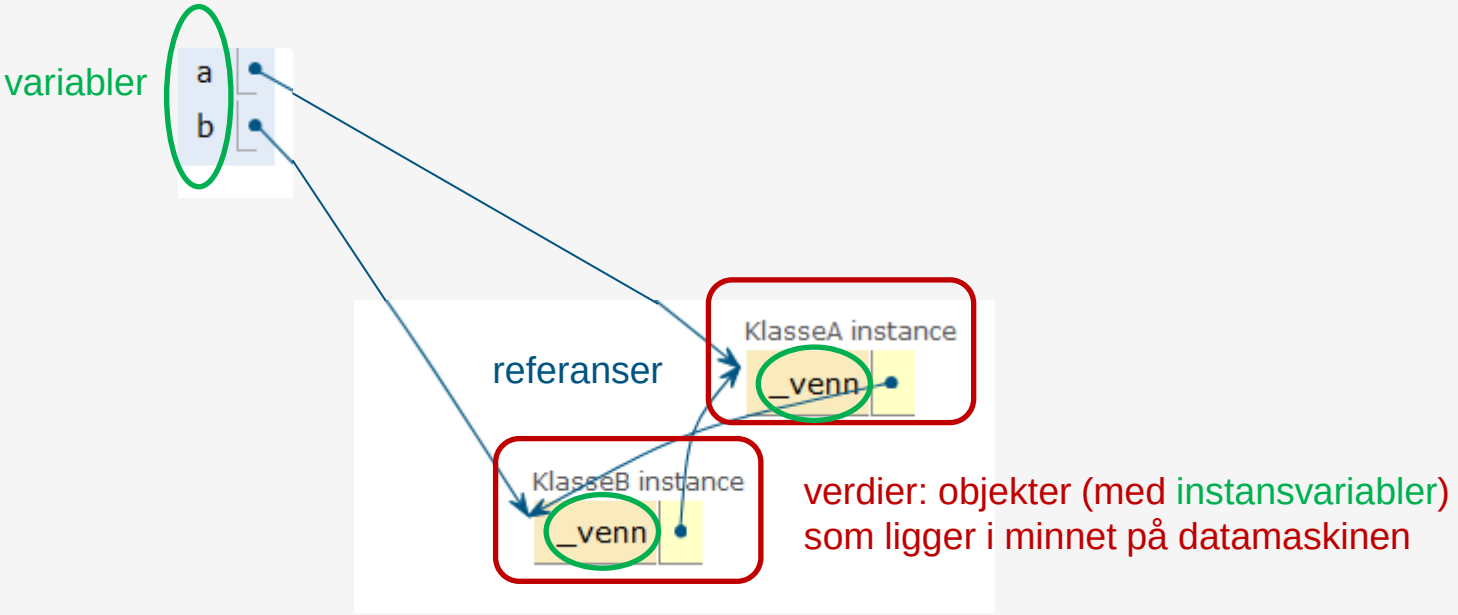
vi trenger ikke tenke på **alle** detaljene samtidig, bare de som er viktige på nivået vi er på akkurat nå

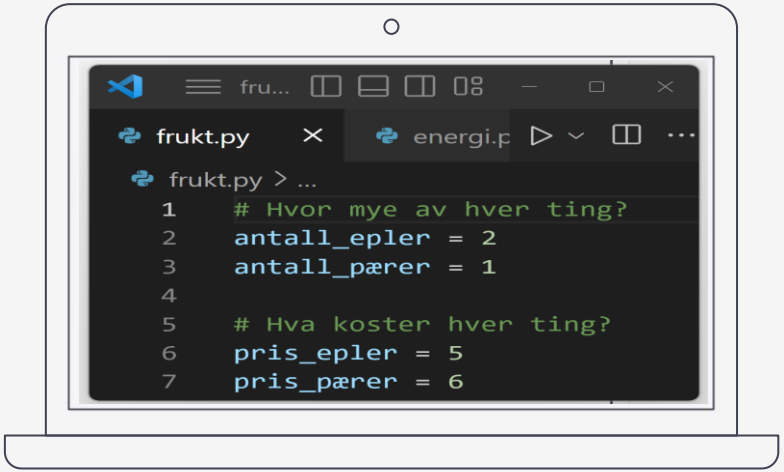
1

referanser

mer om variabler og objekter

pilene i python tutor representerer det vi kaller **referanser**





live-koding

hvorfor referanser er et viktig konsept

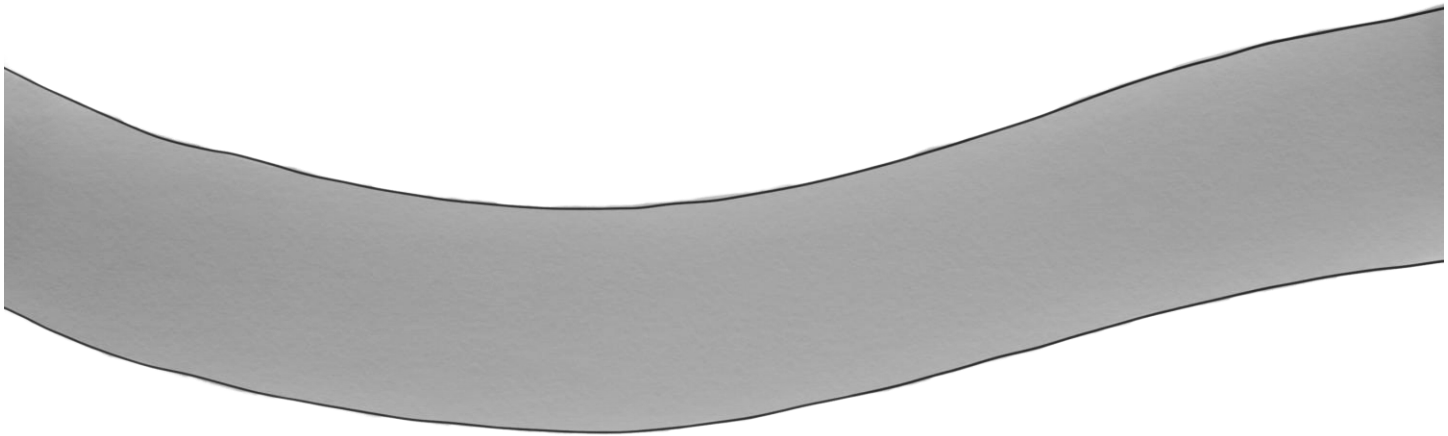
hvorfor_referanser.py
(legges ut i timeplanen på emnesiden etter forelesningen)

husk å se på koden i [python tutor]

