Velkommen til syttende gruppetime!

Gruppe 8

Uke 9: Magiske metoder



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

    def __str__(self):
        print(self._kode + ": " + self._studiepoeng)

emne = Emne("IN1000", 10)
tekst = str(emne)
print(tekst + " høst")
```

Finn tre feil som hindrer koden i å kjøre som den skal.

Fasit

- **self._studiepoeng** er et heltall. Strengkonkatenering med + gir TypeError fordi heltallet ikke automatisk konverteres til tekst. Bruk str(self._studiepoeng)
- **str** skriver ut i stedet for å returnere. Metoden må returnere en streng
- Fordi metoden ikke returnerer en streng, ville **str(emne)** gitt TypeError selv etter at den første typen feilen var rettet, dersom metoden fortsatt bare skrev ut. En **str**-metode må returnere en streng

```
1  class Emne:
2      def __init__(self, kode, studiepoeng):
3          self._kode = kode
4          self._studiepoeng = studiepoeng
5
6      def __str__(self):
7          print(self._kode + ": " + self._studiepoeng)
8
9  emne = Emne("IN1000", 10)
10 tekst = str(emne)
11 print(tekst + " høst")
```

Quiz!



Hva er hovedforskjellen på is og ==?



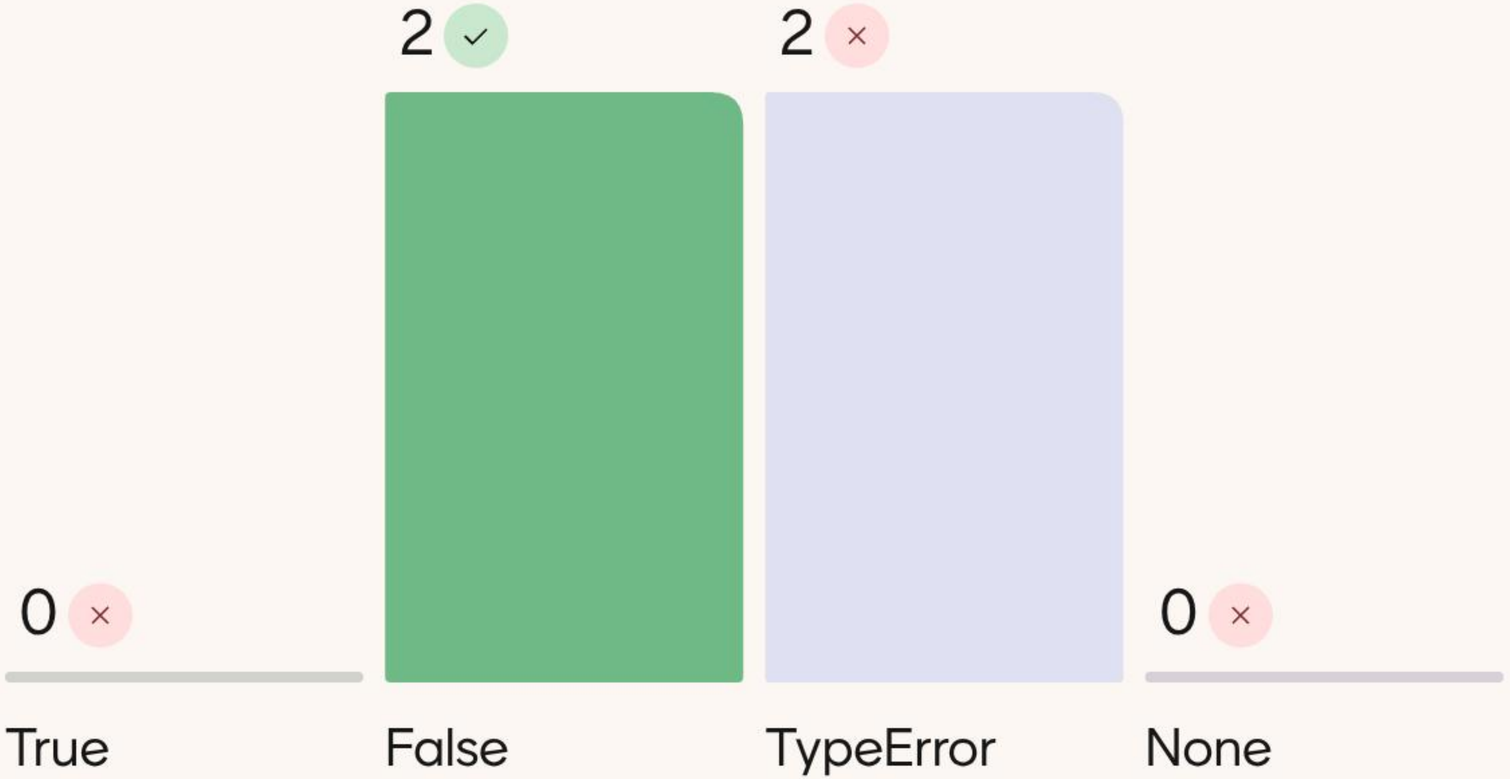
is for innhold, == for type

De gjør det samme

is for tall, == for strenger

is for type, == for innhold

To objekter opprettes med samme argumenter. Klassen har ingen egen __eq__. Hva er riktig?




