

Velkommen til attende gruppetime!

Gruppe 8

Uke 9: Magiske metoder

1. Logg inn i Devilry.
2. Svar på følgende spørsmål i kommentarfeltet i Devilry (**skal** besvares):
    a. Hvordan synes du innleveringen var? Hva var enkelt og hva var vanskelig?
    b. Hvor lang tid (ca) brukte du på innleveringen?
    c. Var det noen oppgaver du ikke fikk til? Hvis ja:
        i. Hvilke(n) oppgave er det som ikke fungerer i innleveringen?
        ii. Hvorfor tror du at oppgaven ikke fungerer?
        iii. Hva ville du gjort for å få oppgaven til å fungere hvis du hadde mer tid?
    d. Hva vil du ha tilbakemelding på?

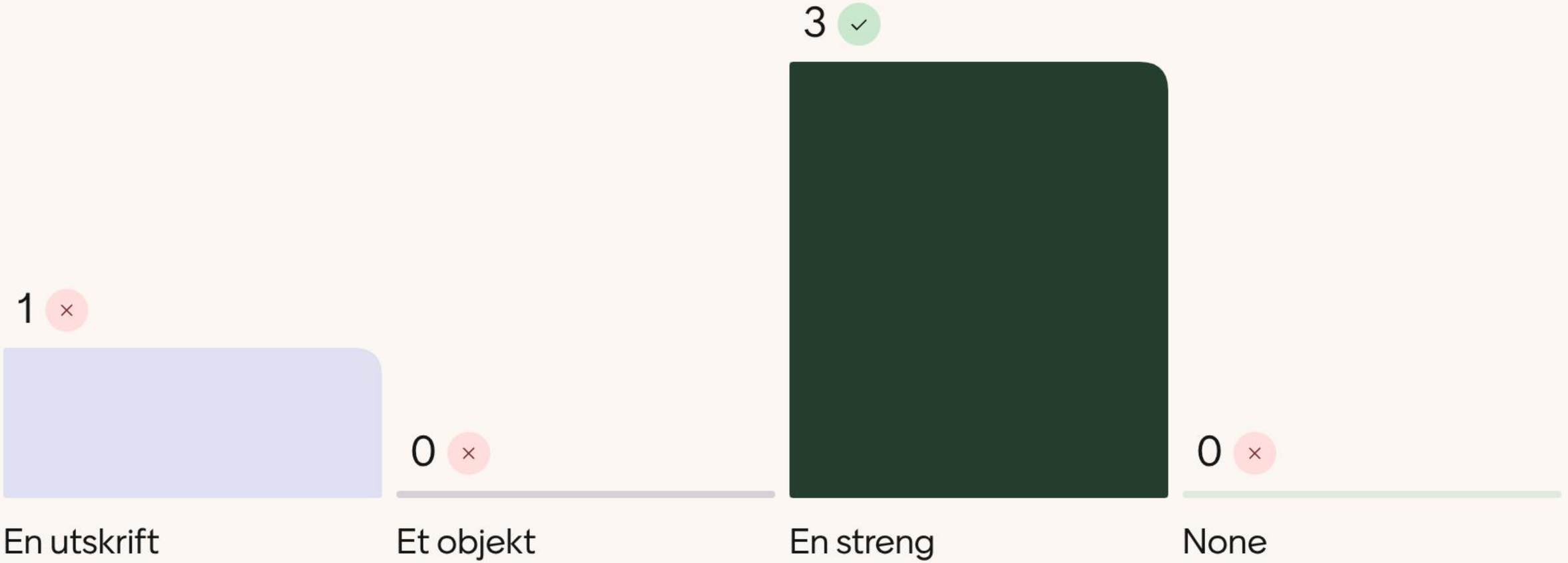
0

Quiz!