Vi trenger å holde orden på tre ting

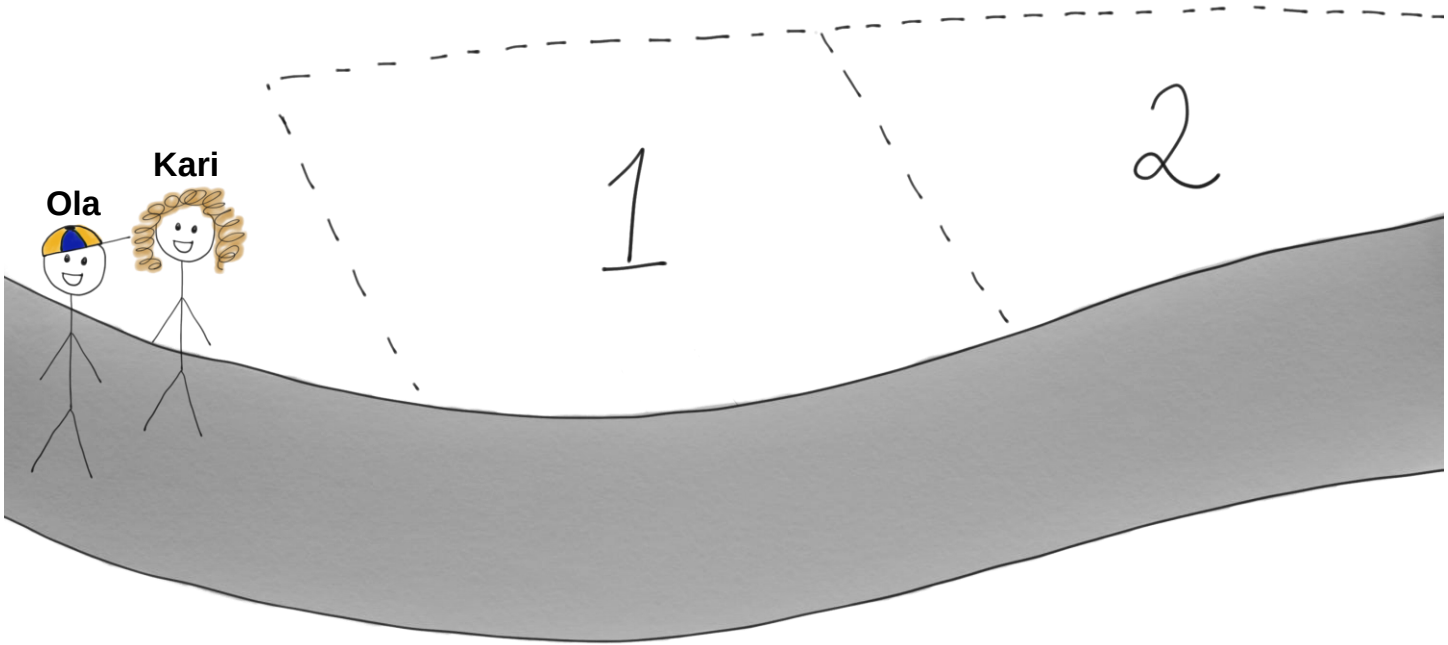
- **Objekt**: har egne verdier av instansvariable (som kan endres)
  - Her illustrert som et farget hus (som kan males om)
- **Referanse**: et tall som representerer en adresse i minnet
  - Her illustrert som et skilt med gatenummer (for en tomt)
- **Variabel**: en variabel kan tilordnes en referanse
  - Her illustrert som strekperson som står ved et gatenummer

