IN1140, H2017 – Oblig 1
Tekst i Python

Innleveringsfrist: 21.09 kl.23.59

I denne oppgaven skal vi jobbe med tekst i Python. Vi må derfor kunne lese inn filer og jobbe med tekststrenger. Vi skal også gjøre et forsøk på et program som utfører såkalt **tokenisering**, dvs bryter en tekst opp i ord. Tokenisering er et nødvendig utgangspunkt for de aller fleste språkteknologiske oppgaver. I denne oppgaven, skal vi også se litt på tilstandsmaskiner.

Innlevering av oppgaven skjer i Devilry. Se emnesiden for mer informasjon om reglement rundt innlevering samt bruk av Devilry. Registrer svarene dine i en fil som angir brukernavnet ditt slik:

`oblig1_brukernavn.py`

En perfekt løsning av denne oppgaven er verdt 100 poeng. Koden må kunne kjøres på IFIs maskiner og må inneholde kommentarer som forklarer hva koden gjør.

1 Strenger i Python (30 poeng)

Python Library Reference inneholder en oversikt over metoder som er tilgjengelig for forskjellige datatyper. For strenger (akkurat som for lister og tupler), finner du relevant informasjon på: `http://docs.python.org/library/stdtypes.html` Alternativt kan du skrive inn `help(str)` i et Python-shell for å se en oversikt over metodene som er tilgjengelig for strenger. Observer at det er flere metoder enn de som vises på skjermen og bruk "space" for å navigere nedover. Bruk "q" for å avslutte.

```
|  count(...)
|      S.count(sub[, start[, end]]) -> int
|
|      Return the number of non-overlapping occurrences of
|      substring sub in
|      string S[start:end].  Optional arguments start and end
|      are interpreted
|      as in slice notation.
```

Dette betyr at du for eksempel kan bruke `count` slik :

```
>>> mystring = "hello"
>>> mystring.count("l")
2
```

Bruk Pythons strengmetoder for å løse følgende oppgaver.

a) **Les inn en tekstfil**: Filen `In01.txt` inneholder norsk avistekst. Last ned filen fra emnesiden.

- Python er i utgangspunktet ikke så glad i norske tegn, så vi må derfor ha en spesiell kommentar som første linje i fila for å få dette til å virke:
  `# encoding: utf-8`
- For å lese inn hele filen som en streng må du åpne den, lese den og lukke den:

```
f = open("<sti-til-filen>")
filinnhold = f.read()
f.close()
```

b) **Tell forekomster**

- Bruk strengmetoden `count` til å avgjøre hvor mange ganger bokstavsekvensen *"er"* forekommer i teksten. (Du trenger ikke bare telle ordet *"er"*, en forekomst av feks *"er"* i *"hjerne"* skal også telles.)
- Hvor mange ord i teksten slutter på *"er"*? (NB! Her er vi ute etter endelsen *"er"* og ikke bare en forekomst hvor som helst i et ord (dvs vi vil telle *"tenker"* som en forekomst men ikke *"hjerne"*). I denne oppgaven trenger du ikke å håndtere tegnsetting; tell bare ordene som slutter på *"er"* og overse de som slutter på *"er,"* eller *"er."*, dvs forekomster av *er* etterfulgt av punktum, komma eller annen tegnsetting. **Hint:** Du kan bruke strengmetodene `split` og `endswith`

c) **Hent ut deler av et ord**

- I Python kan sekvenstypene (strenger, tupler og lister) manipuleres med slice-notasjon. Denne notasjonen lar oss lett hente ut en del av en større sekvens:

```
>>> liste = ["a", "b", "c", "d"]
>>> liste[2:4]
["c", "d"]
>>> liste[:-1]
["a", "b", "c"]
>>> streng = "abcde"
>>> streng[2:4]
"cd"
>>> streng[:-1]
"abcd"
```

  Bruk slice-notasjon til å lage en liste som inneholder de to siste bokstavene fra hvert ord i teksten.
- Sett sammen listen med endelser fra forrige oppgave til en streng av endelser med mellomrom mellom.

2 Tokenisering (20 poeng)

a) **Les inn filen som en liste** En annen måte å lese inn en fil i Python på er å lese inn filen `In01.txt` som en liste, der elementene i listen tilsvarer en linje i filen. Dette gjør vi med en `for`-løkke, slik:

```
f = open("<sti-til-filen>")
for line in f:
    ...
f.close()
```

Vi ønsker nå å få tilgang til alle ordene i teksten. I første omgang antar vi at ordene i teksten er adskilt av mellomrom. Utvid programmet ditt slik at det deler opp hver linje basert på mellomrom. Her skal du benytte deg av en `for`-løkke.

b) **Telle linjer og ord** Du kan nå telle antall ord. Utvid koden din med funksjonalitet for å telle ord. Her skal du ikke bruke `count`-metoden, men heller benytte deg av en teller som økes for hver runde i løkken. Hvor mange ord inneholder filen? Skriv ut resultatet.

3 Tokenisering av norsk tekst (30 poeng)

a) **Tokenisering fra oppgave 2** Ta utgangspunkt i tokeniseringsoppgaven fra oppgave 2. Les inn filen `InO1.txt` og skriv ut resultatet (ett ord per linje) til en fil.

b) **Inspisér resultatet**

- Beskriv feilene du finner og gi minst tre eksempler. F.eks er dine tokens fullstendige ord? Blir tegnsetting og anførselstegn skilt ut fra dine tokens?.

c) **Tokenisering med regulære uttrykk**: Vi skal nå jobbe med å forbedre tokeniseringskoden vår, basert på feilanalysen fra forrige oppgave.

- Skriv et regulært uttrykk som definerer et gyldig norsk ord. Her bør du bruke disjunksjon for å beskrive forskjellige typer ord (f.eks. ulike typer tegnsetting, bokstavsekvenser, osv.).
- Benytt deg av Python's `re.findall` for å hente ut ordene som matcher uttrykket ditt fra linjene med tekst. Denne metoden tar et regulært uttrykk og en streng og returnerer en liste med alle treff. F.eks. vil `re.findall("([0-9])","inf1820")` gi ut listen `['1', '8', '2', '0']`.

d) **Feilanalyse**: Hvilke feil gjenstår? Gi minst to eksempler på feil som gjenstår og gi forslag til hvordan disse kan rettes opp.

4 Endelig tilstandsmaskiner (20 poeng)

Tegn en endelig tilstandsmaskin som gjenkjenner følgende språk, der alfabetet er $\{a, b\}$. Alle tilstandsmaskinene skal være deterministiske.

(a) $\{\ w \mid w$ inneholder minst tre b'er $\}$ (f.eks. skal $abbab$ og $ababaababbb$ aksepteres, men ikke $aaabaaa$ og ab).

(b) $\{\ w \mid$ hver oddetallsposisjon i w er en $b\ \}$ (Maskinen kan godt akseptere den tomme strengen).

(c) $\{\ w \mid w$ inneholder ikke substrengen $bba\ \}$ (Maskinen kan godt akseptere den tomme strengen).

(d) $\{\ w \mid w$ inneholder et partall antall a eller odde antall $b\ \}$

Lagre de ferdige maskinene som .gif(.png|.jpg)-filer, og lever dem sammen med koden din, som separate filer med navnene `4a.gif(.png|.jpg)`, `4b.gif(.png|.jpg)`, `4c.gif(.png|.jpg)` og `4d.gif(.png|.jpg)`.