```python
a = Rektangel(4, 5)
b = Rektangel(4, 5)
print(a == b)
```

Hva skrives ut?

```python
class Sang:
    def __init__(self, tittel):
        self._tittel = tittel

    def __str__(self):
        return "Sang: " + self._tittel

sang = Sang("Natt")
print(sang)
```

4 ✓

Sang: Natt

0 ×

<Sang object at ...>

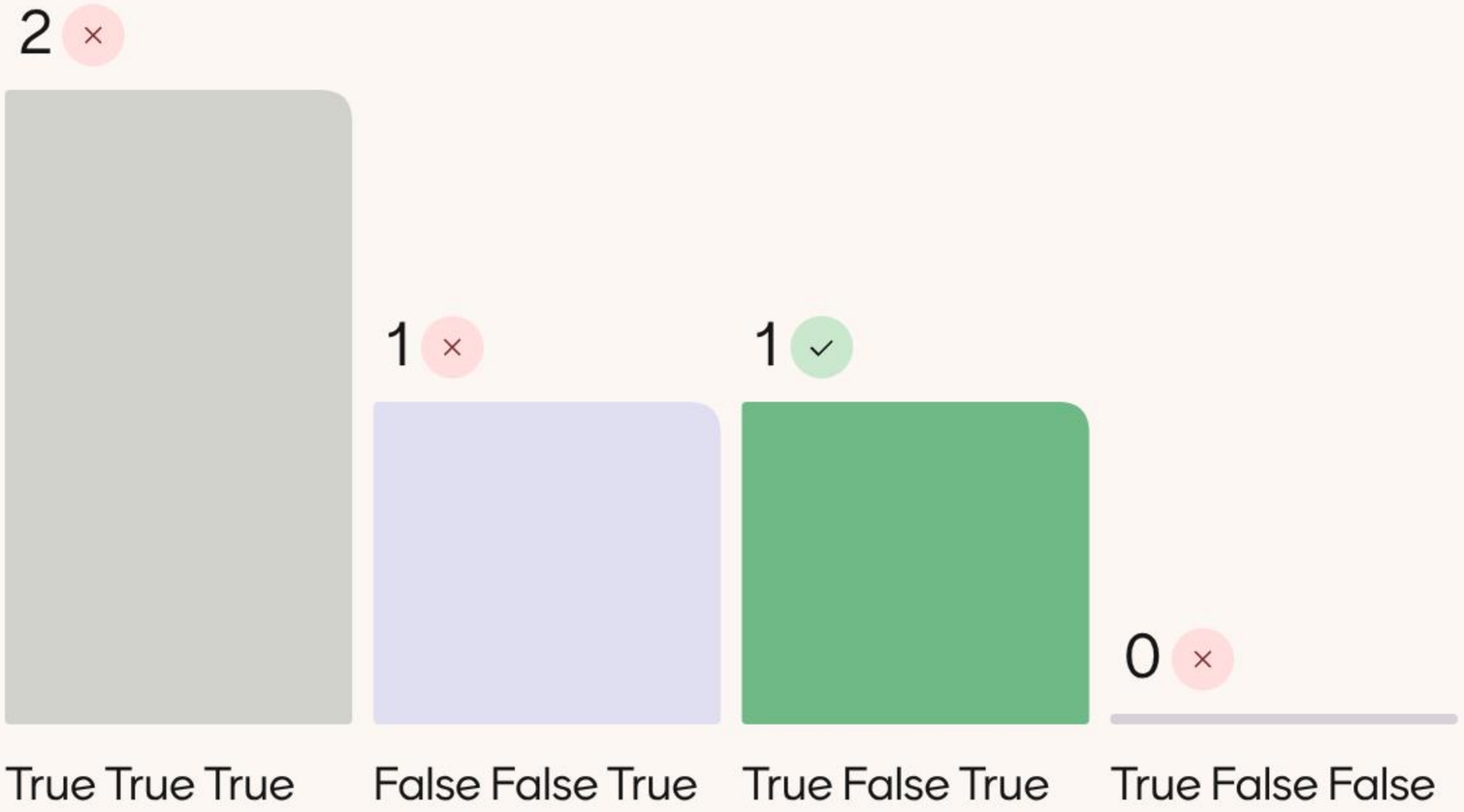
