Objektorientert programmering i Python

IN1000

Høst 2019 – uke 9

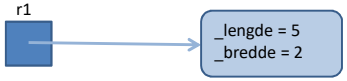
Siri Moe Jensen

Innhold uke 9
Mer komplekse strukturer

- Underveis-evaluering
- Referanser versus objekter (repetisjon + parameteroverføring)
- Spesielle metoder i egendefinerte klasser
  - fra objekt til streng
  - sammenligning av objekter
- Samlinger av objekter i beholdere (containers) som liste, mengde og ordbok.
- Egne klasser med referanser i instansvariable (objekter med objekter)

Underveis-evaluering
Mentimeter

Referanser til objekter



- Variabler som holder rede på objekter kalles *referansevariabler*
- Gjør det mulig å ta vare på og bruke objekter når vi trenger dem, akkurat som heltallsvariabler husker heltall til vi trenger dem.
- Selve objektet kan lagres "hvorsomhelst" i minnet, og være stort eller lite – referansevariabelen trenger bare plass til en *adresse*
- Referansevariabler kan brukes for å kalle på metoder i objektet:

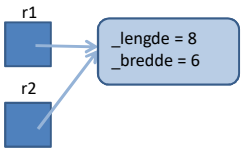
```
refVariabel.metode()
```

Referanser til objekter

- Ofte trenger vi ikke tenke på at selve objektet ikke ligger i variabelen
- Men noen ganger må vi huske forskjellen på referansen og objektet
  - Hvis vi tilordner verdien fra en referansevariabel til en annen (objektet kopieres ikke – variablene refererer til samme objekt!)
  - om vi sammenligner to referanser

Å kopiere en referanse

```
1 class Rektangel :
2     def __init__(self, len, bredde) :
3         self._lengde = len
4         self._bredde = bredde
5
6 r1 = Rektangel(8,6)
7 r2 = r1
```

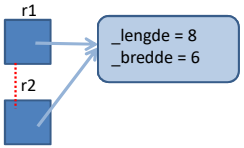


Sammenligning av referansevariable

Kan sammenligne enten:

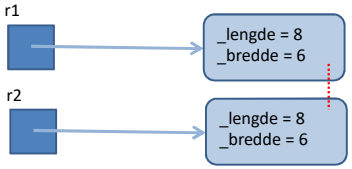
- Referansene, er det lik adresse dvs samme objekt?

```
assert (r1 is r2)
```



- Objektene, er det like objekter?

```
assert (r1 == r2)
```



Sammenligning av referansevariable

- Den som har skrevet klassen vet (bestemmer) hva som gjør to objekter "like".
- Vi kan lage en egen metode i klassen som tester likhet. Hvis metoden får navnet `__eq__` vil operatoren `==` sammenligne objekter av klassen slik vi bestemmer.
- Dette er gjort i Pythons List-klasse, slik at

  `liste1 == liste2`

  sammenligner alle elementene i to lister for oss

eksempel på __eq__ metode

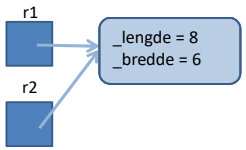
rektangelsml.py

```
1 class Rektangel :
2   def __init__(self, len, bredde) :
3     self._lengde = len
4     self._bredde = bredde
5
6   def __eq__(self, annen) :
7     return (self._lengde == annen._lengde and
8             self._bredde == annen._bredde)
9
```

Sammenligning av referansevariable

assert (rek1 is rek2)

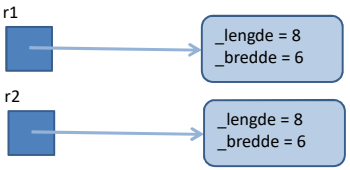
assert (rek1 == rek2)



assert (rek1 is not rek2)

assert (rek1 == rek2)

men bare hvis vi har definert __eq__ i Rektangel-klassen



Spesielle metoder

- `__eq__` er en av mange "magiske" metoder som har en spesiell betydning i Python
- felles er
  - innledende og avsluttende dobbel underscore (__) i metodenavnet
  - kalles på andre måter enn ved metodenavnet
  - eks: `__eq__` kalles når `==` brukes på objekter av klassen

Flere spesielle metoder

- Tabell 9.1 i boka viser en rekke andre spesielle metoder som kan implementere logiske (eks `==`, `!=`, `<`) og aritmetiske (eks `+`, `*`) operatorer for en klasse som trenger det
- `__init__` kalles ved opprettelse av nytt objekt, oppretter og initierer (gir startverdi til) instansvariablene
- `__str__` og `__repr__` gjør om objekter til strenger

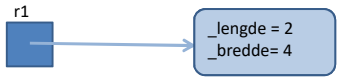
Objekter som strenger

For å gjøre om et objekt til en lesbar streng:

- **`__str__`** *Lag gjerne denne*
  - kalles når vi bruker **`print(s)`** og **`str(s)`**
  - returnerer en *brukervennlig* streng, lag den slik du ønsker
  - hvis den ikke finnes i klassen kalles **`__repr__`**
- **`__repr__`**
  - leverer en *komplett og entydig* representasjon av objektet
  - default: modul , klassenavn og minneadresse for objektet
  - kalles når **`__str__`** ikke finnes i klassen