Minnegaten



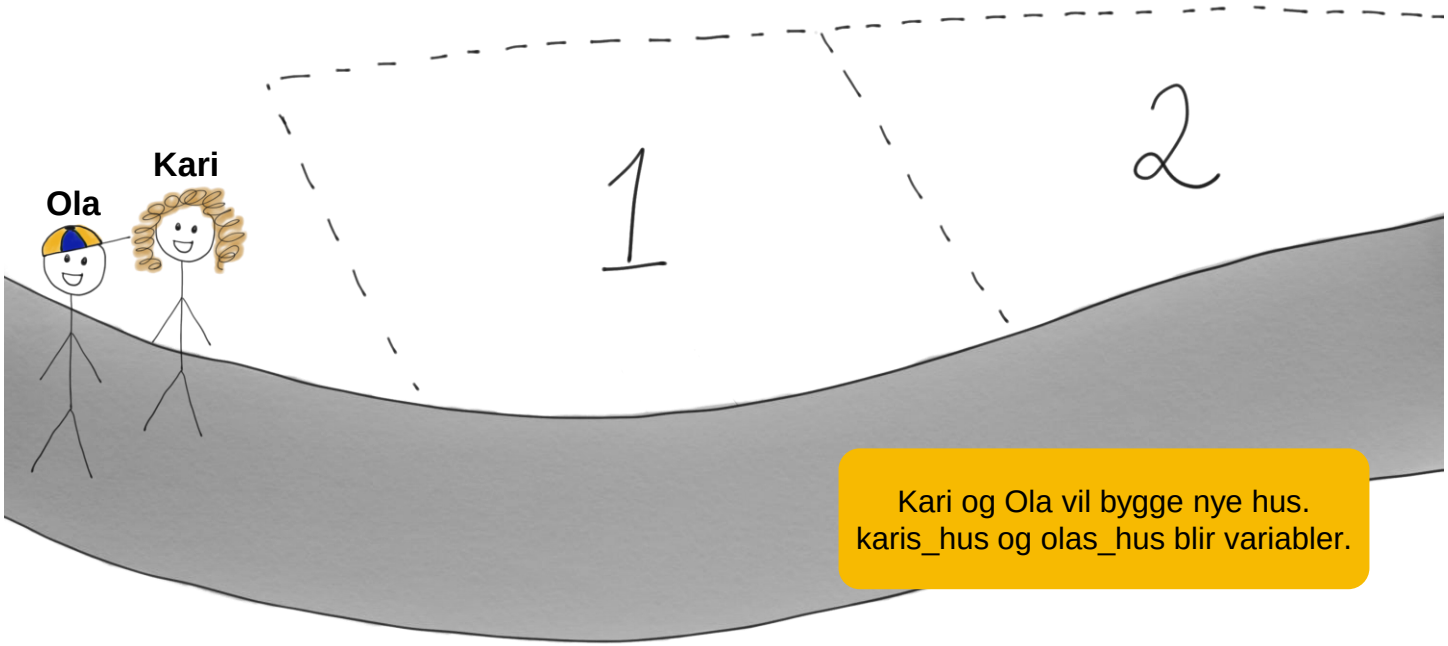
Minnegaten

Ledige tomter = Ledig minne



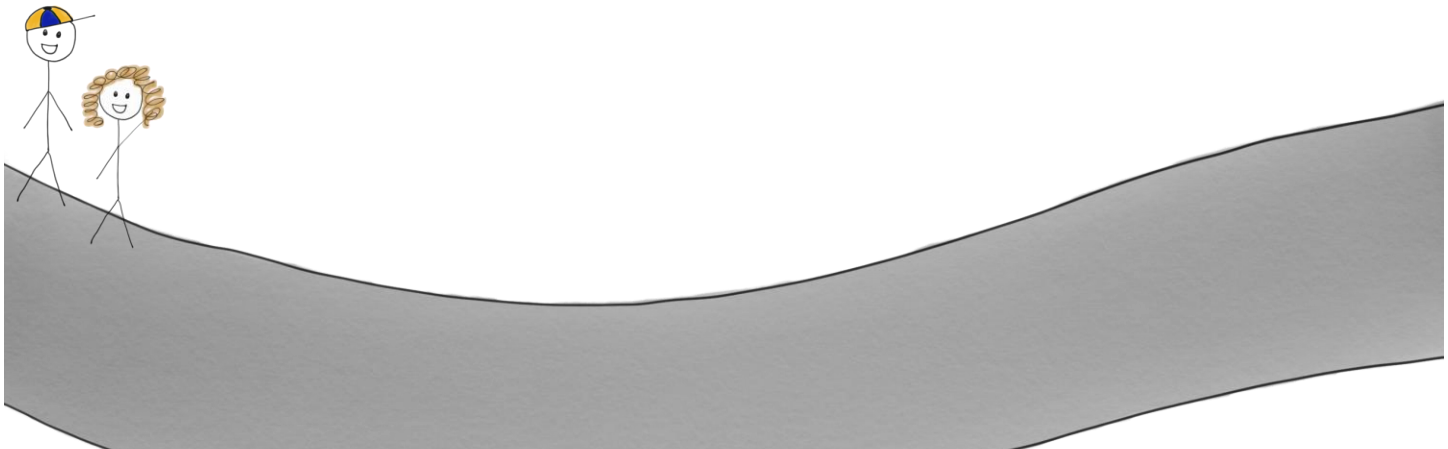
Minnegaten

Ledige tomter = Ledig minne



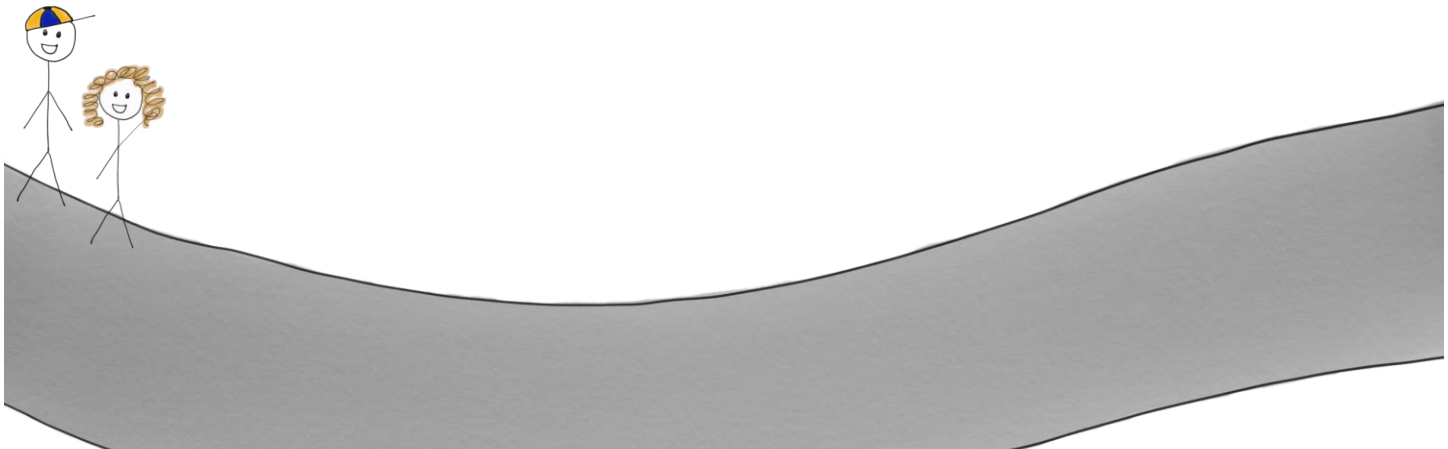
Kari og Ola vil bygge nye hus.
karis_hus og olas_hus blir variabler.

