GRUPPETIME
UKE 8

IN1000 GRUPPE 25 - 18.10.21

PLAN FOR GRUPPETIMEN

- Oblig 6 - hvordan gikk det?
- Jobbe sammen på MetroRetro
- Opprette og bruker objekter - steg for steg med Minnegaten eks.
- Legge til en samling av objekter i person.py

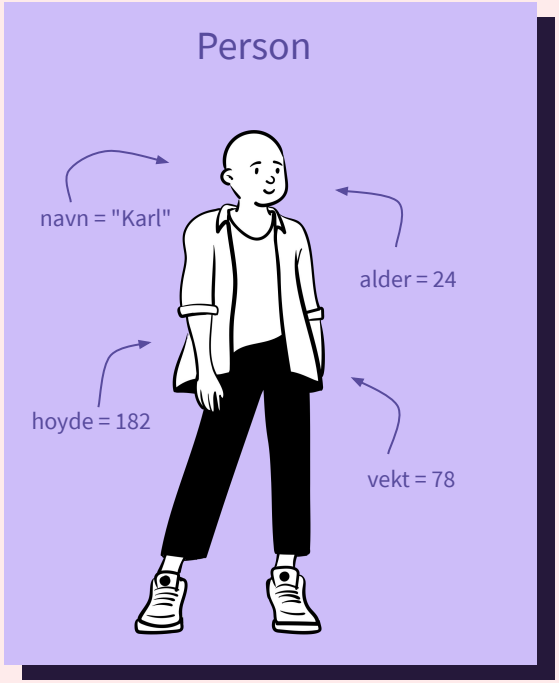
LÆRDOMMER FRA OBLIG 6

- Husk underscore på instansvariabler
  - Dette viser at variablene er private og ikke synlige utenfor klassen.
- Pass på å utføre **ALT** en oppgave ber om - kjipt å miste poeng på eksamen fordi du har glemt å gjøre noe
- I hund.py skulle _metthet ikke tas inn som et parameter
  - self._metthet = 10
- `if dato_objekt.sjekkDag(15)`
  - ```
    def sjekkDag(self, sjekkDag):
        return self._dag == sjekkDag
    ```
    ```

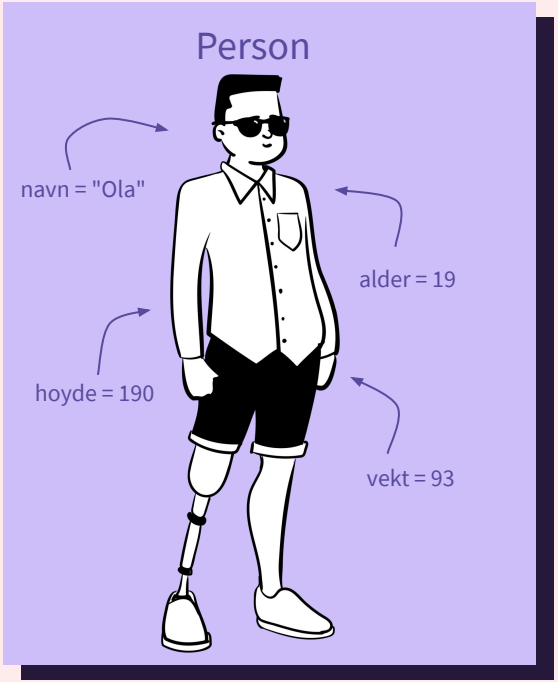
LÆRINGSMÅL UKE 8

- Forstå (mer av) hva som skjer bak kulissene når vi oppretter og bruker objekter
- Kjenne til forskjellen på å endre en referansevariabel og å endre objektet den refererer til
- Kunne skrive programmer med samlinger av (referanser til) objekter
- Kunne sette seg inn i enkle programmer med flere klasser og objekter som refererer til andre objekter

OOP: lage objekter av en klasse




```
karl = Person("Karl", 24, 78, 182)
```




```
ola = Person("Ola", 19, 93, 190)
```

```
class Person:
    def __init__(self, navn, alder, vekt, hoyde):
        self._navn = navn
        self._alder = alder
        self._vekt = vekt
        self._hoyde = hoyde

    def get_navn(self):
        return self._navn

    def set_navn(self, nytt_navn):
        self._navn = nytt_navn

    def skriv_ut_hilsen(self):
        print("Hei, jeg heter",  self._navn, "og jeg er",
              self._alder, "aar gammel")

    def hoyere_enn(self, annen_person):
        if self._hoyde > annen_person._hoyde:
            return True
        return False
```

klassenavn

instansvariabler

konstruktør

instansmetode

klassedefinisjon

```
karl = Person("Karl", 24, 78, 182)
ola = Person("Ola", 19, 93, 190)
```

Opprettelse av nye objekter

```
print(ola.hoyere_enn(karl))
print(karl.hoyere_enn(ola))
```

Kall på instansmetode

```python
class Person:
    def __init__(self, navn, alder, vekt, hoyde):
        self._navn = navn
        self._alder = alder
        self._vekt = vekt
        self._hoyde = hoyde

    def get_navn(self):
        return self._navn

    def set_navn(self, nytt_navn):
        self._navn = nytt_navn

    def skriv_ut_hilsen(self):
        print("Hei, jeg heter",  self._navn, "og jeg er",
              self._alder, "aar gammel")

    def hoyere_enn(self, annen_person):
        if self._hoyde > annen_person._hoyde:
            return True
        return False
```

implementasjon
person.py

```python
karl = Person("Karl", 24, 78, 182)
ola = Person("Ola", 19, 93, 190)

karl.skriv_ut_hilsen()
print(ola.hoyere_enn(karl))
print(karl.hoyere_enn(ola))
```

grensesnitt
test_person.py

enkle typer vs. sammensatte typer

Enkle typer

Inneholder en type verdi

- Heltall (1, 45, -1)
- Sannhetsverdier (True, False)

Sammensatte typer

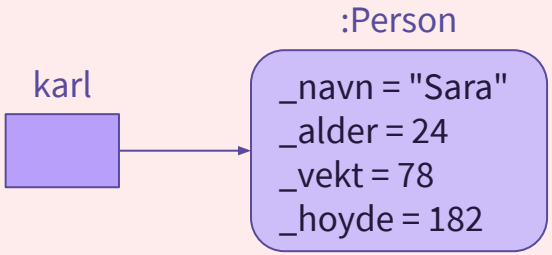
Kan inneholde mer enn en type verdi

- Lister: `.append()` `.pop()`
- Ordbøker: `.items()` `.keys()`
- Strenger: `.upper()` `.strip()`
- Filobjekter: `.write()` `.close()`

vi kan lage våre egne sammensatte typer med klasser!

REFERANSER TIL OBJEKTER

- **Referansevariabler** er variabler som inneholder referansen (adressen i minnet) til objektet
  - Minneadressen (innholdet i referansevariablen) er en **objektreferanse**
- Referansevariabler kan brukes for å kalle på metoder i objektet
  - `karl.set_navn("Sara")` → Nå endrer vi på objektet som karl referer til

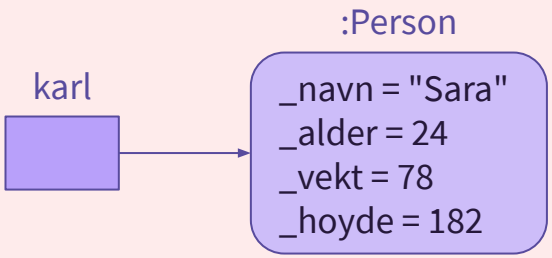



