Flere oppgaver for tekstbehandling, ordlister og Pandas
Avansert kurs ved Folkeuniversitetet

Tekstbehandling

Oppgave 1 *Lipogram*

I 1939 tok forfatter Ernest Vincent Wright på seg utfordringen å skrive en hel roman uten å bruke bokstaven "e". Boken heter *Gadsby* og er over 50 000 ord lang! Tekster som unngår en spesiell bokstav kalles et *lipogram* og det å unngå bokstaven "e" er spesielt vanskelig siden det er den mest brukte bokstaven i engelsk. La oss bruke Python til å teste en bit av *Gadsby* og dobbeltsjekke at Wright ikke har sneket inn noen "e"-er.

a) Lagre første avsnitt av *Gadsby* i en variabel ved å kopiere koden under:

```
gadsby_avsnitt = """If Youth, throughout all
   history, had had a champion to stand up for it;
    to show a doubting world that a child can
   think; and, possibly, do it practically; you
   wouldn't constantly run across folks today who
   claim that "a child don't know anything." A
   child's brain starts functioning at birth; and
   has, amongst its many infant convolutions,
   thousands of dormant atoms, into which God has
   put a mystic possibility for noticing an adult'
   s act, and figuring out its purport"""
```

b) Bruk `in` operatoren til å sjekke om bokstaven `'e'` finnes i avsnittet. Skriv ut resultatet til terminalen

c) **BONUS:** Den andre mest vanlige bokstaven i engelsk er bokstaven "a". Bruk en `for`-løkke til å løkke over hver bokstav i teksten og sjekk om det er en `'a'`. Opprett en variabel som teller antall `'a'`-er i teksten ved å øke med `1` hver gang en bokstav er `'a'`.

Oppgave 2 *Oversette en sangtekst*

Det er kanskje litt delte meninger om hvorvidt man burde oversette sangtekster eller ikke. Disney-sanger blir gjerne oversatt, slik at omtrent samme tekst finnes både på norsk og engelsk (og en hel del andre språk), men ordlyden kan være litt forskjellig.

Her skal vi se på et utdrag av den engelske og norske teksten av «Circle of Life» / «En sirkel av liv», og sammenligne linje for linje hva som faktisk blir sunget. Vi skal jobbe oss fra to strenger, en for den engelske teksten og en for den norske, til et program som skriver ut teksten, linje for linje, annenhver gang på norsk og engelsk.

a) Du kan lage strenger som går over flere linjer, dvs. inkludert linjeskift, med å bruke *triple* fnutter, slik:

```
1 tekst_engelsk = """...
2 ...
3 ..."""
```

Lag to variabler, f.eks, `tekst_engelsk` og `tekst_norsk` som inneholder henholdsvis

```
In the circle of life
It's the wheel of fortune
It's the leap of faith
It's the band of hope
'Til we find our place
On the path unwinding
In the circle, the circle of life
```

og

```
I en sirkel av liv,
Den som favner alt.
Alt fra sorg og savn
til kjærlighet.
Der vi finner vår plass
På den vei vi vandrer.
I en sirkel, en sirkel av liv.
```

Skriv de ut for å sjekke at programmet ditt fungerer så langt.

b) Bruk `splitlines` for å dele opp teksten linjevis, og legge innholdet i en liste. Lag en liste for hvert språk, f.eks. `linjer_engelsk` og `linjer_norsk` .

c) Skriv først ut inneholdet i den ene listen. Lag en løkke som itererer over listen som inneholder sangteksten på engelsk og skriver ut sangteksten linje for linje.

d) Utvid løkken fra forrige deloppgave slik at når en linje er skrevet ut på engelsk, så følger den norske oversettelsen etterpå. Her kan du bruke `zip`-funksjonaliteten, altså skrive (f.eks.)

```
1 for (linje_en, linje_no) in zip(linjer_engelsk,
     linjer_norsk):
2     ...
```

Du skal altså fylle inn det som mangler inni løkken, slik at utskriften på skjermen blir seende slik ut:

```
In the circle of life
I en sirkel av liv,

It's the wheel of fortune
Den som favner alt.

...
```

