IN1000

Uke 8

Dagens plan:

- ‘None’
- ‘Self’ og oppvarmingsoppgave
- Referanser
- Datastruktur

None

```
variabel = None

if variabel is None:
    print("Variabelen er None")
```

Hva er self?

```
class Bankkonto:
    def __init__(self, saldo):
        self._saldo = saldo

    def ta_ut(self, belop):
        self._saldo -= belop
        self._vis_saldo()

    def _vis_saldo(self):
        print(self._saldo)

konto = Bankkonto(1000)
konto.ta_ut(500)
```

Oppgave!

- Skriv en klasse ‘Sirkel’ i filen sirkel.py. En sirkel har en radius. Opprett en konstruktør som oppretter radius (sendt inn som parameter).
- Lag så følgende metoder:
  1. En som returnerer diameteren til sirkelen (radius * 2)
  2. En som returnerer omkretsen til sirkelen (diameter * pi)
  3. En som returnerer arealet til sirkelen (radius^2 * pi)

     *for å få eksponent skriver man <base> ** <eksponent>, f.eks.*

     *er 2^3 = 2 ** 3*
- Opprett så en test-fil der du lager to sirkler med ulik radius. Skriv ut den ene sirkelens omkrets og areal, og den andre sirkelens diameter.

Løsningsforslag:

```python
class Sirkel:
    def __init__(self, radius):
        self._radius = radius

    def diameter(self):
        return self._radius * 2

    def omkrets(self):
        return self.diameter() * 3.14

    def areal(self):
        return (self._radius ** 2) * 3.14
```

Referanser

- Vi har lært at variabler er noe som **holder** på en verdi:

```
string = "Hei, hei"
tall = 5
float = 3.14
boolean = False
```

- Men når vi snakker om klasser og objekter heter det at en **referanse peker på objekt**

```
tesla = Bil("Tesla", 100)
```

Referanser

- Se for dere variabler som små bokser som holder på verdier, mens referanser er små bokser som inneholder adressen til objektet

- Variabler kan inneholde strings, integers, floats eller booleans (**immutable**)
- Referanser peker til f.eks. lister, mengder, ordbøker, klasser man lager selv (**mutable**)

Men hva blir konsekvensene av dette?

```
tesla = Bil("Tesla", 100)

tesla_kopi = tesla

tesla_kopi.endre_fart(250)

print(tesla.hent_fart)
```