```
karl = Person("Karl", 24, 78, 182)
karl.set_navn("Sara")
```

Læringsmål: kjenne til forskjellen på å endre en referansevariabel vs. å endre objektet den refererer til

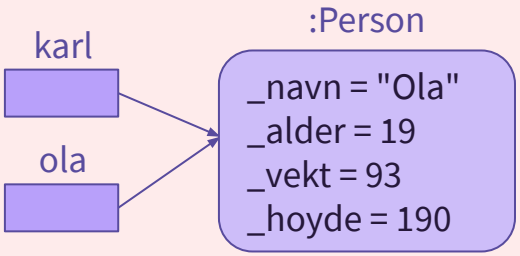
```
karl = Person("Karl", 24, 78, 182)
karl.set_navn("Sara")
```

Vi endrer på objektet som karl referer til
(vi endrer instansvariabelen _navn)




```
ola = Person("Ola", 19, 93, 190)
karl = ola
karl.skriv_ut_hilsen()
```

Vi endrer referansevariabelen
Altså, hvilket objekt karl referer til
(karl peker til ola sitt objekt)



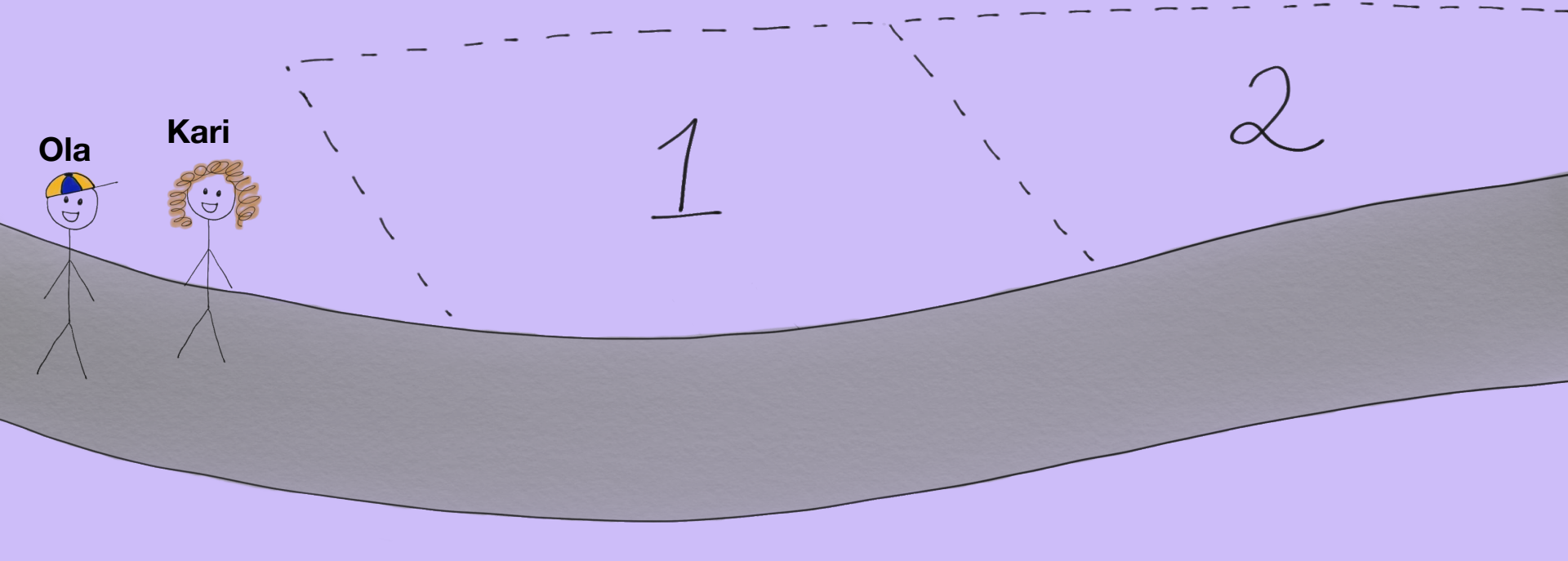
MINNEGATEN

Opprette og bruke objekter

Forskjellen på å endre en referansevariabel vs. å endre objektet den refererer til

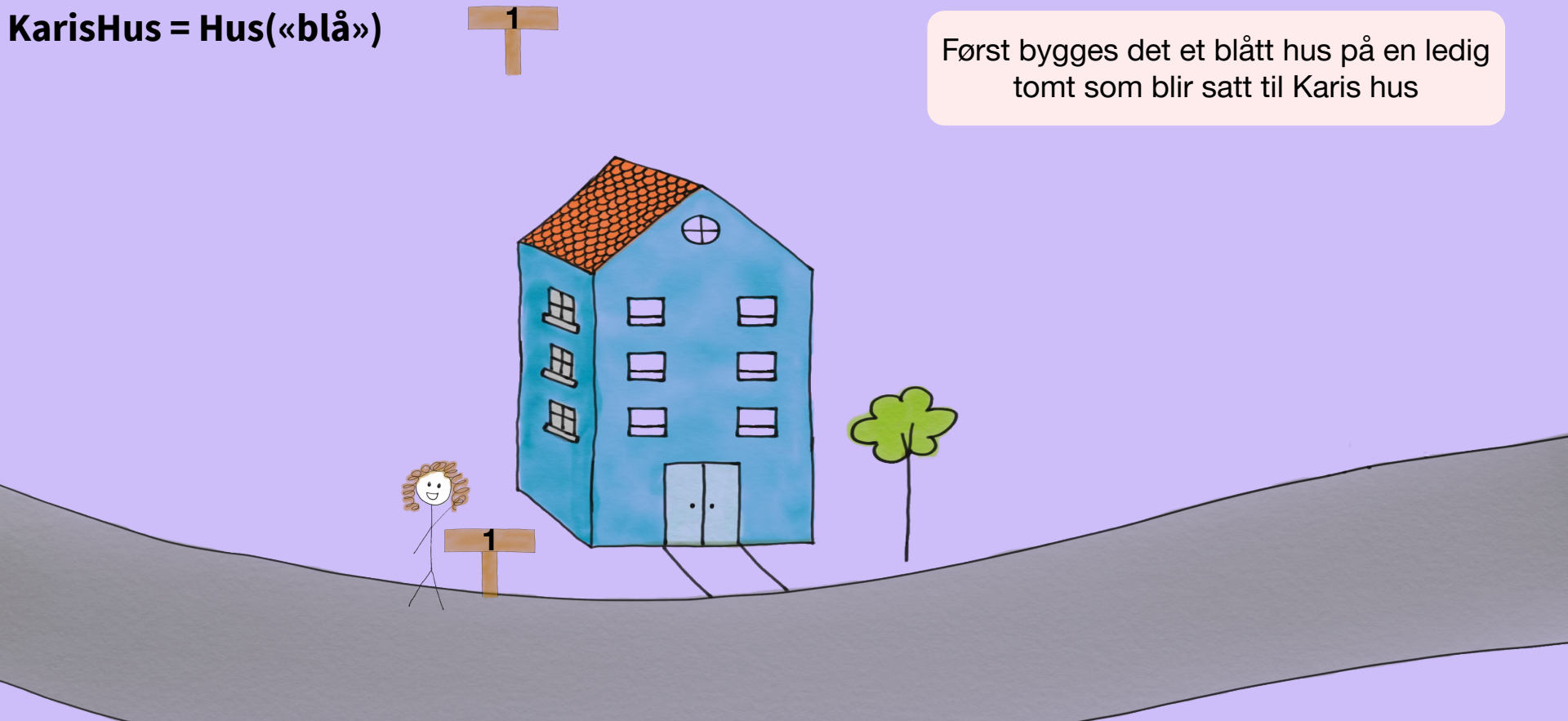
MINNEGATEN

ledige tomter = ledig minne



MINNEGATEN

KarisHus = Hus(«blå»)



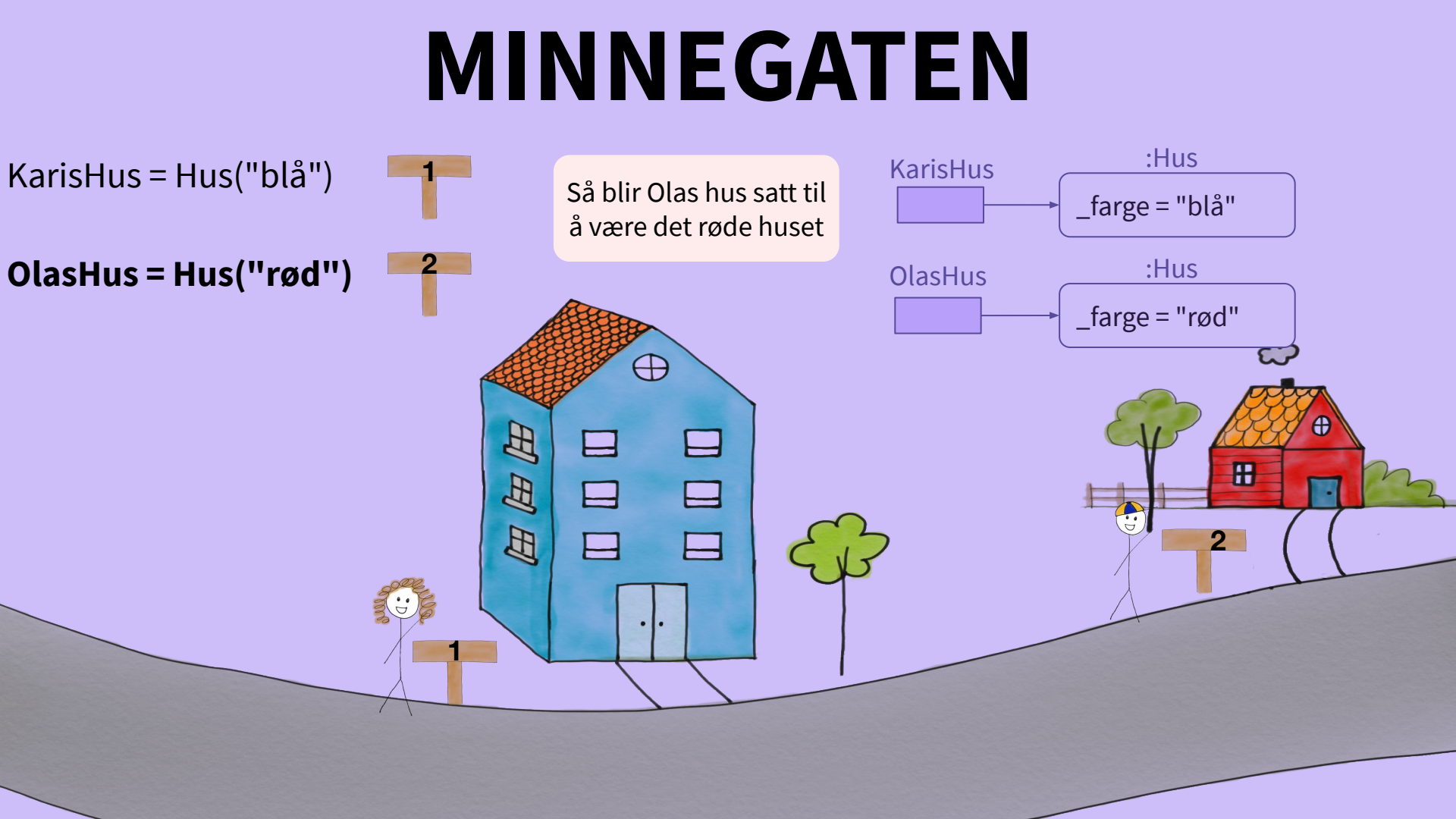
Først bygges det et blått hus på en ledig tomt som blir satt til Karis hus

MINNEGATEN

KarisHus = Hus("blå")

OlasHus = Hus("rød")

