Kort fra uke 7

Forrige uke så vi litt på bakgrunnen for at objektorientert programmering og objekt-orienterte språk dukket opp på 60-tallet – og vi så på hvordan du kan definere egne klasser, for deretter å opprette objekter og bruke metodene i disse i programmet ditt.

Så hva er egentlig klasser og objekter?

I Python så langt har dere brukt verdier av ulike innebygde **typer**: Heltall, sannhetsverdier, strenger og ordbøker. Strenger, lister og ordbøker er eksempler på mer sammensatte typer enn for eksempel hel- og flyttall. Dels inneholder de mer enn en verdi, dels tilbyr de tjenester for å manipulere disse dataene. Eks:

- Lister kan inneholde mange enkeltverdier og tilbyr tjenester som
  - append(element)
  - pop()
- Strenger består av 0 til mange tegn og kan manipuleres på ulike måter
  - fjerne blanke med strip()
  - gjøre om til små bokstaver med lower()

Dette er veldig generiske og anvendelige typer helt uavhengig av applikasjonsområde. Ved å definere **våre egne klasser** kan vi selv «konstruere» slike sammensatte – men mer skreddersydde – typer, og deretter opprette og bruke ett eller flere objekter av hver type. Dette er nyttig i veldig mange sammenhenger!

Forrige uke så vi på to enkle eksempler – klassene

- Student (som kun skulle brukes for å registrere en students oppmøte på gruppetimer)
- Navn (som lagret 3 deler av et navn og tilbød tjenester for å presentere dem på ulike måter)

```
class Student:
    def __init__(self, navn):
        self._antDeltatt = 0
        self._navn = navn

    def registrer(self):
        self._antDeltatt += 1

    def hentDeltakelse(self):
        return self._antDeltatt

    def hentNavn(self):
        return self._navn
```

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

  def sortert(self):
    # innhold

  def naturlig(self):
    # innhold
```

En klassedefinisjon er med andre ord en beskrivelse av et **mønster** – omtrent som en kakeform. Selv om vi har en klassedefinisjon i programmet vårt, har vi ingen data – og kan heller ikke gjøre noe med dem. Vi har kakeformen, men ingen kaker.

Når vi **oppretter et objekt** av klassen vår ved å bruke klassenavnet med parenteser bak (og eventuelt noen argumenter i disse) kan vi ta vare på dette objektet i en variabel – og bruke tjenestene vi har definert i klassen ved å kalle på metoder for dette objektet.

```
nyDeltaker = Deltaker()
```

Vi kan lage så mange slike objekter av en klasse vi ønsker i et program. Når klassen først er skrevet kan vi også importere den (ved å oppgi modulnavn, oftest første del av filnavnet) til andre programmer og opprette objekter av samme type der – uten å kopiere inn koden.

```
from navn import Navn

def hovedprogram():
    navn1 = Navn("Siri", "Moe", "Jensen")
    navn2 = Navn("Geir", "Kjetil", "Sandve")

    print (navn1.sortert())
    print (navn2.sortert())
    print (navn2.naturlig())

hovedprogram()
```