IN1000 – Uke 9

Gruppe 8

Plan for timen

- Bittelitt repetisjon
- Magiske (spesielle) metoder
- Tidligere eksamener

Litt repetisjon

- Variabler med konstant startverdi, trenger ikke å være parameter

```
def __init__(self, tall, tekst):
    self._tall = tall
    self._tekst = tekst
    self._verdi = 0
```

Oppgave

Hvilke verdier får variablene til instansene av objektene av klasse Person her?

```
person = Person(18, "Kari")

def __init__(self, alder, navn):
    self._alder = 0
    self._navn = "Karl"

def __init__(self, alder, navn):
    self._alder = navn
    self._navn = alder

def __init__(self, alder, navn):
    self._alder = alder
    self._navn = navn

def __init__(self, a, b):
    self._alder = a
    self._navn = b

def __init__(self, alder, navn):
    self._a = alder
    self._n = navn
```

Løsningsforslag

_alder = 0

_navn = “Karl”

_alder = 18

_navn = “Kari”

_alder = “Kari”

_navn = 18

_a = 18

_n = “Kari”

_alder = 18

_navn = “Kari”

Metoder – når brukes return?

- Instansmetoder kan returnere noe, men må ikke.
- return og print() har ikke noe med hverandre å gjøre.
- return brukes gjerne når vi vil hente en verdi.

```
def liste_har_negativt_tall(liste:list):
    for tall in liste:
        if tall < 0:
            return True
        else:
            return False
```

Magiske metoder

- **Magiske metoder** er spesielle metoder i Python som vi kan overskrive for å bruke bestemte operatorer.
- Markeres med to understreker __ før og etter navnet på metoden.
  - Eksempel vi allerede har sett: __init__
- Vi skal aldri kalle på metoden direkte med navnet, f.eks. ikke: objekt.__str__()

__str__

- Implementering av denne gjør at vi enkelt kan representere objektet vårt som en streng. Det er programmerer som definerer hvordan denne utskriften skal se ut.
- Kalles når vi brukes str(objekt) eller print(objekt)

```python
class Hund:
    def __init__(self, navn, alder):
        self._navn = navn
        self._alder = alder

    def __str__(self):
        return self._navn


class Hund:
    def __init__(self, navn, alder):
        self._navn = navn
        self._alder = alder

hund = Hund("Pompel", 4)
print(hund)
```

```python
class Hund:
    def __init__(self, navn, alder):
        self._navn = navn
        self._alder = alder

    def __str__(self):
        return self._navn


class Hund:
    def __init__(self, navn, alder):
        self._navn = navn
        self._alder = alder

hund = Hund("Pompel", 4)
print(hund)
```

I første tilfellet vil utskriften være lik **self._navn**.
I det andre tilfellet vil utskriften være en **minneadresse**.

```
lisbethyang@macbookair web % python3 t
Pompel
lisbethyang@macbookair web % python3 t
<__main__.Hund object at 0x103149d30>
lisbethyang@macbookair web %
```

__eq__

- Gjør det mulig å sammenligne objekter med == basert på det programmerer definerer til å være “like objekter”.
- Kalles automatisk når vi bruker == mellom to objekter av en klasse med denne metoden definert.

```python
class Rektangel:
    def __init__(self, lengde, bredde):
        self._lengde = lengde
        self._bredde = bredde

    def __eq__(self, annen):
        return (self._lengde == annen._lengde and self._bredde == annen._bredde)

r1 = Rektangel(8, 6)
r2 = Rektangel(8, 6)
print(r1 == r2) # printer True

r3 = Rektangel(6, 8)
print(r2 == r3) # printer False
```

Sammenligning: *is* vs. ==

- X is Y: Er det *samme objekt* (lik adresse)?
- X == Y: Er det *like objekt* (likt innhold)?

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

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

person1 = Person("Per")
person2 = person1

print(person1 is person2)   # True
print(person1 == person2)   # True

person1 = Person("Per")
person2 = Person("Per")

print(person1 is person2)   # False
print(person1 == person2)   # True
```

Sjekke om noe er *None*

- Generelt: Bruk **is** i stedet for ==
- Grunnen til dette er at None er ett objekt – vi kan ikke ha to instanser av None
- **is** gir derfor alltid riktig svar, mens == er avhengig av hvordan en klasse har implementert __eq__