# INF1001 - Obligatorisk innlevering 6

Frist: 17.10.16 12:00

## Introduksjon

I denne oppgaven skal vi utvikle et verktøy for enkel analyse av tekster. Verktøyet skal kunne brukes til å finne ut hvor mange ord det er i en tekst, hvilke ord som forekommer ofte i teksten, og andre interessante spørsmål. For å gjøre dette trenger vi å vite hvor mange ganger et ord forekommer i en gitt tekst. Vi begynner med å innføre begrepet *term*. En term er et (*normalisert*) ord, et enkelt distinkt enhet av et språk, separert av mellomrom i en tekst. I teksten:

```
hei på deg
```

finner vi tre termer, `hei`, `på`, og `deg`. I denne oppgaven er termene normalisert ved at store og små bokstaver regnes som like, så `asp`, `Asp` og `ASP` er alle samme term.
En term er representert av objekter av klassen `Termteller`, som i tillegg til en tekststreng som representerer termen selv, inneholder data om hvor mange ganger termen forekommer i teksten. Vi trenger også en datastruktur for å ta vare på alle termene i teksten. Dette gjør vi ved et objekt av typen `Ordliste`. Ordlista skal holde oversikt over alle termene (objekter av klassen `Termteller`) som forekommer i teksten.

## Deloppgave 1: Termteller

**Filnavn:** *termteller.py* og *termteller_test.py*
Vi starter med å representere termene i teksten og hvor mange ganger de forekommer. Dette gjøres ved objekter av klassen `Termteller`. Denne skal inneholde to instansvariabler, en strengvariabel som representerer selve termen, og en heltallsvariabel som representerer antall ganger termen forekommer i teksten. Ved å lage et objekt av klassen `Termteller` for hver nye term som forekommer i teksten, og øke antallet hver gang den termen forekommer, sparer vi plass i datamaskinens minne. Det blir lettere å søke opp informasjon om hver enkel term, når man samler alle data i ett objekt.

1. Skriv klassen `Termteller`. Klassen skal ha en konstruktør der de to instansvariablene opprettes.

2. For å kunne bruke klassen `Termteller` til noe nyttig trenger vi noen metoder. I denne oppgaven skal du implementere metodene som er beskrevet under, i klassen `Termteller`:

   ```
   def hent_term(self)
   ```

   Her skal strengen som representerer termen returneres.

   ```
   def hent_antall(self)
   ```

   Her skal antall forekomster av termen returnes.

   ```
   def inkrement(self)
   ```

   Her skal antall forekomster av termen økes med 1.

3. Lag et lite testprogram som viser at klassen fungerer. Testen skal minimum opprette to objekter av typen `Termteller`, øke antallet på det ene objektet, og skrive ut termen og antall forekomster til begge objektene. Testprogrammet behøver ikke være stort, og kan for eksempel se slik ut:

   ```
   skog = Termteller("skog")
   tre = Termteller("tre")

   skog.inkrement()

   # test hente-metodene
   print(skog.hent_term(), skog.hent_antall())
   print(tre.hent_term(), tre.hent_antall())
   ```

   Dette eksempelet vil gi følgende utskrift:

   ```
   skog 2
   tre 1
   ```

Synes du denne oppgaven var vanskelig? Se øvingsoppgave 7.01 og oppgaver fra uke 6.

## Deloppgave 2: Ordliste

**Filnavn:** *ordliste.py* og *ordliste_test.py*
Vi trenger også en liste over alle termene som forekommer i en tekst: klassen `Ordliste`. En `Ordliste` skal holde styr på alle termene i teksten. Dette gjør den ved hjelp av en liste med objekter av typen `Termteller`.

1. Skriv klassen `Ordliste`. Denne skal inneholde en instansvariabel, en liste som skal inneholde objekter av typen `Termteller`. Opprett instansvariabelen i konstruktøren.

2. Skriv en metode `les(self, tekst)` som leser inn teksten, splitter den opp i termer (ord) og legger termene i ordlista. Argumentet `tekst` som sendes inn til metoden skal være en tekstfil.

   **Hint:** Her kan det være lurt å bruke metoden `legg_til_term(self, term)` fra neste oppgave.

   Vi skal jobbe med filer med flere termer på en linje. Termene på hver linje er separert av mellomrom. For å hente ut termene fra hver linje, kan dere bruke følgende kode:

   ```
   linje = "her er et eksempel med flere termer"
   termer = linje.split()
   for term in termer:
       # prosesser hver term
   ```

3. Skriv en metode `legg_til_term(self, term)` som tar inn strengen `term`. Metoden skal legge inn en ny term i ordlista hvis den ikke finnes fra før. Hvis termen allerede er der, skal antall forekomster økes med 1.

   NB! Som nevnt innledningsvis regnes store og små bokstaver som like, så "asp", "Asp" og "ASP" er alle samme term. For å få til dette kan det være lurt å bruke metoden `term.lower()` som konverterer alle bokstavene i termen til små bokstaver.