0

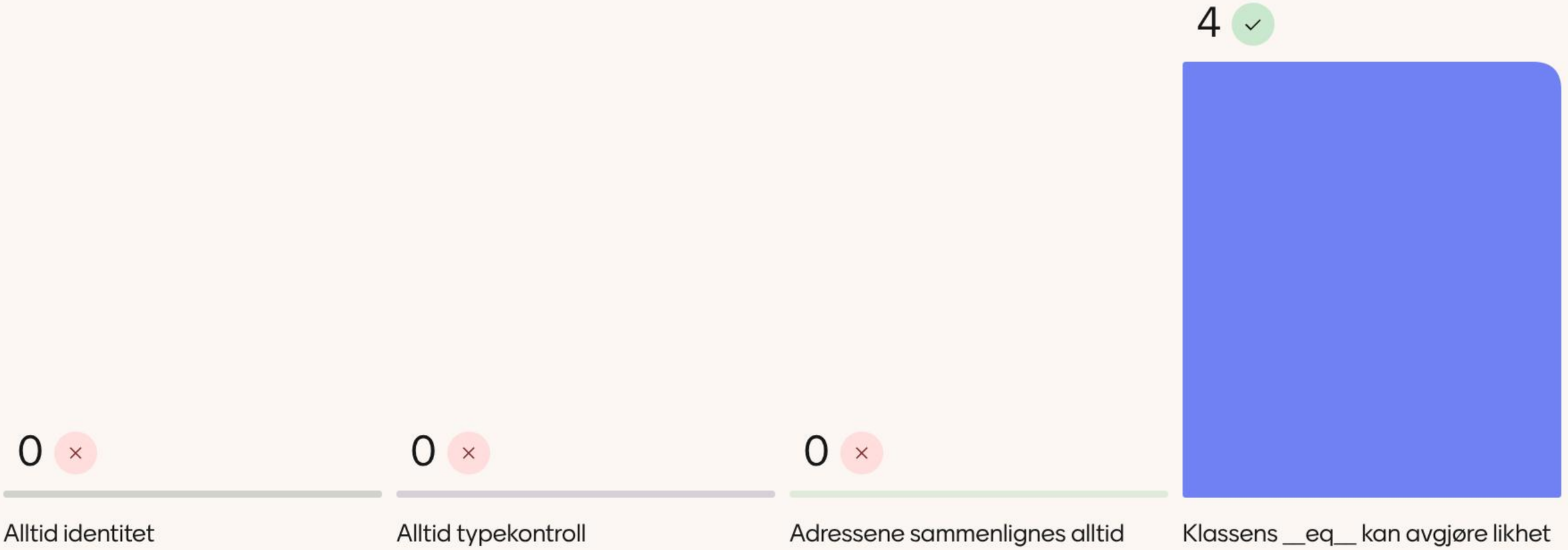
Hva skal __str__ returnere?



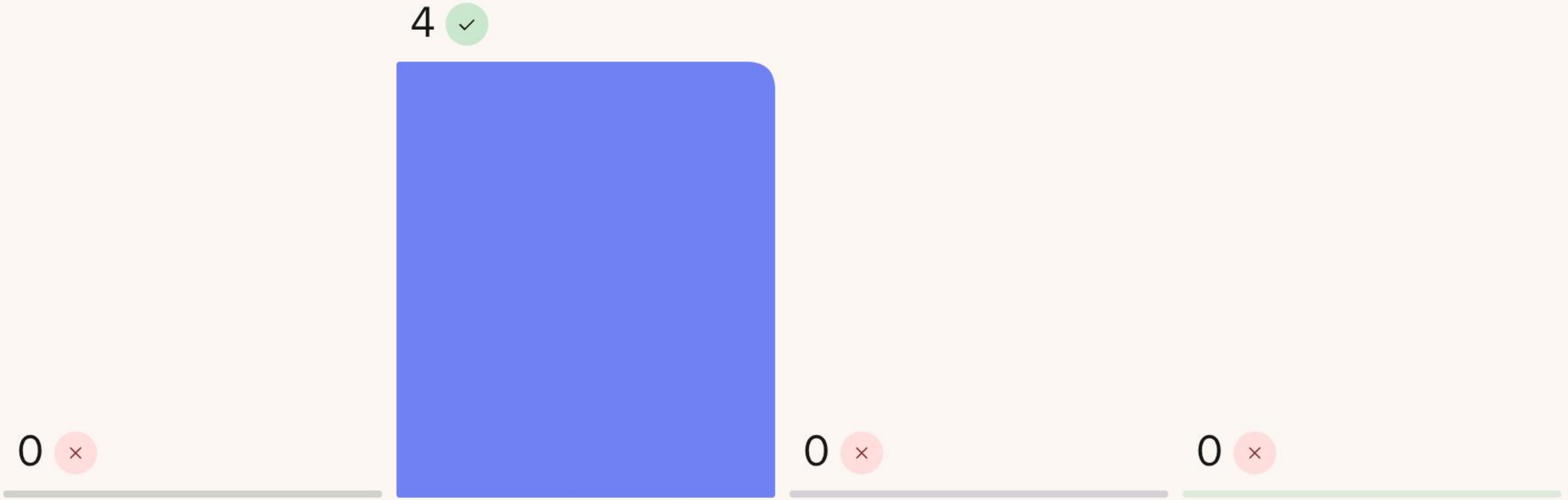
Hva undersøker "a is b"?