Oppgave 3 *Tacofredag til torsdag*

Stian har et program for å automatisk sende epost til sine kollegaer for å invitere til sammenkomster. For å invitere til fredagstaco skrev han en invitasjon

inn i en tekststreng slik:

```
invitasjonstekst = """
Hei alle! Førstkommende fredag har alle som ønsker
   muligheten til å bli med på årets første
   fredagstaco!
Spør du meg så finnes ingen bedre måte å feire
   fredagen på enn med taco!!
Vi sees (forhåpentligvis) på fredag,
Stian
"""
```

Etter å ha skrevet inn teksten fant Stian ut at fredag ikke passer bra fordi det overlapper med vaffelkvelden til en kollega. Han vil derfor modifisere invitasjonen slik at det står torsdag i stedet for fredag.

a) Lim invitasjonsstrengen over inn i en `Python` fil

b) Bruk `replace` til å bytte ut alle steder det står "fredag" med "torsdag" i invitasjonsteksten.

c) Skriv ut den nye invitasjonen til terminalen

Oppgave 4

Vi har fått tilsendt en liste med middagsretter gjennom en chattetjeneste og har kopiert den rett inn i Python under variabelen `middagsliste`:

```
middagsliste =
'"laks_og_ovnsbakte_grønnsaker","gulrotsuppe","
   kyllingwok","mossaka","spansk_gazpacho","quiche","
   fylt_paprika","hamburgere","sushi","
   kylling_med_potet-_og_tomatsalat","gyoza","
   meksikansk_taco","linsesuppe","tortelloni","pizza
   ","potet-_og_purresuppe","soyamarinert_laks","
   spanske_kjøttboller","lammegryte","steambuns","ø
   rret_med_quinoasalat","shepherds_pie","tomatsuppe
   ","helstekt_kylling","kremet_pasta_med_ørret","
```

```
    kylling-_og_sopprisotto","enchiladas","
    vegetarisk_thaigryte","asiatisk_kyllingsalat","
    ungarsk_gulasj","kylling_tikka_masala"'
```

Dessverre er den uoversiktlig å lese, og det vil vi gjøre noe med. Resultatet bør se slik ut:

```
asiatisk kyllingsalat
enchiladas
fylt paprika
gulrotsuppe
gyoza
hamburgere
helstekt kylling
kremet pasta med ørret
kylling med potet- og tomatsalat
kylling tikka masala
kylling- og sopprisotto
kyllingwok
laks og ovnsbakte grønnsaker
lammegryte
linsesuppe
meksikansk taco
mossaka
pizza
potet- og purresuppe
quiche
shepherds pie
soyamarinert laks
spansk gazpacho
spanske kjøttboller
steambuns
sushi
tomatsuppe
tortelloni
ungarsk gulasj
vegetarisk thaigryte
ørret med quinoasalat
```

a) Bruk string-funksjonen `.split()` for å skille hver middagsrett ut i en liste.

b) Bruk en løkke til å gå gjennom hver middagsrett i lista og bruk `replace` til å bytte ut alle understreker med mellomrom i middagsrettene. Fjern også alle anførselstegn.

c) Sorter lista med funksjonen `sorted`.

d) Slå sammen lista med `''.join(middagsretter)` (modifiser så det funker slik du ønsker).

e) Skriv den nye versjonen av middagsrettlista til fil *middagsretter.txt*.

Ordlister

Oppgave 5 *Verdens høyeste fjell*

I denne oppgaven skal vi øve oss litt på geografi, og litt på bruk av oppslagsverk. Vi skal bruke et eksisternede oppslagsverk og trekke ut informasjon fra dette, der vi igjen kommer til å lage nye oppslagsverk for å skrive ut informasjonen på en hensiktsmessig måte.