karis_hus = Hus("blå")



karis_hus = Hus("blå")

Først bygges det et blått hus på en ledig tomt



karis_hus = Hus("blå")

Først bygges det et blått hus på en ledig tomt

karis_hus = ~~Hus("blå")~~ 1

karis_hus = ~~Hus("blå")~~ 1

Deretter blir Karis hus satt til å være det blå huset

karis_hus = ~~Hus("blå")~~ 1

Deretter blir Karis hus satt til å være det blå huset

karis_hus = ~~Hus("blå")~~ 1

olas_hus = Hus("rød")

karis_hus = ~~Hus("blå")~~

olas_hus = Hus("rød")

Nå må det bygges det et rødt hus på en ledig tomt

karis_hus = Hus("blå")

olas_hus = Hus("rød")

karis_hus = ~~Hus("blå")~~

olas_hus = ~~Hus("rød")~~

Så blir Olas hus satt til å være det røde huset

karis_hus = ~~Hus("blå")~~

olas_hus = ~~Hus("rød")~~

karis_hus = ~~Hus("blå")~~

olas_hus = ~~Hus("rød")~~

olas_hus = karis_hus

karis_hus = ~~Hus("blå")~~

olas_hus = ~~Hus("rød")~~

olas_hus = karis_hus

Ola sitt hus er det samme som Kari sitt hus

karis_hus = ~~Hus("blå")~~

olas_hus = ~~Hus("rød")~~

olas_hus = karis_hus

Ola sitt hus er det samme som Kari sitt hus

karis_hus = ~~Hus("blå")~~

olas_hus = ~~Hus("rød")~~

olas_hus = ~~karis_hus~~

olas_hus.mal("gul")

olas_hus.mal("gul")

Ola vil male sitt hus gult

olas_hus.mal("gul")

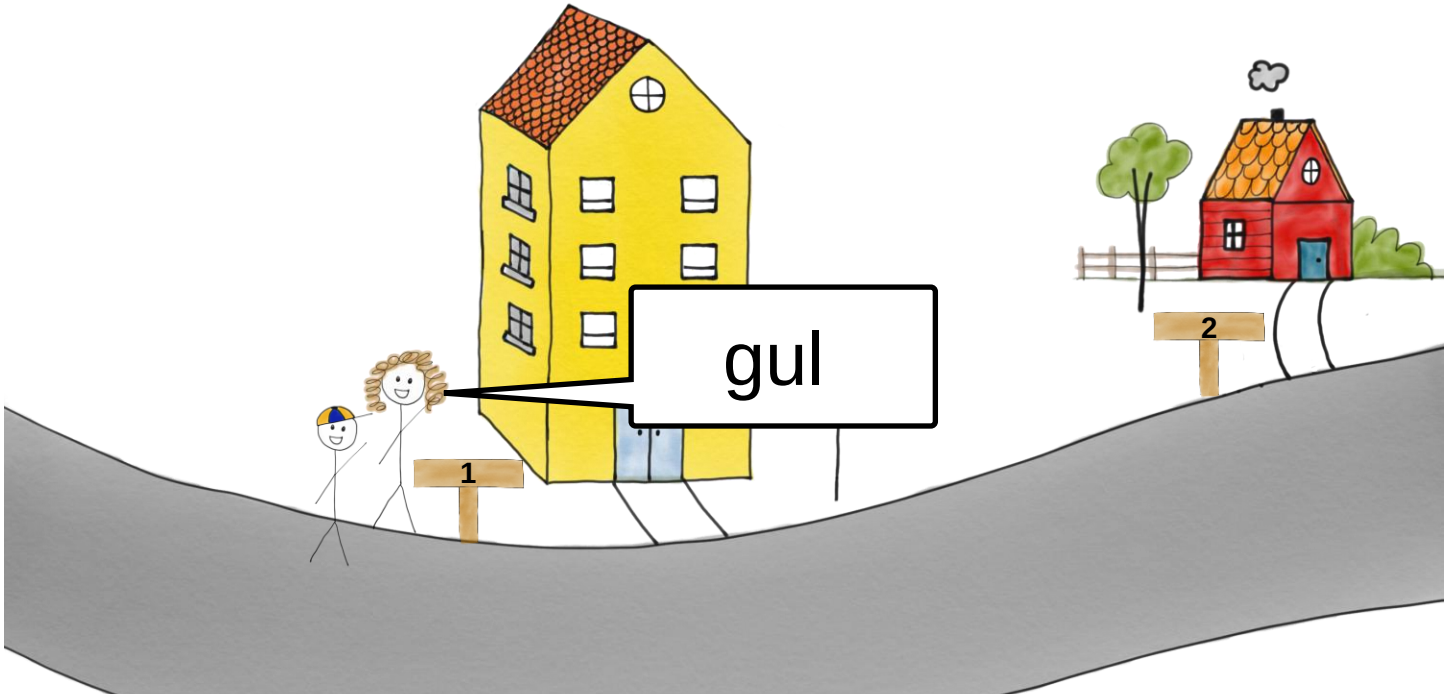
```
olas_hus.mal("gul")

karis_hus.hent_farge()
```

Hvilken farge har Kari sitt hus nå?

olas_hus.mal("gul")

karis_hus.hent_farge()

Hvilken farge har Kari sitt hus nå?