Så blir Olas hus satt til å være det røde huset



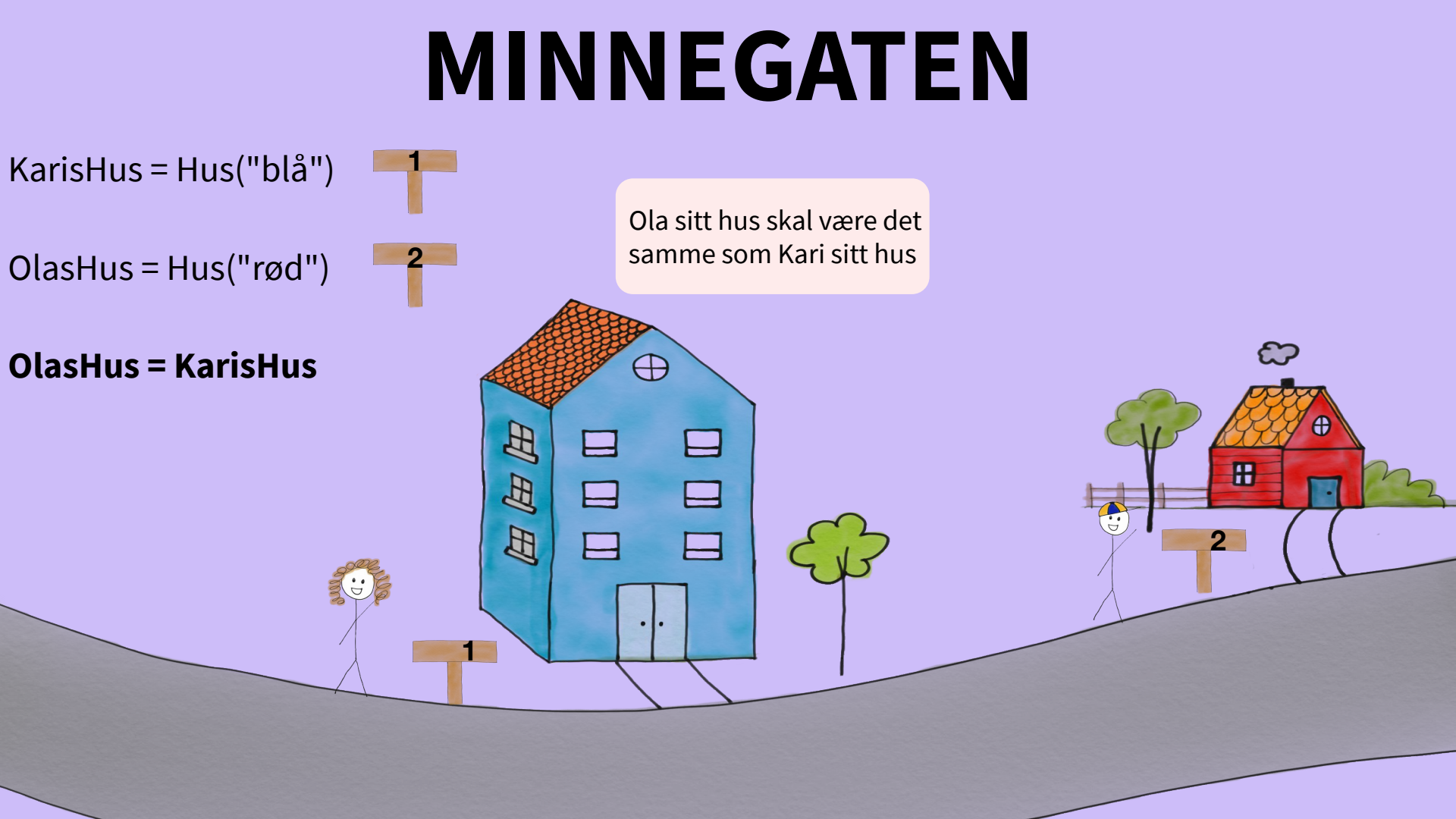
MINNEGATEN

KarisHus = Hus("blå")

OlasHus = Hus("rød")

OlasHus = KarisHus

Ola sitt hus skal være det samme som Kari sitt hus

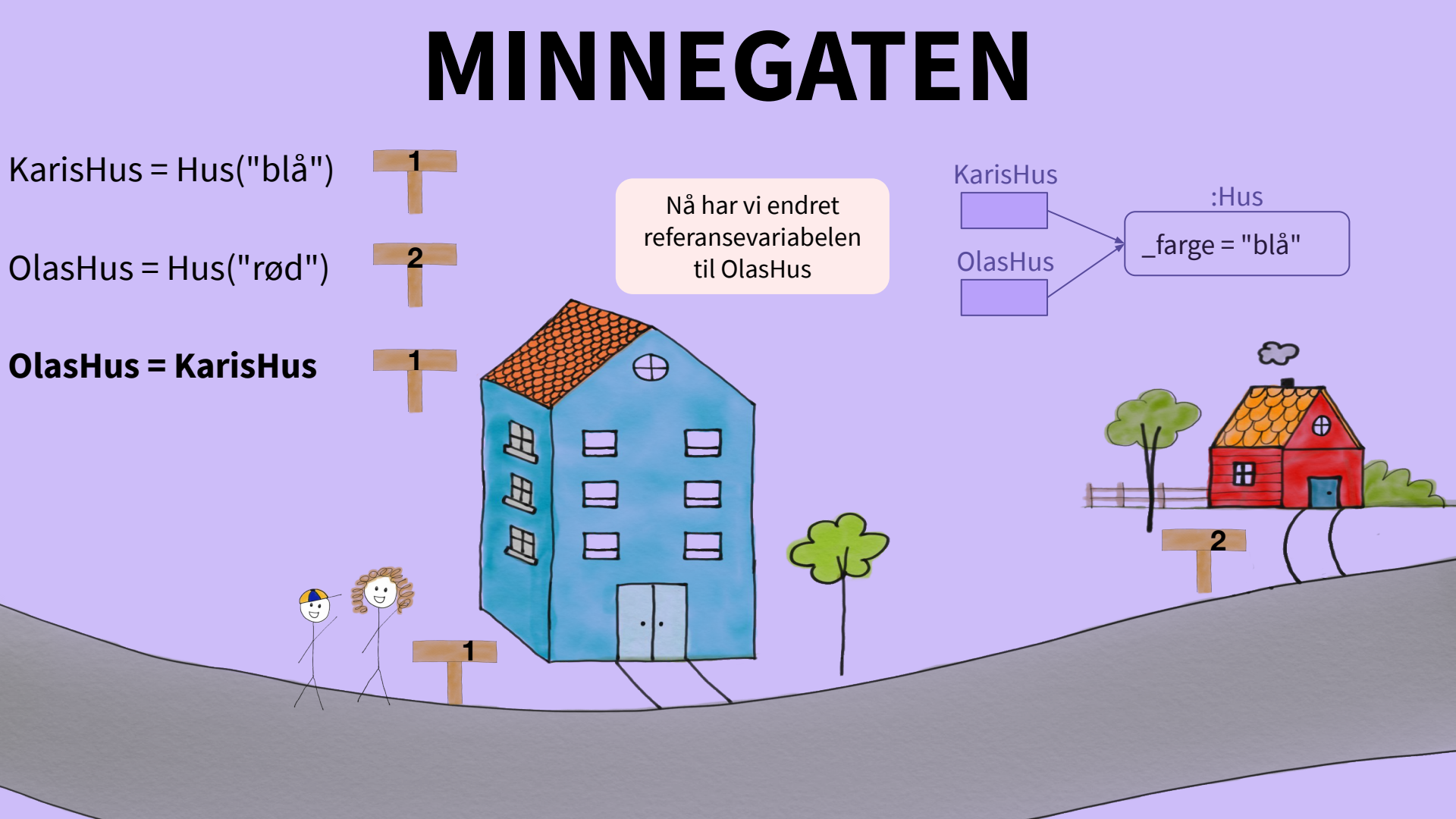


MINNEGATEN

KarisHus = Hus("blå")

OlasHus = Hus("rød")

OlasHus = KarisHus



Nå har vi endret referansevariabelen til OlasHus

MINNEGATEN

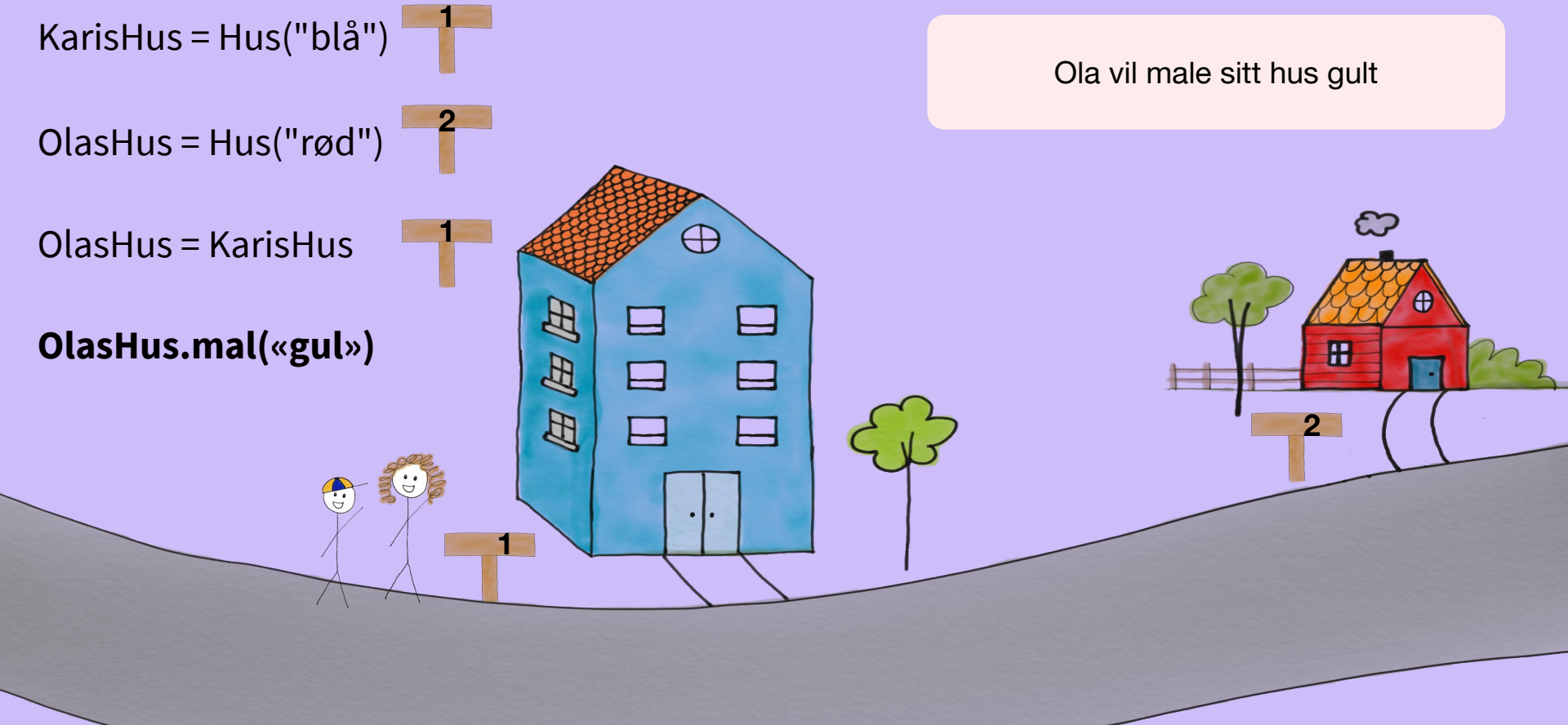
KarisHus = Hus("blå")

OlasHus = Hus("rød")

OlasHus = KarisHus

OlasHus.mal(«gul»)