Hva skjer når a == b?



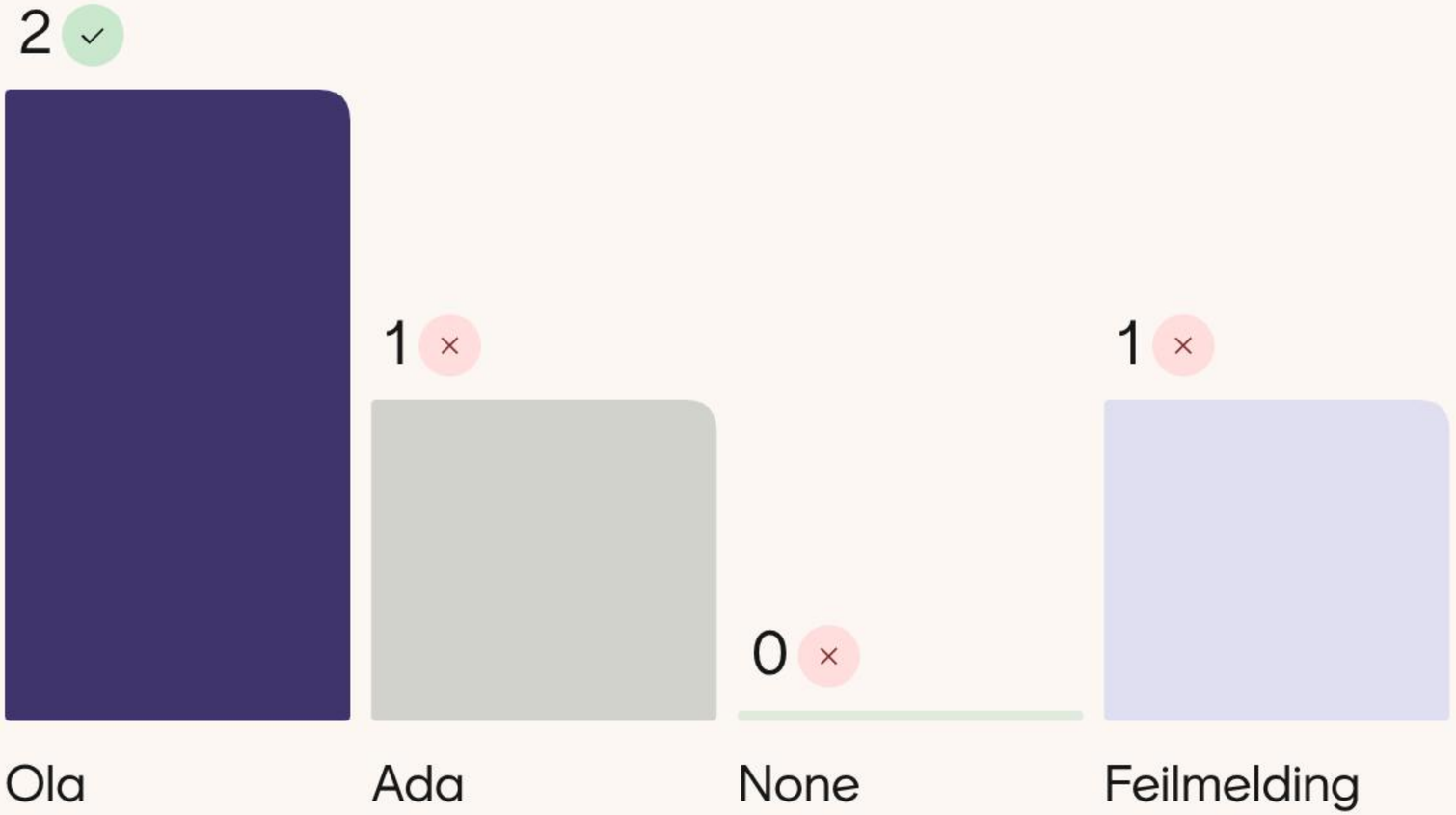
Hva er en fordel med å bruke __str__ i en klasse?



Hva er forskjellen på __str__ og __repr__?