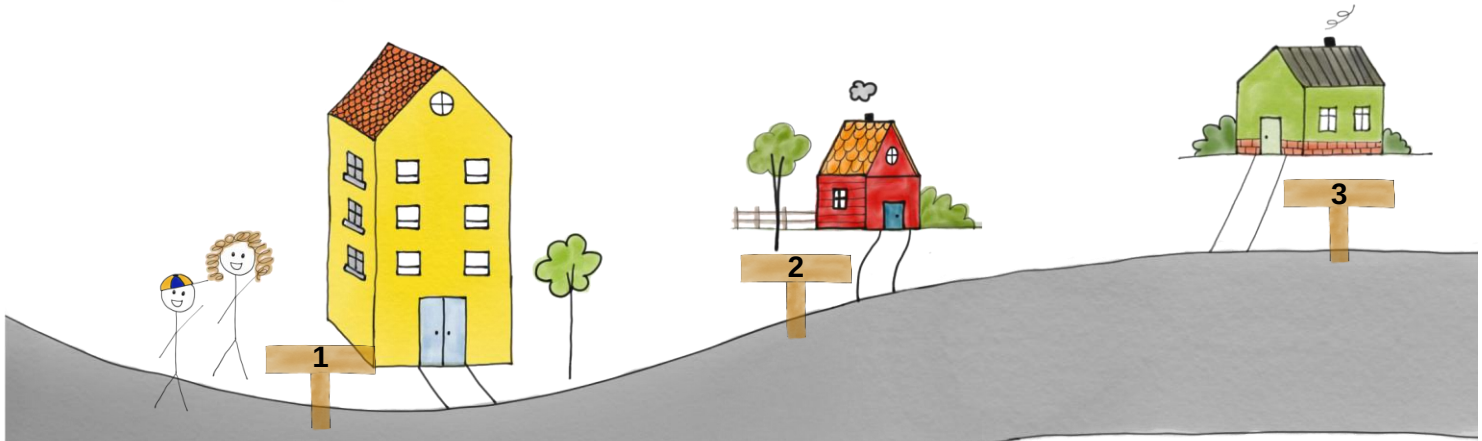
```
olas_hus.mal("gul")

karis_hus.hent_farge()

olas_hus = Hus("grønn")
```

```
olas_hus.mal("gul")

karis_hus.hent_farge()

olas_hus = Hus("grønn")
```

Nå må det bygges et nytt grønt hus på en ny tomt

```
olas_hus.mal("gul")

karis_hus.hent_farge()

olas_hus = Hus("grønn")
```

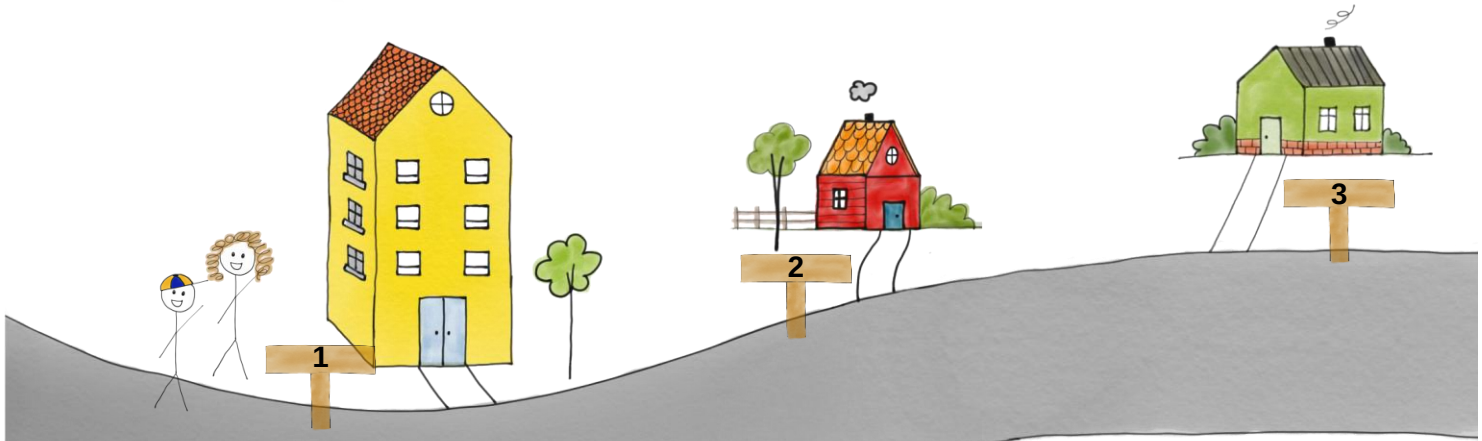
```
olas_hus.mal("gul")

karis_hus.hent_farge()

olas_hus = Hus("grønn")
```



olas_hus.mal("gul")