Ola vil male sitt hus gult



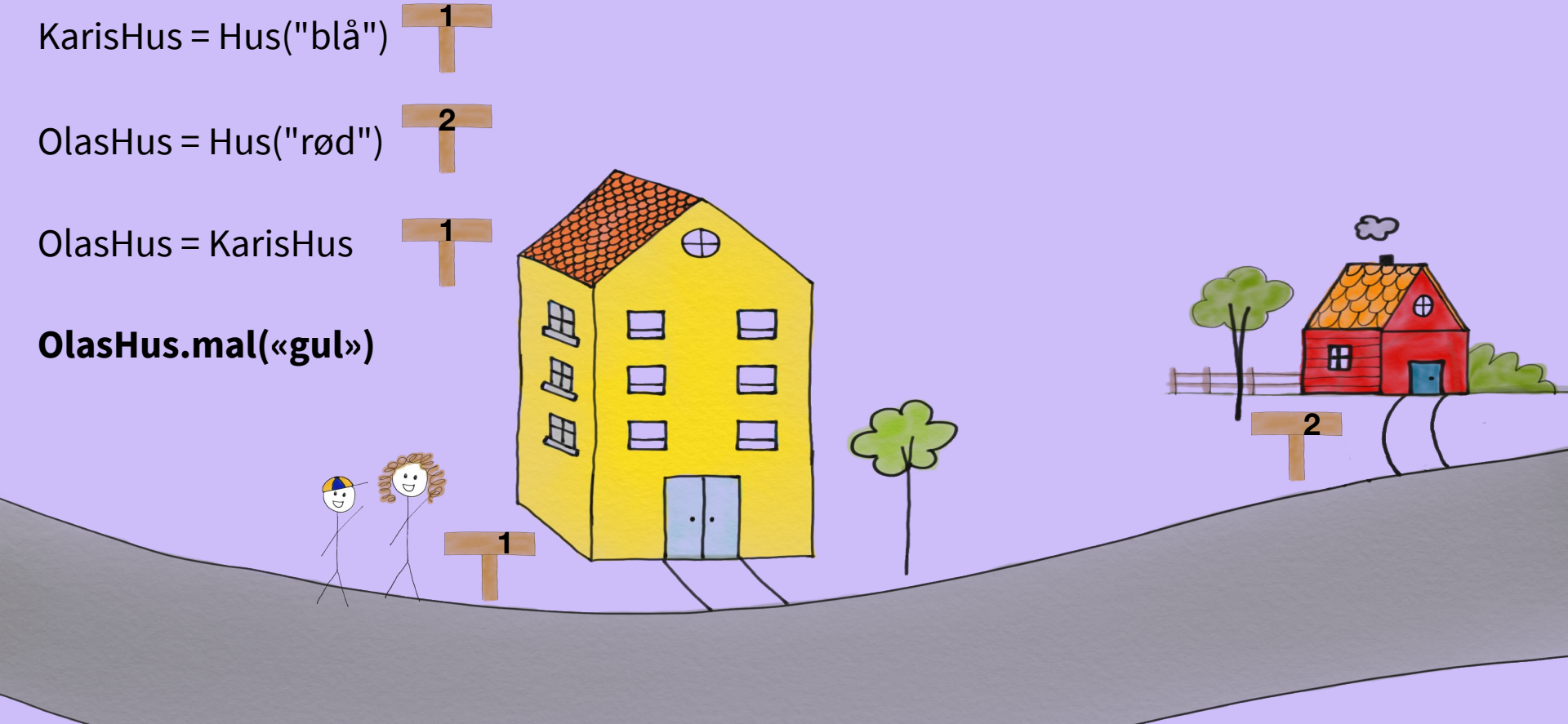
MINNEGATEN

KarisHus = Hus("blå")

OlasHus = Hus("rød")

OlasHus = KarisHus

OlasHus.mal(«gul»)



MINNEGATEN

KarisHus = Hus("blå")

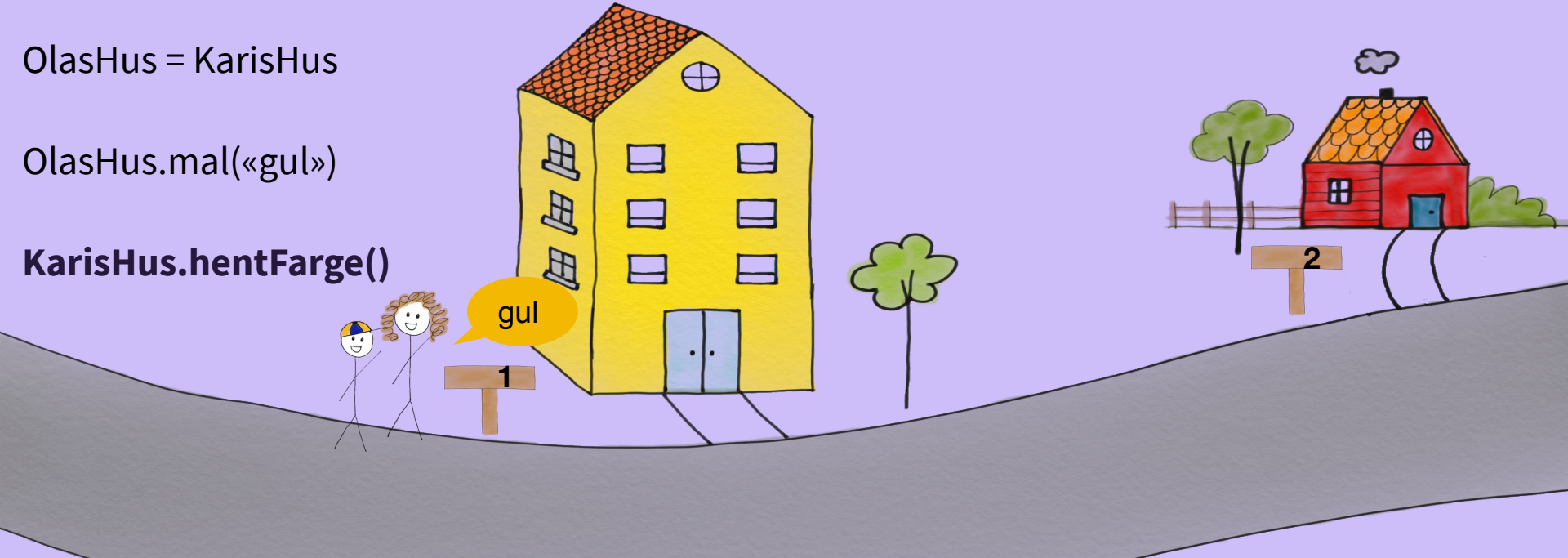
OlasHus = Hus("rød")

OlasHus = KarisHus

OlasHus.mal(«gul»)

KarisHus.hentFarge()

Hvilken farge har Kari sitt hus nå?



MINNEGATEN

KarisHus = Hus("blå")

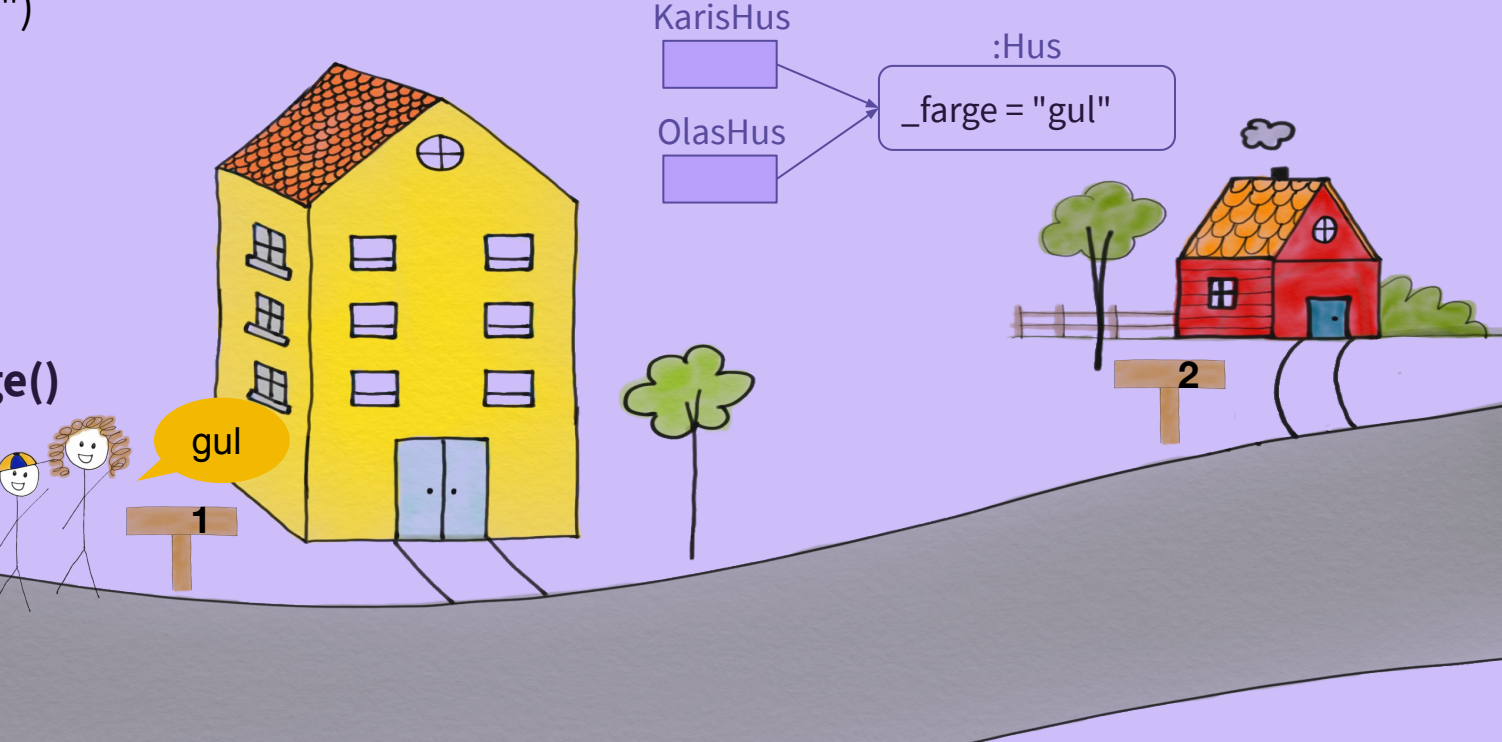
OlasHus = Hus("rød")

OlasHus = KarisHus

OlasHus.mal(«gul»)

KarisHus.hentFarge()