   **Hint:** Bruk klassen `Termteller`. Her kan det også være lurt å bruke metoden `finn_term(self, term)` fra neste oppgave.

4. Skriv en metode `finn_term(self, term)` som finner og returnerer en gitt termteller (et objekt av klassen `Termteller`) i ordlista. Hvis termen ikke finnes, skal vi returnere `None`. Merk at termen som sendes inn (argumentet `term`) her skal være en streng.

5. Skriv en metode `skriv_alle(self)` som skriver ut alle termer i ordlista til terminalen.

6. Skriv metoden `antall_termer(self)` som finner ut og returnerer hvor mange ulike termer det finnes i ordlista.

7. Skriv en metode `antall_forekomster(self, term)` som finner ut hvor mange ganger strengen `term` forekommer i teksten.

8. Skriv en metode `vanligste(self)` som finner og returnerer termen som forekommer oftest i teksten. Her ønsker vi å returnere et objekt av typen `Termteller`. Hvis det er flere termer som forekommer flest ganger, holder det å finne ett av dem.

9. **Ekstraoppgave (NB frivillig!):** Skriv en metode `antall_vanligste(self, antall)` som finner og returnerer de `antall` vanligste termene i teksten. Eksempel: Hvis antall er 10, skal de 10 vanligste termene returneres.

10. Test klassen med et lite testprogram. Dette kan for eksempel se ut som eksempelet under. Lag en liten testfil som programmet testes med, og sørg for at alle utskrifter gir forventet resultat. Merk at det kan være lurt å teste hver metode med en gang den er skrevet.

```
ordliste = Ordliste()

# innlesning
ordliste.les("tekst.txt")

# legge til term
ordliste.legg_til_term("skog")
ordliste.legg_til_term("skog")

# finne termer
termteller = ordliste.finn_term("skog")
# termteller = ordliste.finn_term("tre")
if (termteller):
    print(termteller.hent_term())
else:
    print("Fant ikke termen...")

# skriv ut alle termer
ordliste.skriv_alle()

# antall termer
antall_termer = ordliste.antall_termer()
print("Antall termer:", antall_termer)

# antall forekomster
antall_forekomster = ordliste.antall_forekomster("skog")
print("Antall forekomster av 'skog':", antall_forekomster)

# vanligste term
vanligste = ordliste.vanligste()
print("Den vanligste termen:", vanligste.hent_term(), vanligste.hent_antall())
```

Synes du denne oppgaven var vanskelig? Se øvingsoppgave 7.02, 7.03 og 7.04.

## Deloppgave 3: A study in scarlet

**Filnavn:** *scarlet.py*
Vi skal teste programmet vårt på den første Sherlock Holmes-boka *A study in scarlet* av Sir Arthur Conan Doyle. Last ned filen scarlet.txt[1] som inneholder boka i riktig format. Lag et program som benytter klassene `Termteller` og `Ordliste` til å beregne og skrive ut følgende informasjon om boka:

1. Hvor mange ulike termer forekommer i boka?
2. Hvor mange ganger forekommer termen *Holmes*?
3. Hvor mange ganger forekommer *elementary*?
4. Hvilken term forekommer flest ganger?

**Hint** Det tar noen sekunder å sjekke hele boka, så du kan spare mye tid på å lage en kort testfil du bruker til programmet er ferdig.

## Krav til innleveringen

1. Oppgaven må kunne kjøres på IFI sine maskiner.
2. Kun .py-filene, README.txt, scarlet.txt og en eventuell egenlaget testfil skal leveres inn.
3. Koden skal inneholde gode kommentarer som forklarer hva programmet gjør.
4. Programmet skal inneholde gode utskriftssetninger som gjør det enkelt for bruker å forstå.

## Fremgangsmåte for innlevering i INF1001

1. Lage en fil som heter README.txt. Følgende spørsmål skal være besvart i filen:
   - Hvterman synes du innleveringen var? Hva var enkelt og hva var vanskelig?
   - Hvor lang tid (ca) brukte du på innleveringen?
   - Var det noen oppgaver du ikke fikk til? Hvis ja:
     - Hvilke(n) oppave er det som ikke fungerer i innleveringen?
     - Hvorfor tror du at oppgaven ikke fungerer?

[1] Tekstfila er hentet fra Project Gutenberg, `http://www.gutenberg.org/cache/epub/244/pg244.txt` og modifisert noe. Blant annet er alle skilletegn fjernet.

   - Hva ville du gjort for å få oppgaven til å fungere hvis du hadde mer tid?

2. Logg inn på Devilry.
3. Lever alle .py-filene og .txt-filene i *samme innlevering*.
4. Husk å trykke lever og sjekk deretter at innleveringen din er komplett.

Den obligatoriske innleveringen er minimum av hva du bør ha programmert i løpet av en uke. Du finner flere oppgaver for denne uken her.