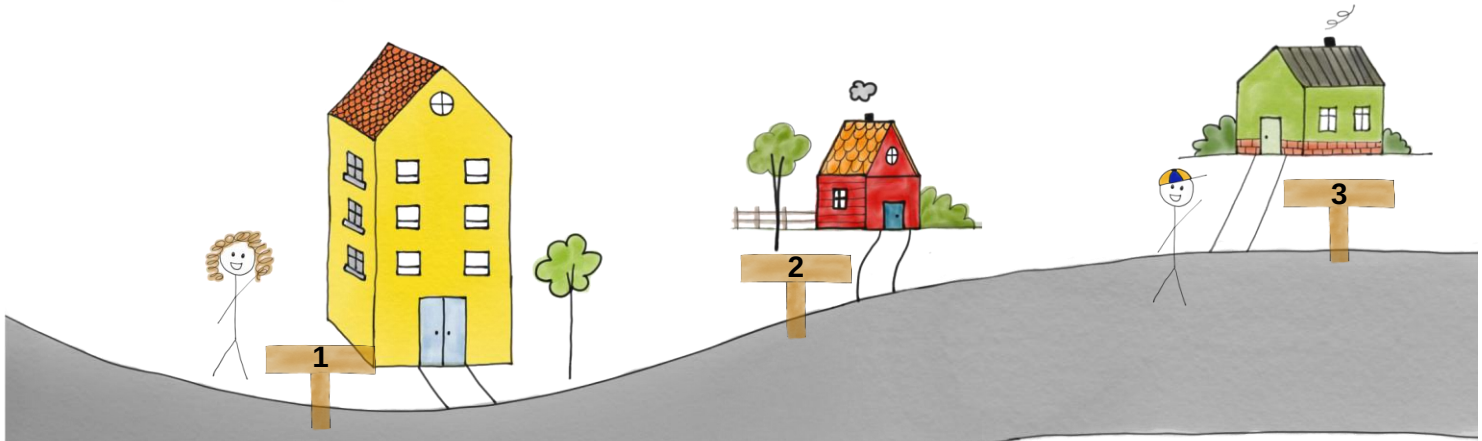
karis_hus.hent_farge()

olas_hus = ~~Hus("grønn")~~ 



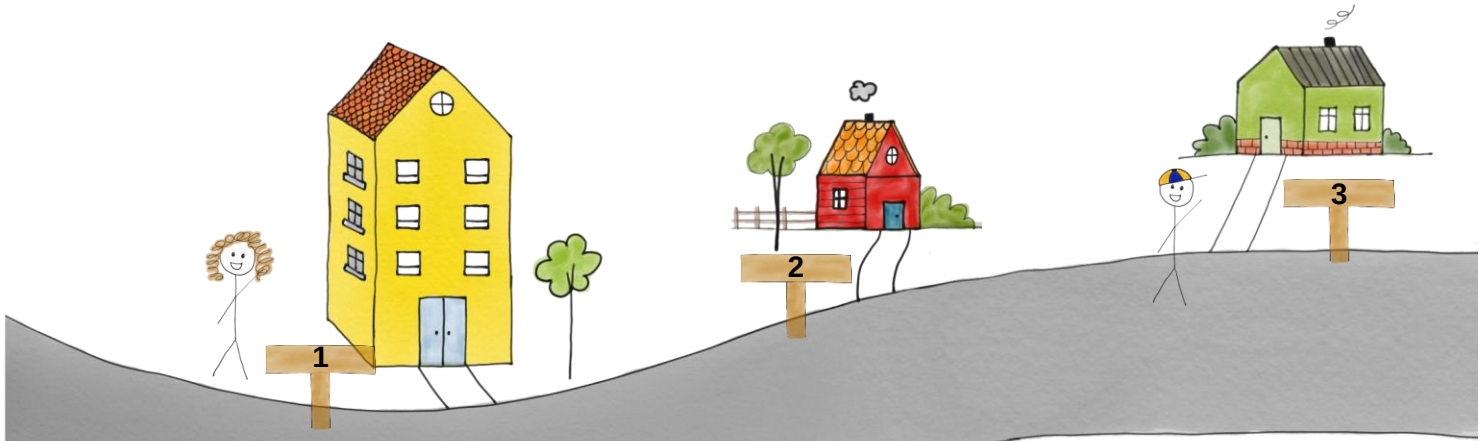
```
olas_hus.mal("gul")

karis_hus.hent_farge()

olas_hus = ~~Hus("grønn")~~ 3
```

Ola sitt hus er endret til å være det nye huset



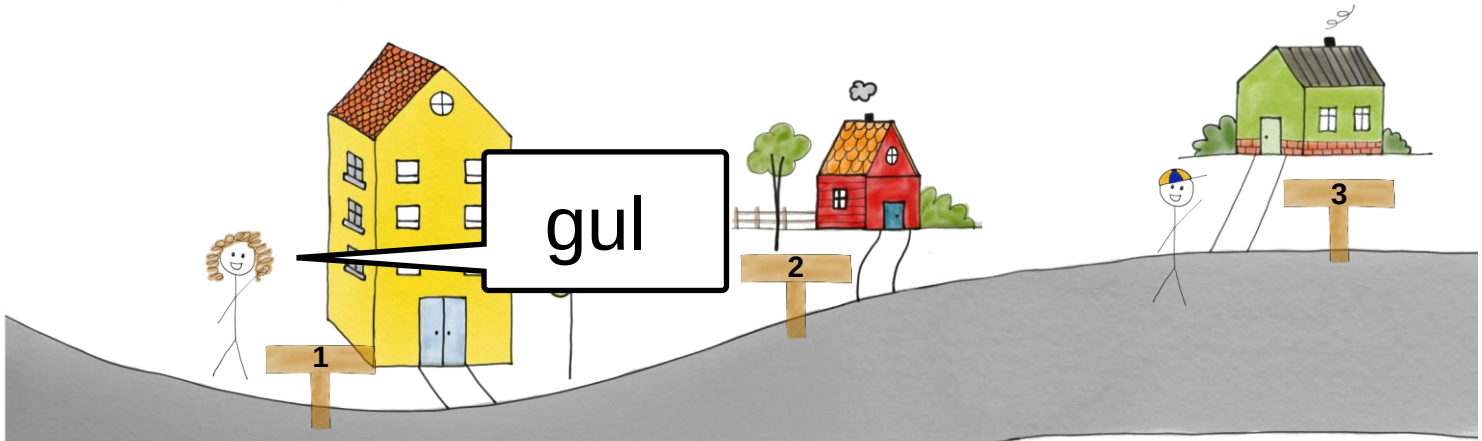
```
olas_hus.mal("gul")

karis_hus.hent_farge()

olas_hus = ~~Hus("grønn")~~ 3

karis_hus.hent_farge()
```

Hvilken farge har Kari sitt hus nå?



```
olas_hus.mal("gul")

karis_hus.hent_farge()

olas_hus = ~~Hus("grønn")~~

karis_hus.hent_farge()
```

Hvilken farge har Kari sitt hus nå?