"Skrive ut" objekter

Printer vi en referanse, får vi en (ofte litt kryptisk) tekst ut:




```
print(r1)
```

```
C:\Programmering>
C:\Programmering>python rektangel.py
<__main__.Rektangel object at 0x0000025FC328E6A0>

C:\Programmering>
```

resultat av __repr__

eksempel på __str__ metode

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

  def __str__(self) :
    penStreng = "Lengde: " + self._lengde + \
        ", bredde: " + self._bredde)
    return penStreng

r1 = Rektangel(8,6)
print (r1)
```

```
>python rek.py
Lengde: 8, bredde: 6
```

Oppsummert spesielle (magiske) metoder

- Spesielle/ magiske metoder kalles ikke ved navn.
  - __init__ kalles bak kulissene når vi oppretter et nytt objekt
  - __str__ kalles bak kulissene med str(r1) eller print(r1)
  - __eq__ kalles bak kulissene når vi bruker == mellom objekter av klassen
- Ofte nyttig å skrive **`__str__`** og **`__eq__`** for egne klasser
- NB: Metoden __str__ skal *returnere* en streng, ikke printe den!
- Hvis behov for å sortere f eks, trengs logiske operatorer (minimum **<** eller **>**) som kan teste rekkefølge

Oppgave

- Skriv ut innholdet i objektet n1
- Sjekk om objektene n1 og n2 er like

```
class Navn:
    def __init__(self,fornavn,mellom,etter):
        self._fornavn = fornavn
        self._mellom = mellom
        self._etter = etter

    def __str__(self):
        return self.naturlig()

    def __eq__(self, annen):
        return (str(self) == str(annen))

    def naturlig(self):
        natNavn = self._fornavn + " " + self._mellom + " " + self._etter
        return natNavn

n1 = Navn("a", "b", "c")
n2 = Navn("d", "e", "f")
```

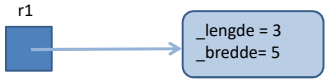
Løsning

```
class Navn:
    def __init__(self,fornavn,mellom,etter):
        self._fornavn = fornavn
        self._mellom = mellom
        self._etter = etter

    def __str__(self):
        return self.naturlig()

    def __eq__(self, annen):
        return (str(self) == str(annen))

    def naturlig(self):
        natNavn = self._fornavn + " " + self._mellom + " " + self._etter
        return natNavn

n1 = Navn("a", "b", "c")
n2 = Navn("d", "e", "f")


print (n1)
assert (n1 == n2)
```

«dot-notasjon»

- For å endre (eller lese av) ***innholdet*** i ***objektet***, kaller vi på metoder for det objektet vi er interessert i. Metoden adresserer vi med «*dot-notasjon*» på referansevariabelen.

```
r1.reduser(1,1)          # angir objekt og metode
print(r1.areal())
```



«dot-notasjon» i flere ledd

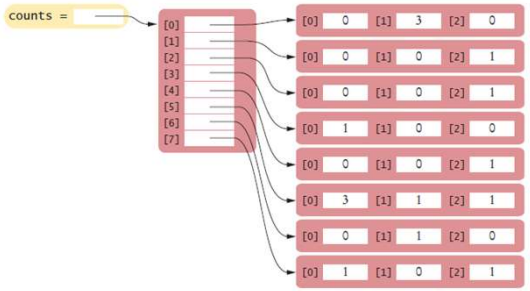
- En referanse er en verdi
- Konstanter, variable og funksjoner er eksempler på uttrykk som evaluerer til en verdi -> for eksempel en referanse
- uttrykk kan brukes i andre uttrykk

```
from rektangel import Rektangel
                                  # Oppretter nytt Rektangel objekt og
print(Rektangel(10,15).areal())   # kaller på metoden areal for dette,
                                  # printer til slutt returverdi fra areal
```

Samlinger av verdier (se uke 3)

- Beholdere (containers) er viktige verktøy i programmering
- Gjør det mulig å organisere og arbeide med samlinger av verdier – også (referanser til) objekter
- Beholdere tilbyr ulike egenskaper – velges ut fra behov
- Så langt har vi sett på
  - Lister (List). Rekkefølge, nummerert
  - Mengder (Set). Unummerert, uten dubletter
  - Ordbøker (Dictionary). Par av nøkkel (typisk tekst) – verdi
- Verdiene kan selv være (referanser til) samlinger – for eksempel lister

Samling av samlinger (liste med lister)



Eksempel: Informatikk-emner (kurs)

- Vi skal lage et program for å velge informatikk-emner
- Initielle krav: Kunne liste opp alle emner med id (emnekode), antall poeng og høst eller vår-semester
- Designer en klasse Emne med instansvariable som over
- Bruker beholdere for å organisere Emne-objekter
  - liste
  - ordbok

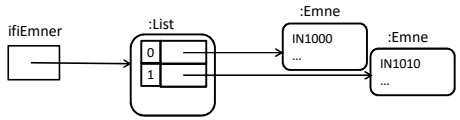
En klasse for emner