Du kan ta utgangspunkt i følgende kode – kopier denne over til øverst i programmet ditt:

```
1 fjelltoppinfo = {
2                 "Annapurna I" : {"høyde" : 8091, "land
                     " : ["Nepal"]},
3                 "Cho Oyu" : {"høyde" : 8188, "land" :
                     ["Nepal", "Kina"]},
4                 "Dhaulagiri I" : {"høyde" : 8167, "
                     land" : ["Nepal"]},
5                 "K2" : {"høyde" : 8611, "land" : ["
                     Pakistan", "Kina"]},
6                 "Kangchenjunga" : {"høyde" : 8586, "
                     land" : ["Nepal", "India"]},
7                 "Lhotse" : {"høyde" : 8516, "land" : [
                     "Nepal", "Kina"]},
```

```
8                   "Makalu" : {"høyde" : 8485, "land" : [
                        "Nepal", "Kina"]},
9                   "Manaslu" : {"høyde" : 8163, "land" :
                        ["Nepal"]},
10                  "Mount Everest" : {"høyde" : 8848, "
                        land" : ["Nepal", "Kina"]},
11                  "Nanga Parbat" : {"høyde" : 8126, "
                        land" : ["Pakistan"]},
12                  }
```

Dette er en oversikt over verdens 10 høyeste fjelltopper, deres høyde (i meter over havet) og hvilke land disse ligger i.

a) Lag et nytt tomt oppslagsverk, `etter_høyde`. Gå igjennom elementene i `fjelltoppinfo` og overfør relevant informasjon til `etter_høyde` slik at denne får høyde som nøkler og navn på fjelltoppene som elementer.

b) Lag en sortert liste over alle høydene ved å *kaste* `etter_høyde.keys()` til en liste, som du deretter sorterer. Bruk denne til å skrive ut alle fjelltoppene etter høyde, slik:

```
Fjelltopper etter høyde:
  Mount Everest (8848 moh.)
  K2 (8611 moh.)
  ...
```

c) Lag et nytt oppslagsverk som angir hvilke fjelltopper som ligger i hvilket land, dvs. har land som nøkler og lister over hvilke fjell som ligger i hvert land som elementer. Skriv disse ut, omtrent slik:

```
Fjelltopper etter land:
  Nepal: Annapurna I, Cho Oyu, Dhaulagiri I, ...
  ...
```

d) Til slutt vil vi skrive ut hvilket fjell som er det høyeste i hvert land (av de som er med i oversikten over).

Lag en løkke som går igjennom hvert land (du kan bruke `etter_land.keys()`). For hvert land skal du gå igjennom alle landene som ligger i dette landet, og bruke `fjelltoppinfo` til å finne ut hvor høyt hvert fjell

er. Finn en måte å sammenligne disse på, slik at du finner ut hvilket fjell som er høyest for hvert land.

Skriv ut resultatet omtrent slik:

```
Det høyeste fjellet i Nepal er Mount Everest (8848
    moh.)
...
```

Oppgave 6 *Morsekode*

I denne oppgaven skal du bruke løkker og oppslagsverk til å lage din egen morseoversetter som tar inn en beskjed i vanlige bokstaver og skriver ut beskjeden med morsetegn.

Kodesnutten under oppretter et oppslagsverk som lagrer halve morsealfabetet slik at bokstavene er nøkler og morsetegnene er verdier:

```
morsekode = { 'a':'.-', 'b':'-...', 'c':'-.-.',
              'd':'-..', 'e':'.', 'f':'..-.',
              'g':'--.', 'h':'....', 'i':'..',
              'j':'.---', 'k':'-.-', 'l':'.-..',
              'm':'--', 'n':'-.'}
```

a) Finn en oversikt over morsealfabetet (f.eks: https://no.wikipedia.org/wiki/Morsealfabetet#Tegnsettet) og fyll inn resten av alfabetet i oppslagsverket

b) Bruk `input` til å be brukeren om en beskjed og lagre resultatet i en variabel, `beskjed`

c) Opprett en tom tekststreng `kodet_beskjed =''` som vi skal bruke til å lagre morsekoden.

d) Nå skal vi printe ut hver bokstav i beskjeden for seg selv. For å gjøre det kan vi bruke en `for`-løkke. Bruk kodemønsteret `for bokstav in beskjed` for å iterere gjennom hver bokstav i beskjeden og print ut variabelen `bokstav` for hver iterasjon.