tesla_kopi



tesla

Men hva blir konsekvensene av dette?

```
tall = 5

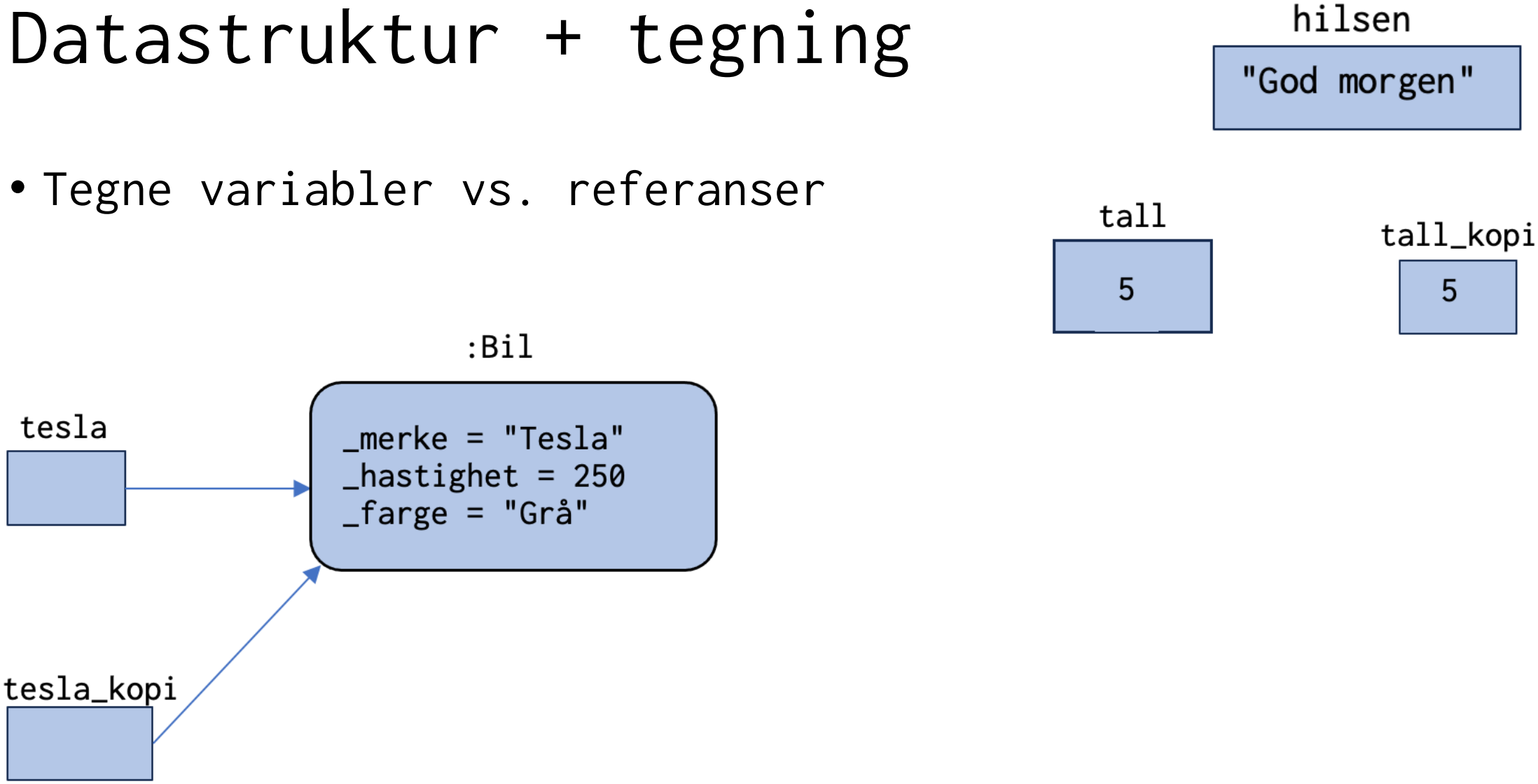
tall_kopi = tall

tall_kopi += 3

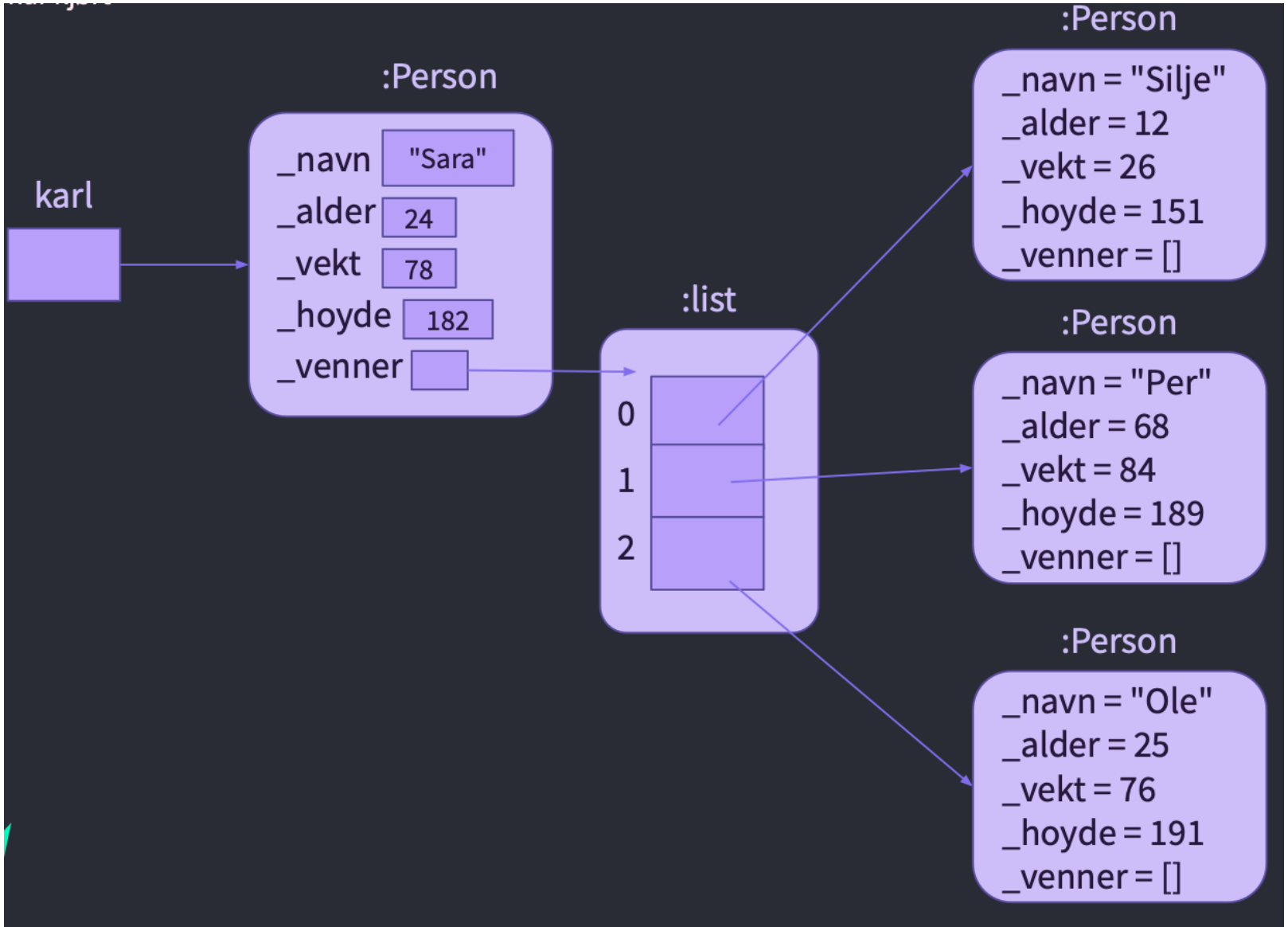
print(tall)
```