Hva skrives ut?




```python
class Student:
    def __init__(self, navn):
        self._navn = navn

a = Student("Ada")
b = a
b._navn = "Ola"
print(a._navn)
```

Hva er problemet med denne metoden?

```
1 def __str__(self):
2     print(self.navn)
```



Den bruker feil attributt

__str__ kan ikke bruke self

Metoden må hete str

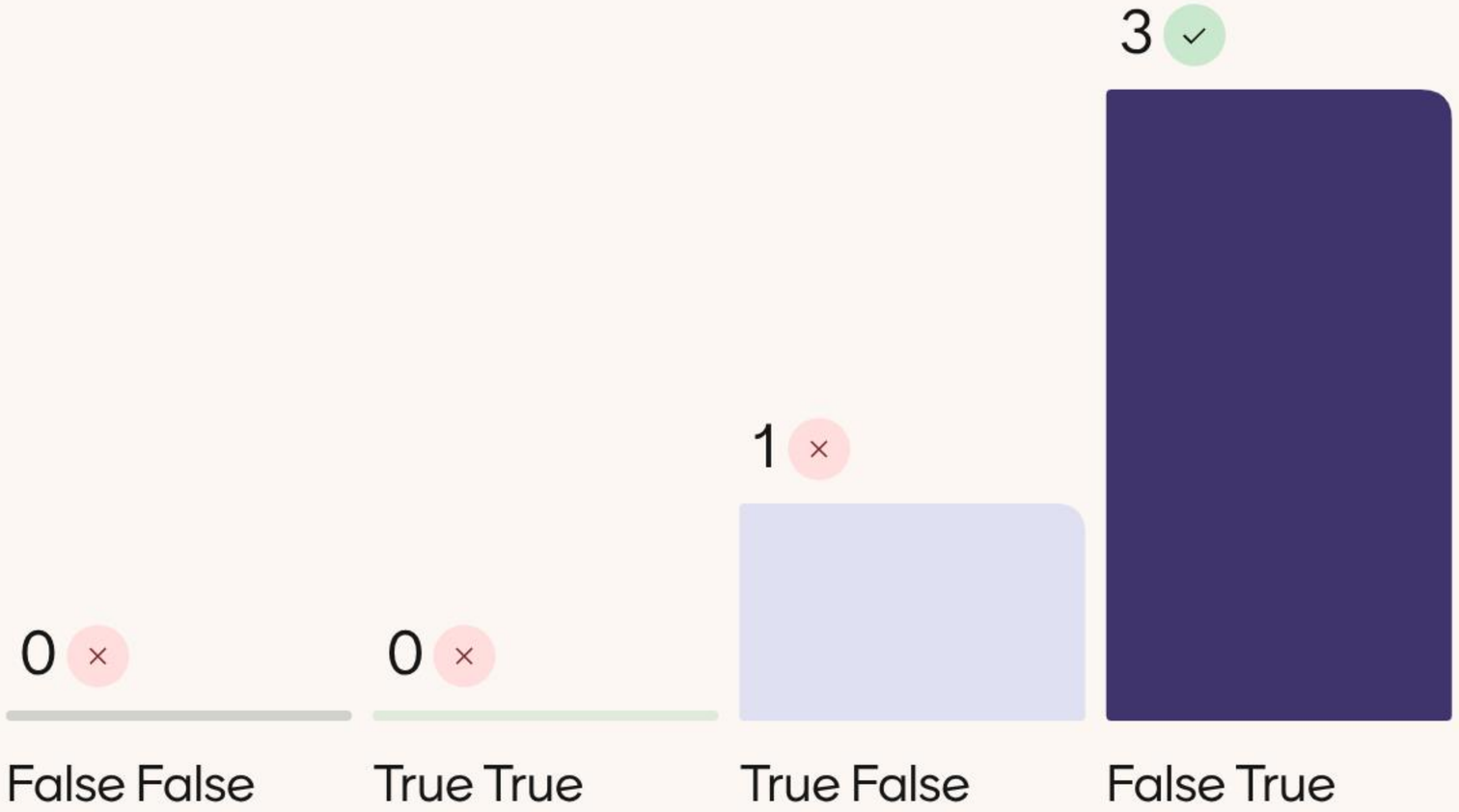
Den skriver ut tekst, men returnerer ikke en streng

Hva skrives ut?

```python
class Student:
    def __init__(self, nummer):
        self._nummer = nummer

    def __eq__(self, other):
        return self._nummer == other._nummer

a = Student(123)
b = Student(123)
print(a is b, a == b)
```



Hva er tilfelle når metoden returnerer None?