```
class Emne :
    def __init__ (self,emnekode,sem, stp) :
        self._emnekode = emnekode
        self._semester = sem
        self._studiepoeng = stp

    def __str__ (self) :
        linje = (self._emnekode + "(" +
                 self._semester + "): " +
                 str(self._studiepoeng) + " studiepoeng")
        return linje
```

Liste med objekter av egen klasse

- Eksempel: ifiEmner
- Hvert element i listen er (en referanse til) et emne-objekt

```
ifiEmner = []
ifiEmner.append(Emne("IN1000","host",10))
ifiEmner.append(Emne("IN1010","var",10))  # ..osv osv

for ettEmne in ifiEmner :
    print(ettEmne)            # kaller __str__()
```




```
IN1000(host): 20 studiepoeng
IN1010(var): 20 studiepoeng
```

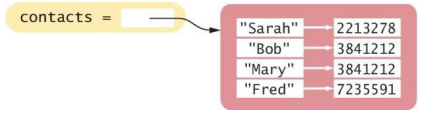
Samlinger av Emne-objekter

- Mer presist: Samlinger av referanser til Emne-objekter
- Dictionaries (ordbøker, maps) kan også ha referanser som verdier, typisk med en instansvariabel fra objektet som nøkkel
- Eks: Dictionary med Emne-objekter
  - Nøkkel: Emnekode
  - Verdi: Referanse til objektet for det emnet

Opprette Dictionary (ordbok)

- Et program som slår opp telefonnummer
- Bruker en ordbok der navn er nøkkel, og telefonnummer er verdien

```
contacts = { "Fred": 7235591, "Mary": 3841212, "Bob":
   3841212, "Sarah": 2213278 }
```

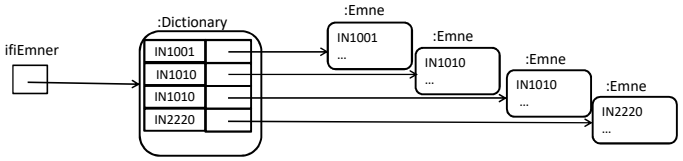


Ordbok med referanser

- Verdiene i en ordbok (dictionary) kan være referanser til objekter
- Eksempel: Ordbok `ifiEmner` med emner
- Nøkkel (entydig): Emnekode
- Verdi: Referanse til et Emne-objekt

Ordbok med referanser II

```
ifiEmner = {}              #opprettet tom ordbok

ifiEmner["IN1000"] = Emne("IN1000","host",10)
ifiEmner["IN1010"] = Emne("IN1010","var",10)
# etc

for ettEmne in ifiEmner.values() :
    print(ettEmne)       # Kaller __str__()
```



Instansvariabler som refererer andre objekter

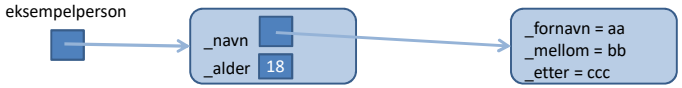
- Vi har sett på lister og ordbøker med (referanser til) objekter
- Sist uke så vi på klassen Person, med instansvariabel **_fulltNavn** som refererte til et objekt av klassen **Navn**

```
class Person :
  def __init__(self,fulltNavn,alder) :
    self._navn = fulltNavn
    self._alder = alder
```

Implementasjon av Person

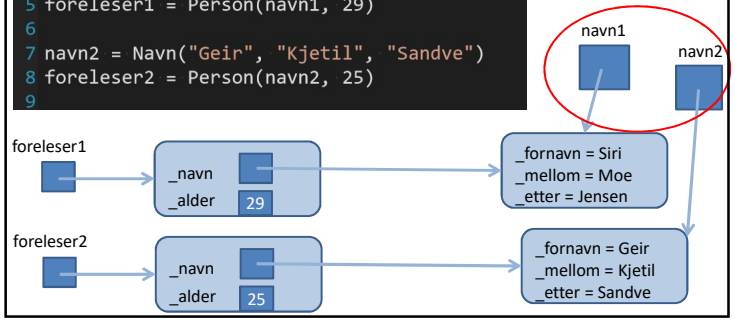
Filen person.py

```
class Person :
  def __init__(self,fulltNavn,alder) :
    self._navn = fulltNavn
    self._alder = alder
```



Klassen Person: Testprogram

```
testperson.py
1 from navn import Navn
2 from person import Person
3
4 navn1 = Navn("Siri", "Moe", "Jensen")
5 foreleser1 = Person(navn1, 29)
6
7 navn2 = Navn("Geir", "Kjetil", "Sandve")
8 foreleser2 = Person(navn2, 25)
9
```



Neste gang

- Matriser: Lister av lister (av lister av lister av...)
- Å lage egne beholdere
- Liveprogrammering av et større eksempel:
  Å sette det hele sammen

Takk for i dag!