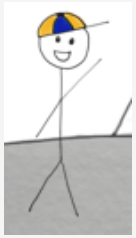
referanser (minne-adresser) kan vi i utgangspunktet se ved å printe objektene

```
print(olas_hus)
print(karis_hus)
```


```
<__main__.Hus object at 0x00000214D83EFEE0>
<__main__.Hus object at 0x00000214D83EFA90>
```

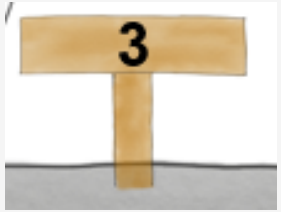
variabel `olas_hus`

(merkelapp vi setter på en verdi)



referanse `0x00000214D83EFA90`

(hvor verdien ligger i minnet)



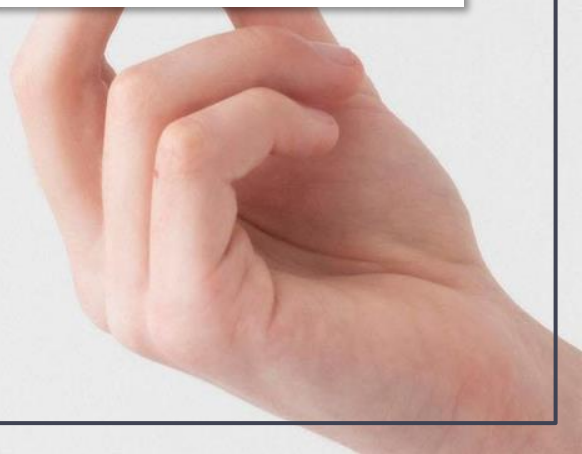
objekt `__main__.Hus object`

(selve verdien)

mentimeter:

menti.com

kode: 6461 2384



in1000
uke 9
nytt lærestoff

oversikt og forståelse i kontekst

læremål

0

samlinger av objekter

1

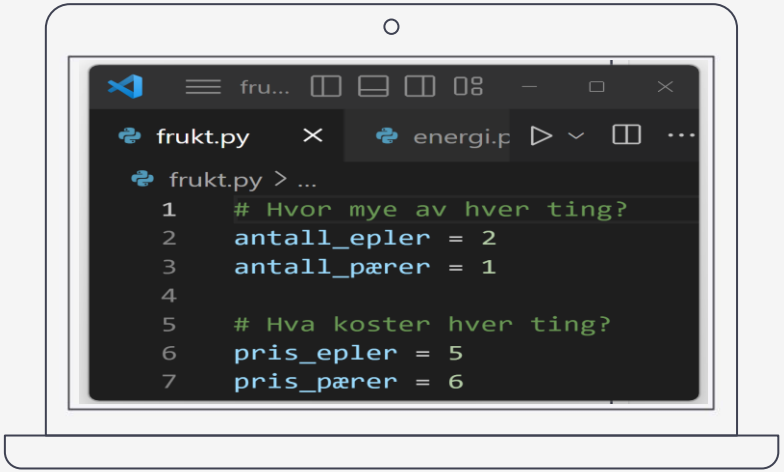
“magiske” metoder

(+midtveisevaluering)

0

samlinger

med referanser til flere objekter



live-koding

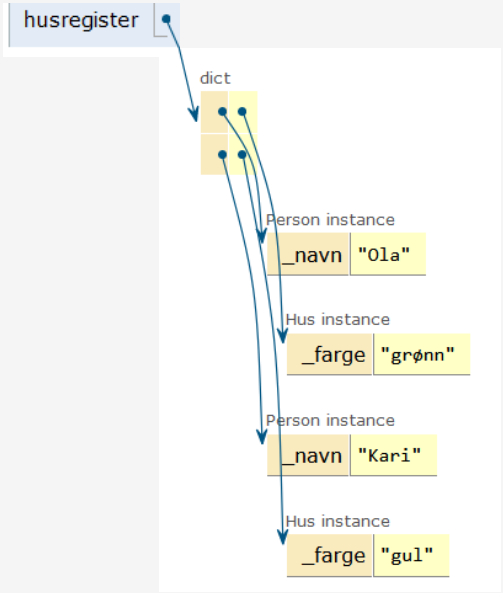
hvordan lage en klasse og objekter

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

variabel → en referanse → ett objekt

samling → flere referanser → hver til ett objekt

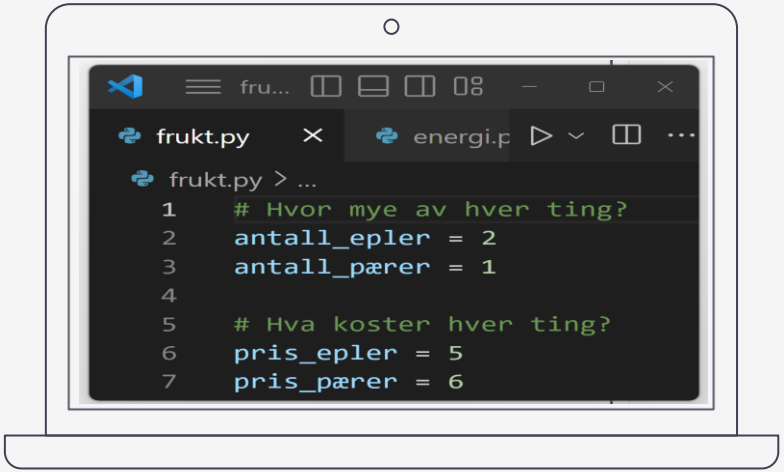
```
1  class Hus:
2      def __init__(self, farge):
3          self._farge = farge
4
5      def mal(self, farge):
6          self._farge = farge
7
8      def hent_farge(self):
9          return self._farge
10
11 class Person:
12     def __init__(self, navn):
13         self._navn = navn
14
15     def hent_navn(self):
16         return self._navn
17
18 husregister = {}
19
20 husregister[Person("Ola")] = Hus("grønn")
21 husregister[Person("Kari")] = Hus("gul")
```