1 ×

Metoden har funnet den første studenten

3 ✓

Ingen student ble funnet

0 ×

Listen er automatisk slettet

1 ×

Programmet avsluttes alltid

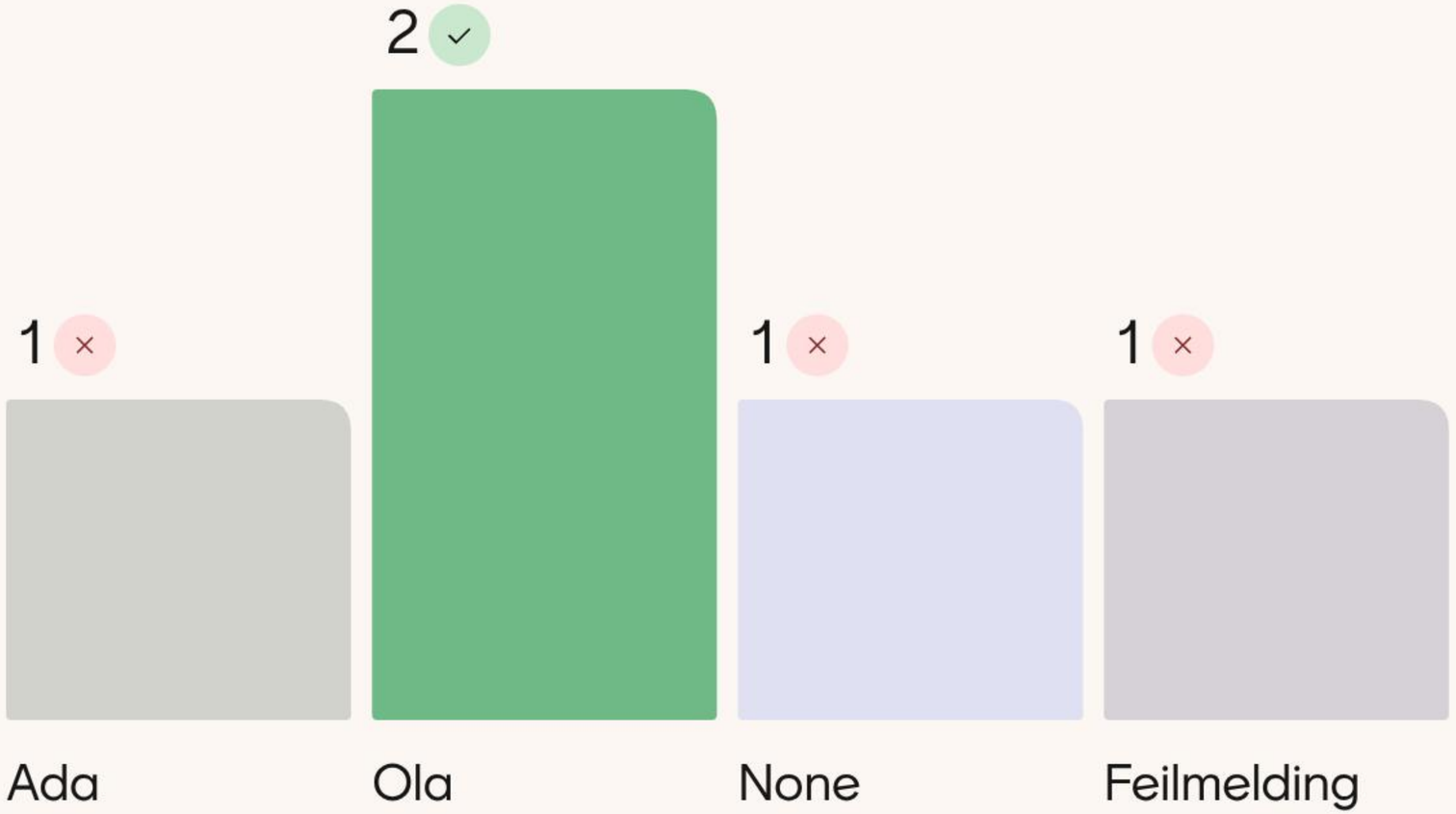
```python
def finn_student(self, nummer):
    for student in self.studenter:
        if student._nummer == nummer:
            return student
    return None
```

Hva skrives ut?

```python
class Navn:
    def __init__(self, fornavn):
        self._fornavn = fornavn

class Person:
    def __init__(self, navn):
        self._navn = navn

navn = Navn("Ada")
person = Person(navn)
navn._fornavn = "Ola"
print(person._navn._fornavn)
```



Koden fungerer, men hva ville dere forbedret?