Datastruktur + tegning

- Tegne variabler vs. referanser



Enda et eksempel



Oppgave!

Oppgave 2
Lag en klasse Blomst. En blomst har en art, en høyde, en instansvariabel som sier om blomsten har det bra eller ikke (denne verdien er en boolean), og en verdi som forteller hvor lenge siden blomsten ble vannet sist. Konstruktøren setter alle disse verdiene, når en blomst blir opprettet vil blomsten alltid ha det bra og det vil være 0 dager siden den ble vannet sist.

En blomst har en metode hent_status som returnerer verdien som forteller om blomsten har det bra eller ikke.

I tillegg har den metoden skriv_ut_info, som skriver ut en info streng om blomsten

Videre har en blomst en metode neste_dag, som øker antall dager siden den ble vannet med en. Hvis det er mer enn tre dager siden blomsten ble vannet sist vil statusen til blomsten være at den har det dårlig. Hvis statusen til blomsten er at den har det bra vokser blomsten 1 cm.

Den siste metoden en blomst har er metoden vann. Hvis det er mindre enn 3 dager siden blomsten ble vannet vil statusen til blomsten bli dårlig (over vanning), ellers vil statusen være bra.

Skriv denne klassen + lag et testprogram for å teste klassen din

```python
class Blomst:
    def __init__(self, art, hoyde):
        self._art = art
        self._hoyde = hoyde
        self._har_det_bra = True
        self._vannet_sist = 0

    def hent_status(self):
        return self._har_det_bra

    def skriv_ut_info(self):
        print(f"Arten er {self._art}, og hoyden er {self._hoyde}")

    def neste_dag(self):
        self._vannet_sist += 1
        if self._vannet_sist > 3:
            self._har_det_bra = False
        else:
            self._hoyde += 1

    def vann(self):
        if self._vannet_sist < 3:
            self._har_det_bra = False
        else:
            self._har_det_bra = True
```