Nå har vi endret objektet som KarisHus og OlasHus referer til



MINNEGATEN

KarisHus = Hus("blå")

OlasHus = Hus("rød")

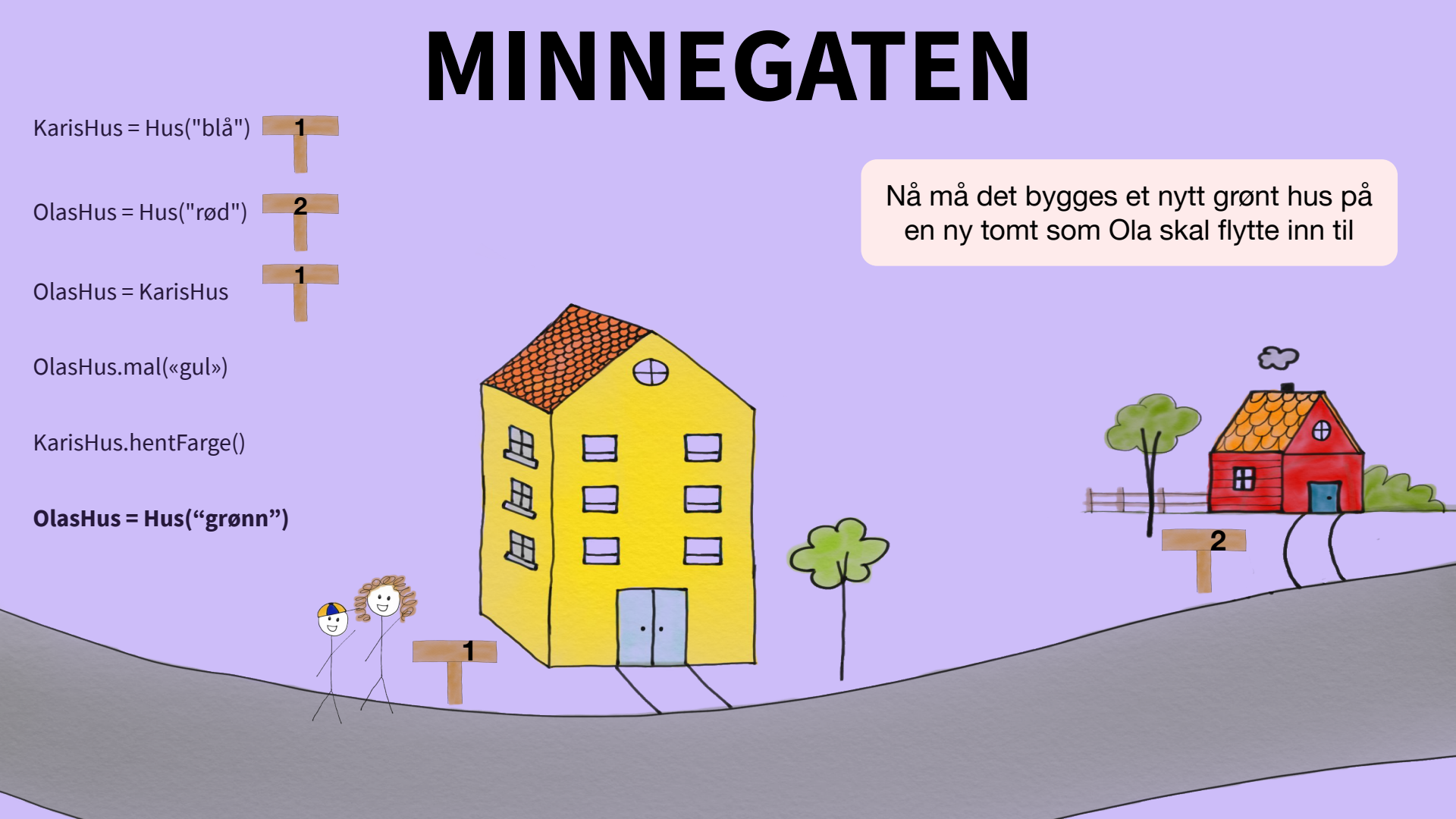
OlasHus = KarisHus

OlasHus.mal(«gul»)

KarisHus.hentFarge()

OlasHus = Hus(“grønn”)

Nå må det bygges et nytt grønt hus på en ny tomt som Ola skal flytte inn til



MINNEGATEN

KarisHus = Hus("blå")

OlasHus = Hus("rød")

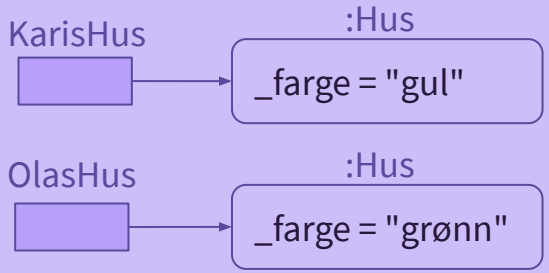
OlasHus = KarisHus

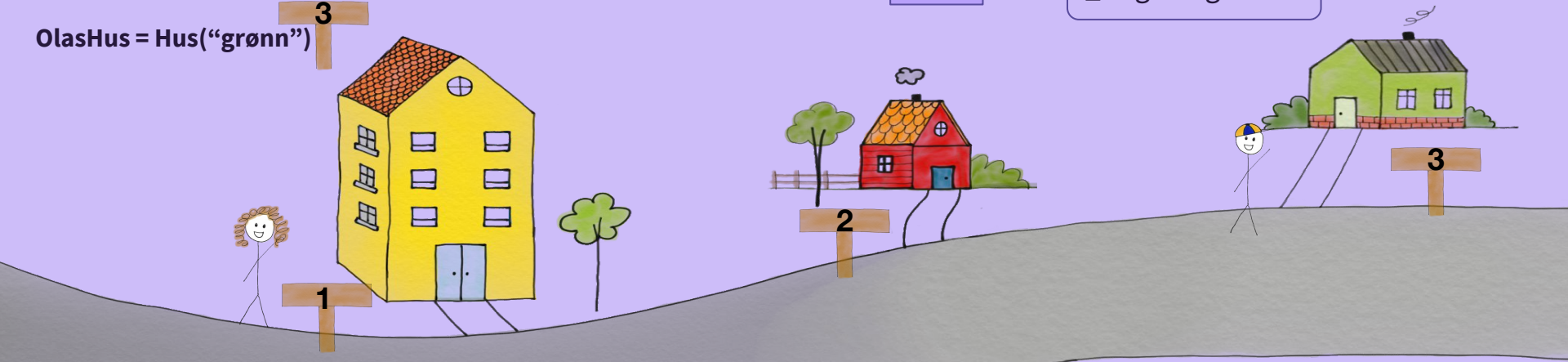
OlasHus.mal(«gul»)