0 ×

Natt

0 ×

Ingenting

Hva skrives ut?



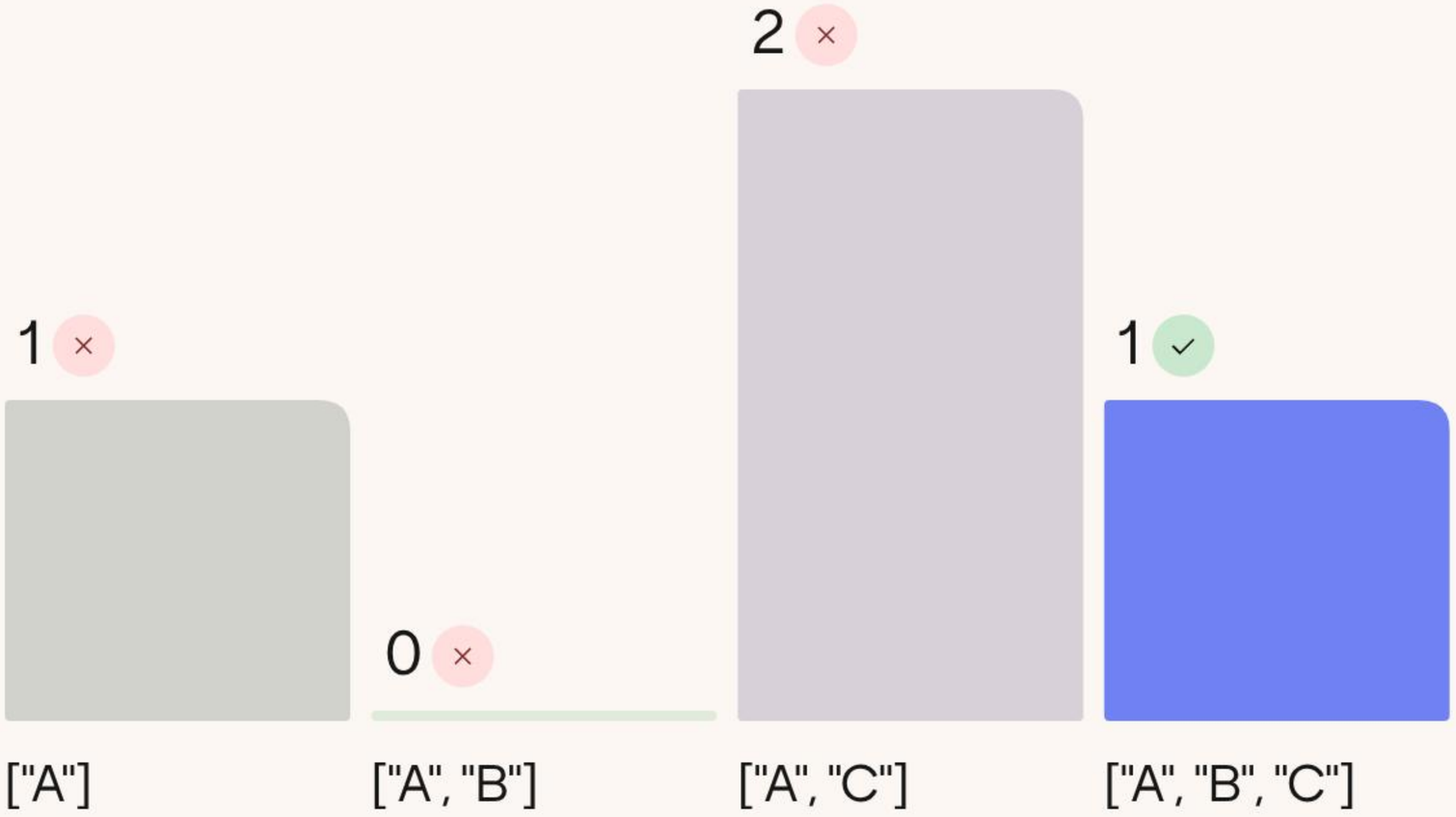

```python
class Vare:
    def __init__(self, id):
        self._id = id

    def __eq__(self, annen):
        return self._id == annen._id

v1 = Vare(7)
v2 = Vare(7)
v3 = v1

print(v1 == v2)
print(v1 is v2)
print(v1 is v3)
```

Hva skrives ut?