e) Neste steg er å faktisk kode beskjeden. Bruk betingelsen `if bokstav in morsekode` inne i løkka for å sjekke at en bokstav finnes i morsealfa-

betet. Dersom den gjør det skal du slå opp i `morsekode`-oppslagsverket for finne rett morsetegn og legge det til `kodet_beskjed` **HINT:** `kode +=morsekode[bokstav]`

f) Hvis `beskjed` inneholder en bokstav som ikke finnes i morsealfabetet kan vi ikke oversette den. Bruk en `else` blokk til å legge til `bokstav` til `kodet_beskjed` dersom `bokstav` ikke finnes i morsekode.

g) Skriv ut den morsekodete beskjeden etter løkka.

h) Test programmet ditt. Skriv for eksempel inn beskjeden `'hei'`. Finner du noen svakheter eller mangler med programmet? Skriv dem ned før du går videre.

i) Et problem med programmet vi har laget er at det ikke er mellomrom mellom morsetegnene. Det gjør det vanskelig å se når et tegn starter og slutter. Fiks dette problemet ved å legge til et mellomrom etter hvert morsetegn.

j) Et problem med programmet nå er at er vanskelig å skille mellom ord siden mellomrom nå brukes både for å skille mellom ord og bokstaver. I morsekode bruker man derfor dobbelt mellomrom for å skille mellom ord. For å fikse dette trenger du en ekstra betingelse som sjekker om bokstav er et mellomrom. Dersom det er tilfelle skal du legge til to mellomrom til `kodet_beskjed`. **HINT:** bruk `elif`

k) Test programmet ditt med å skrive inn "python er kult!" og sjekk at du får et output som ligner dette:

```
Skriv en beskjed: python er kult!
Morsekode:
.--. -.-- - .... --- -.   . .-.   -.- ..- .-.. - !
```

Gratulerer! Du har nå en helt egen morseoversetter

l) **BONUS:** Kom du på noen andre svakheter/forbedringsmuligheter i oppgave g)? Prøv å utvide programmet ditt til å bli så robust som mulig. Forslag til forbedringer kan være: Hva gjør du hvis en bokstav er "-" eller "."? Hva med tall eller andre spesialtegn?

Pandas

Oppgave 7 *Været med Pandas*

I denne oppgaven skal vi analysere værmeldingene over et år for et ukjent sted i Brazil med statistikkbiblioteket Pandas.

a) Last ned filen *været.csv* og lag et nytt Python-program. Sørg for at begge filene ligger i samme mappe. Ta en titt på csv-filen ved å åpne den og se hvordan den ser ut. Vi kan nå begynne å analysere dataene i filen. Bruk Pandas til å lese inn csv-filen til en dataframe.

b) Analyser filen ved å printe ut informasjon om den. Bruk `head()`, `info()`, `describe()`, `dtypes`, `shape` og lignende til å få en oversikt over dataene som er lest inn. Bruk gjerne dokumentasjonen til Pandas for å se hvordan for eksempel `head()` fungerer.

c) Hva er gjennomsnittlig makstemperatur og hva er gjennomsnittlig minstetemperatur? Hvor mye skiller det seg fra medianen for samme kategorier?

d) Hvilken dag hadde høyest makstemperatur og hva var minstetemperaturen denne dagen? Regnet det? Finn det samme for dagen med lavest minstetemperatur.

e) Tell hvor mange dager det var regn den dagen og hvor mange dager det ikke var regn den dagen. Gjør det samme for regn i morgenkategorien. Stemmer de overens?

f) Plot hvor mange dager det regner og hvor mange dager det ikke regner det er i et barplot. Bruk `dataframe["RainToday"].value_counts()` for å finne verdiene som skal plottes. Deretter finnes det en innebygd `plot`-funksjon i `value_counts().plot(kind="bar")`.

g) Lag en dataframe som inneholder kun dager det regner der minstemperaturen er over 10 grader. Sorter dataframen på makstemperatur i synkende rekkefølge.

h) Fjern alle kategoriene bortsett fra dagen, minstetemperatur, makstemperatur og om det regner i morgen.

i) Skriv den nye dataframen til en ny excel-fil *været_regn.xls*.