KarisHus.hentFarge()

OlasHus = Hus(“grønn”)

Ola sitt hus er endret til å være det nye huset





MINNEGATEN

KarisHus = Hus("blå")

OlasHus = Hus("rød")

OlasHus = KarisHus

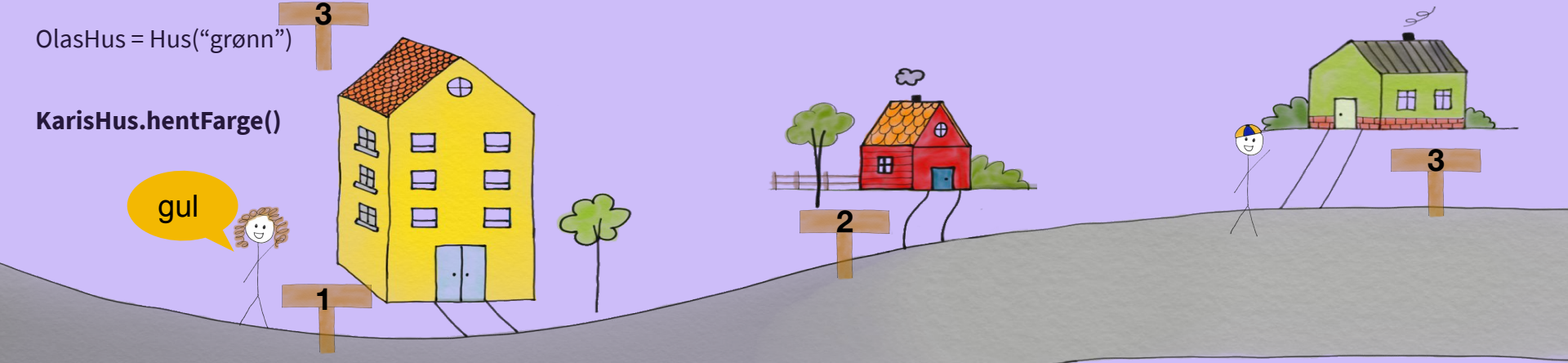
OlasHus.mal(«gul»)

KarisHus.hentFarge()

OlasHus = Hus(“grønn”)

KarisHus.hentFarge()

Hvilken farge har Kari sitt hus nå?



METRORETRO OPPGAVE 1

person.py

person.py

Dette er en forenklet versjon av person.py slik at koden passer på slidet. Du kan finne hele person.py filen på gruppesiden