```python
class Samling:
    def __init__(self):
        self._varer = []

    def legg_til(self, vare):
        self._varer.append(vare)

    def hent_varer(self):
        return self._varer


s1 = Samling()
s1.legg_til("A")
s2 = s1
varer = s2.hent_varer()
varer.append("B")

s1.legg_til("C")
print(s2.hent_varer())
```

Hvilken av følgende endringer ville gitt en uavhengig liste?

4 ✓

varer = s2.hent_varer().copy()

0 ×

varer = s2.hent_varer()

0 ×

s2 = s1

0 ×

varer = s1._varer

```python
class Samling:
    def __init__(self):
        self._varer = []

    def legg_til(self, vare):
        self._varer.append(vare)

    def hent_varer(self):
        return self._varer


s1 = Samling()
s1.legg_til("A")
s2 = s1
varer = s2.hent_varer()
varer.append("B")

s1.legg_til("C")
print(s2.hent_varer())
```



Litt repetisjon...

__str__ og __repr__

- __**str**__ gir en lesbar tekstrepresentasjon av et objekt, typisk for brukeren
- __**repr**__ gir en representasjon som er nyttig for utviklere og feilsøking
- print(objekt) bruker __**str**__ hvis den er definert. Hvis ikke, brukes normalt __**repr**__
- Når vi skriver ut en liste eller ordbok med egendefinerte objekter, brukes repr() på elementene
- __**str**__ skal returnere en streng, ikke skrive den ut med print()
- **!! Bruk str(objekt) eller print(objekt), ikke kall objekt.__str__() direkte**

__eq__

- **==** sammenligner objekter ut fra klassens definisjon av likhet
- **is** undersøker om to referanser peker på nøyaktig samme objekt.
- Uten en egen **__eq__**-metode sammenlignes normalt objektidentitet for egendefinerte objekter
- **__eq__**(self, annen) sammenligner objektet som self viser til, med objektet som annen viser til.
- Metoden bør returnere **True** eller **False**.

Aggregering, innkapsling og grensesnitt

- **Aggregering** innebærer at et objekt bruker eller inneholder referanser til objekter fra andre klasser.
- Eksempel: Gruppe inneholder Person-objekter, og hvert Person-objekt bruker et Navn-objekt.
- Dette gir en avhengighet mellom klassene: Endringer i en klasses grensesnitt kan påvirke andre klasser
- Grensesnittet beskriver tjenestene klassen tilbyr. Instansvariabler og "hjelpemetoder" bør normalt ikke eksponeres.
- Metodenes argumenter og returverdier må ha de forventede typene

Tenk/Diskuter: Når skal to Student-objekter regnes som like? Er det nok at studentnummeret er likt, eller må også navn og studieprogram stemme overens? Hvem bestemmer hva som gjør to objekter like?

- *Klassen bestemmer hva som gjør to objekter like, ved å definere __eq__.*
- *Likhet bør baseres på egenskapene som er relevante for klassen, ikke nødvendigvis på hvordan objektet vises som tekst.*

Oppgaver

- **Oppgave 1 A-C henger sammen!**
- 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

Oppgave 1A

Gitt en klasse Vare med instansvariablene *vareid og* varenavn, implementer **eq** slik at to varer er like når både vare-ID og varenavn er like. Metoden skal håndtere argumenter av feil type uten å krasje.

Oppgave 1B

Du har en liste med Vare-objekter. Skriv en metode som tar imot en vare og returnerer True hvis en tilsvarende vare finnes i listen, ellers False. Bruk **eq** der det er hensiktsmessig. Ikke endre listen under søket.

Oppgave 1C

Lag en klasse Butikk som inneholder en liste med Vare-objekter. Implementer en metode for å legge til varer uten dubletter, og str som returnerer en lesbar oversikt over alle varene. Bruk en løkke og la Vare håndtere sin egen tekstrepresentasjon.

Ferdig?

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