- En student har skrevet følgende program for å administrere emner
- Finn minst **to** ting som kunne vært gjort bedre.
- Forklar hvorfor hver endring kan være nyttig.
- Tenk over hva dere har lært hittil i IN1000

```python
class Emne:
    def __init__(self, kode, studiepoeng):
        self.kode = kode
        self.studiepoeng = studiepoeng

    def skriv_ut(self):
        print(self.kode + ": " +
              str(self.studiepoeng) +
              " studiepoeng")


def skriv_ut_emner(emner):
    for emne in emner:
        emne.skriv_ut()


emner = [
    Emne("IN1000", 10),
    Emne("IN1010", 10),
    Emne("IN2000", 20)
]

skriv_ut_emner(emner)
```

0

(Forslag) Koden fungerer, men hva ville dere forbedret?

- `kode` og `studiepoeng` kan hete `_kode` og `_studiepoeng`
- Klassen kan implementere `__str__` i stedet for å bruke `skriv_ut()`.
- Håndterer ikke tomme verdier

```python
class Emne:
    def __init__(self, kode, studiepoeng):
        self.kode = kode
        self.studiepoeng = studiepoeng

    def skriv_ut(self):
        print(self.kode + ": " +
              str(self.studiepoeng) +
              " studiepoeng")


def skriv_ut_emner(emner):
    for emne in emner:
        emne.skriv_ut()


emner = [
    Emne("IN1000", 10),
    Emne("IN1010", 10),
    Emne("IN2000", 20)
]

skriv_ut_emner(emner)
```

0

Tenk dere at dere lager et program med klassene Gruppe og Student. En gruppe inneholder flere studenter, og dere vil lage en funksjon som finner alle studentene til et studieprogram. Bør metoden ligge i Gruppe, i Student eller som en egen funksjon utenfor klassene? Hvorfor?

- Student kan ha ansvar for informasjon om en student
- Gruppe kan ha ansvar for å håndtere studentene i gruppen.

0

Hvordan bør vi lagre objekter når vi trenger å finne dem igjen? Ville dere brukt en liste eller en ordbok for å lagre emner av klassen "Emne"? Hvorfor?

- En liste er nyttig når det er begrenset lengde på den (duplikater? hmm)
- En ordbok med emnekoden som nøkkel gjør det enkelt å slå opp et bestemt emne.

0

Oppgaver

- Hvis du synes oppgaven er enkel, tenk på mulige utvidelser!
- Om du ønsker kan du jobbe med Trix, oblig eller oppgaver fra tidligere gruppetimer dersom du mener at det gir deg mer nytte
- **Hvis du synes ALT er lett, prøv deg på et større prosjekt som bruker det meste av IN1000-pensum, tips:**
  - Lag det med noe du faktisk er interessert i som fra et spesifikk franchise
  - Øv deg på å lage det utvidbart, altså ryddig og oversiktlig slik at du kan bygge videre på det neste semester, neste år og senere i livet

0

Oppgave 1

Lag en klasse Emne med attributtene kode og studiepoeng. Lag deretter en liste med minst tre emneobjekter.

Skriv en funksjon som finner og returnerer alle emner med minst 10 studiepoeng.

Krav: Funksjonen skal returnere en liste med objekter, ikke skrive dem ut.

Ekstra: Hva returneres dersom ingen emner oppfyller kravet?

0

Oppgave 2

Lag en klasse Student med attributtene navn og studentnummer. To studentobjekter skal regnes som like dersom de har samme studentnummer.

Implementer __eq__ og test at sammenligningen fungerer med ==.

0

Oppgave 3

Lag en klasse Vare med attributtene navn og pris. Lag en ordbok der varenavnet er nøkkel og vareobjektet er verdi.

Skriv en funksjon som beregner totalprisen for alle varene i ordboken.

Krav: Test funksjonen med minst tre varer.

Ekstra: *Lag en funksjon som returnerer en ny ordbok med bare varene som koster mindre enn 100 kroner.*

0

Ferdig?

- Tenk tilbake på ukene til nå, hva er noe du sliter med i IN1000?
  - Jobb med det
  - Bruk filter på Trix

0