```python
class Person:
    def __init__(self, navn, alder, vekt, hoyde):
        self._navn = navn
        self._alder = alder
        self._vekt = vekt
        self._hoyde = hoyde
        self._venner = []

    def get_navn(self):
        return self._navn

    def set_navn(self, nytt_navn):
        self._navn = nytt_navn

    def legg_til_venn(self, venn):
        self._venner.append(venn)

    def skriv_ut_venners_navn(self):
        print(f"{self._navn} er venner med:")
        for venn in self._venner:
            print(venn.get_navn())
```

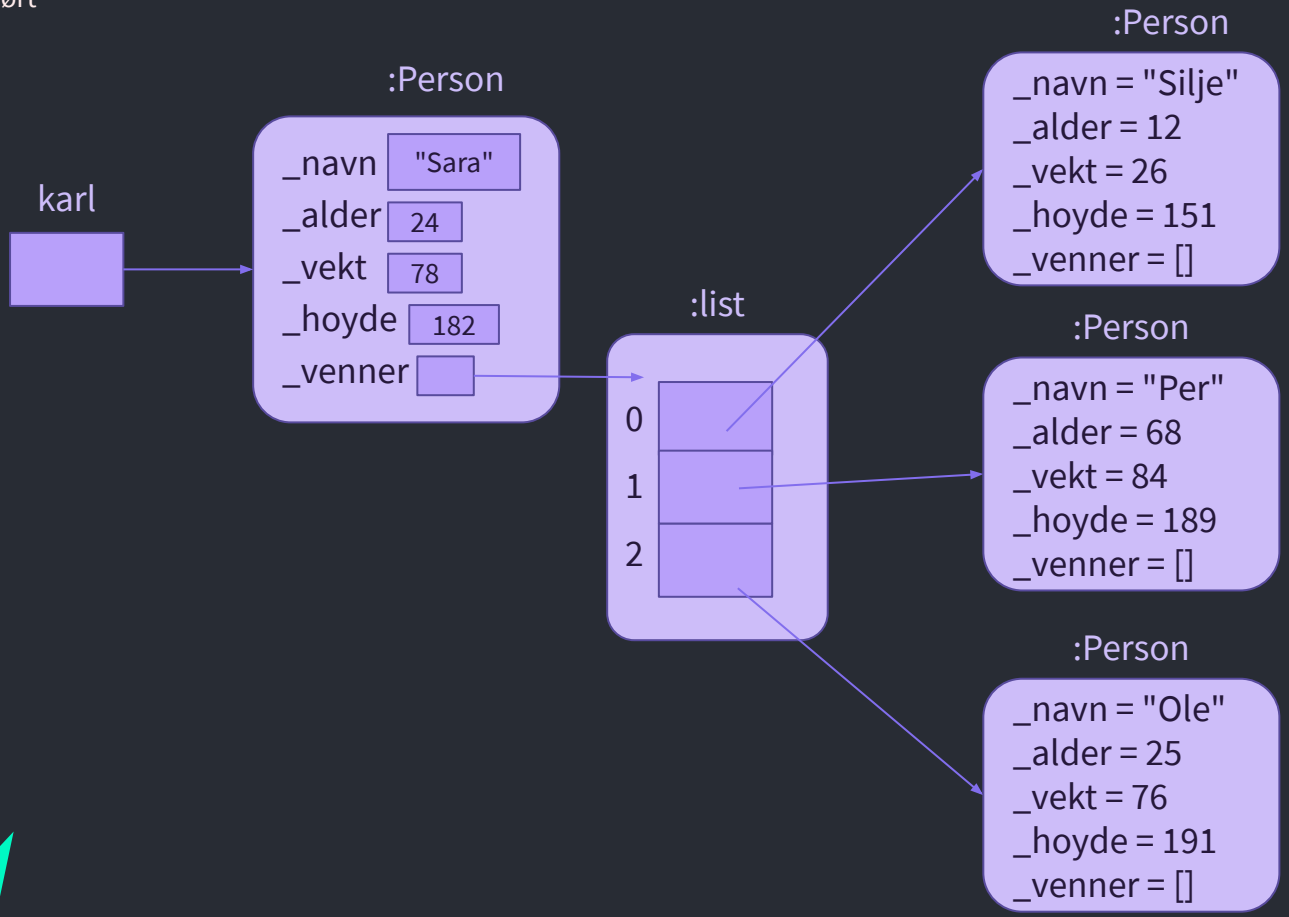
test_person.py

```python
from person import Person

silje = Person("Silje", 12, 46, 151)
per = Person("Per", 68, 84, 189)
ole = Person("Ole", 25, 76, 191)

person_liste = [silje, per, ole]
for person in person_liste:
      person.skriv_ut_hilsen()

# Vi lager objekter basert på Karl og Ola som vi så på tidligere slides
karl = Person("Karl", 24, 78, 182)
ola = Person("Ola", 19, 93, 190)

# Vi endrer objektet karl ved å endre _navn til "Sara"
karl.skriv_ut_hilsen()
karl.set_navn("Sara")
karl.skriv_ut_hilsen()

# Vi endrer objektet karl ved å legge til 3 venner
for person in person_liste:
      karl.legg_til_venn()

karl.skriv_ut_venners_navn()


# Endrer referansevariabelen til karl
# nå referer karl til ola # (karl peker på ola sitt objekt)
karl = ola
karl.skriv_ut_hilsen()
```

METRORETRO
OPPGAVE 2

datastruktur tegning

Datastruktur tegning til hvordan referansevariabelen karl ser ut etter test_person.py har kjørt

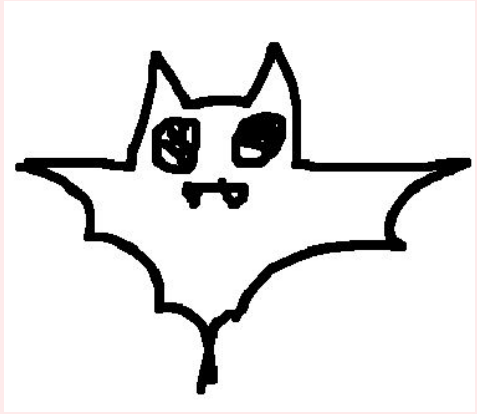


KONTAKT



...

sirisoll@uio.no
@sirisoll på Mattermost



UKENS MESTERVERK: “ikke blind”

- ***A bat redemption story***