1

“magiske”

metoder som
__init__, __str__ og __eq__



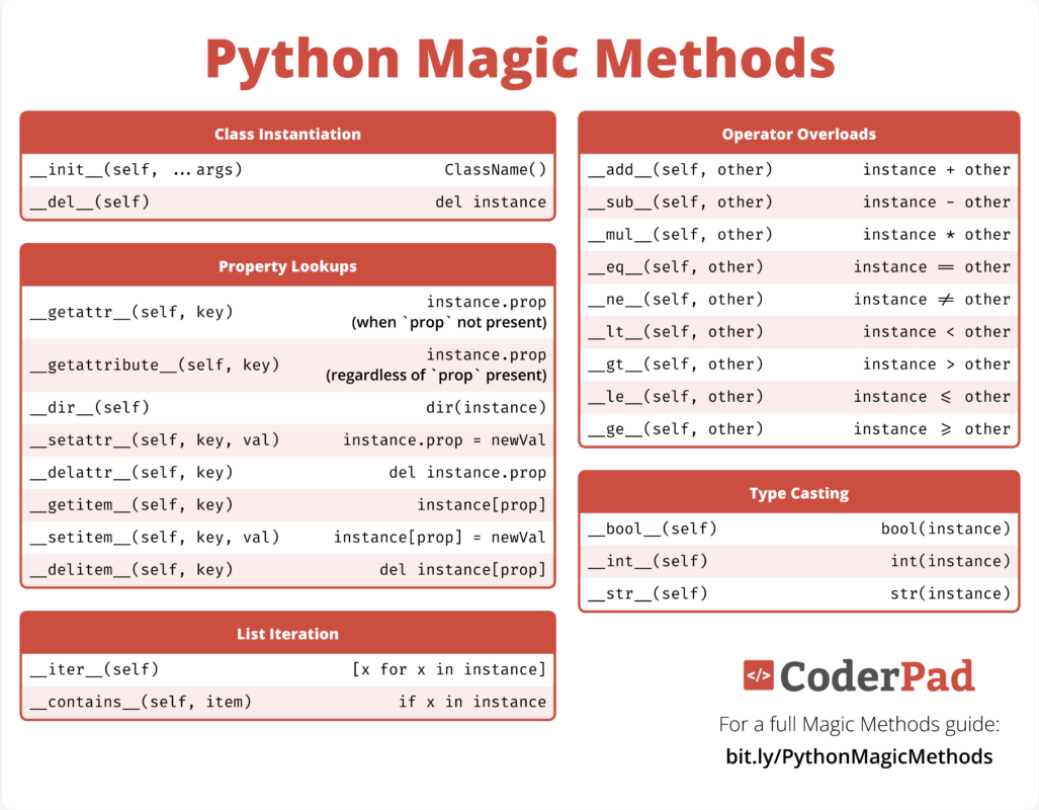
live-koding

hvordan lage en klasse og objekter

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

husk å se på koden i [python tutor]

(det finnes mange andre magiske metoder)



Python Magic Methods

Class Instantiation

`__init__(self, ...args)`	`ClassName()`
`__del__(self)`	`del instance`

Property Lookups

`__getattr__(self, key)`	`instance.prop` (when `prop` not present)
`__getattribute__(self, key)`	`instance.prop` (regardless of `prop` present)
`__dir__(self)`	`dir(instance)`
`__setattr__(self, key, val)`	`instance.prop = newVal`
`__delattr__(self, key)`	`del instance.prop`
`__getitem__(self, key)`	`instance[prop]`
`__setitem__(self, key, val)`	`instance[prop] = newVal`
`__delitem__(self, key)`	`del instance[prop]`

List Iteration

`__iter__(self)`	`[x for x in instance]`
`__contains__(self, item)`	`if x in instance`

Operator Overloads

`__add__(self, other)`	`instance + other`
`__sub__(self, other)`	`instance - other`
`__mul__(self, other)`	`instance * other`
`__eq__(self, other)`	`instance == other`
`__ne__(self, other)`	`instance ≠ other`
`__lt__(self, other)`	`instance < other`
`__gt__(self, other)`	`instance > other`
`__le__(self, other)`	`instance ⩽ other`
`__ge__(self, other)`	`instance ⩾ other`

Type Casting

`__bool__(self)`	`bool(instance)`
`__int__(self)`	`int(instance)`
`__str__(self)`	`str(instance)`

CoderPad

For a full Magic Methods guide:
bit.ly/PythonMagicMethods

viktige begreper nevnt i dag:

- **innkapsling:** skjule lavnivå-detaljer
- **grensesnitt:** hvordan kommunisere med andre nivå (andre klasser/hovedprogram)
- **variabel:** navn/merkelapp på et objekt
- **samling:** samme, for flere objekter
- **referanse:** adressen til objektet i minnet
- **objekt:** klump med data i minnet (som vi kan kalle klassens metoder på)

midtveisevaluering:
https://nettskjema.no/a/331080


slidesgo