Data i Python

#hack4no

Vi vil nå utvikle en enkel geografi-quiz. Underveis vil vi bli kjent med noen avanserte datatyper som lister og tabeller, samt lære hvordan vi kan skrive til og lese fra fil.

1 Mange ting på en gang

Vanligvis peker en variabel til en ting om gangen: Et navn, et årstall, en tekst og så videre. Ofte vil man se på flere ting på en gang. Til dette har Python flere litt mer avanserte datatyper. En `list` eller liste er omtrent det det høres ut som: En liste av ting. Disse tingene kan være tall, tekststrenger eller til og med andre lister. Lister markeres med firkantparanteser, `[]`:

PYTHON SHELL

```
>>> handleliste = ["melk", "brød", "løk", "potet", "melk"]
>>> len(handleliste)
5
>>> sorted(handleliste)
['brød', 'løk', 'melk', 'melk', 'potet']
```

Vi kan også legge til og fjerne ting fra lister, eller hente ut enkelte ting eller dellister. Legg merke til at Python begynner å telle fra 0 i stedet for 1.

PYTHON SHELL

```
>>> handleliste.append("brunost")
>>> handleliste
['melk', 'brød', 'løk', 'potet', 'melk', 'brunost']
>>> handleliste[0]
'melk'
>>> handleliste[1]
'brød'
>>> handleliste[3:5]
['potet', 'melk']
>>> del handleliste[0]
>>> handleliste
['brød', 'løk', 'potet', 'melk', 'brunost']
```

En annen viktig datatype er `dict` (*dictionary*). I en `dict` ordnes par av nøkler og verdier. Disse markeres av krøllparanteser, `{}`:

PYTHON SHELL

```
>>> hovedsteder = {"Norge": "Oslo", "Danmark": "København", "Sverige": "
    Stockholm"}
>>> len(hovedsteder)
3
>>> hovedsteder["Sverige"]
'Stockholm'
>>> hovedsteder["Finland"] = "Helsinki"
>>> hovedsteder
{'Finland': 'Helsinki', 'Norge': 'Oslo', 'Sverige': 'Stockholm', 'Danmark':
     'København'}
>>> list(hovedsteder)
['Finland', 'Norge', 'Sverige', 'Danmark']
```

Legg merke til at rekkefølgen på tingene i en `dict` kan forandre seg.

2 En enkel quiz

Vi skal nå lage et enkelt spørreprogram basert på disse nye datatypene. Skriv inn programmet `enkel_quiz.py` på neste side.

I tillegg til at vi bruker `list`er og `dict`er er det også et par nye ting som skjer her:

- Helt først sier vi `import random`. Python er et veldig kraftig programmeringsspråk, men mye av effektiviteten ligger i pakker vi må si i fra når vi vil bruke. Her ønsker vi å spørre om hovedstaden i et tilfeldig land, og vi bruker derfor pakken `random` for å plukke ut et land med funksjonen `random.choice`.

enkel_quiz.py

```
import random

hovedsteder = {"Norge": "Oslo", "Danmark": "København", "Sverige":
               "Stockholm", "Finland": "Helsinki", "Island": "Reykjavik"}

land = random.choice(list(hovedsteder))
svar = input("Hva er hovedstaden i " + land + "? ")

if svar == hovedsteder[land]:
    print("Riktig")
else:
    print("Feil")
```

- For å sjekke om svaret er riktig sammenligner vi `svar` med `hovedsteder[land]`. Til slike sammenligninger bruker vi to likhetstegn, `==`.
- Ved hjelp av `if` og `else` kan vi få programmet til å oppføre seg forskjellig avhengig av om svaret er riktig eller ikke. Legg merke til kolonet, `:`, og innrykket. Innrykk brukes i Python for å markere kode som hører sammen.

3 Flere spørsmål

Vi kan også be programmet vårt stille oss flere spørsmål om gangen. Til dette bruker vi løkker. Den vanligste løkken er noe som kalles en `for`-løkke. Denne kan typisk gjøre noe for hver ting i en liste. For eksempel vil det følgende programmet spørre oss om alle landenes hovedsteder:

alle_land.py

```
hovedsteder = {"Norge": "Oslo", "Danmark": "København", "Sverige":
               "Stockholm", "Finland": "Helsinki", "Island": "Reykjavik"}

for land in hovedsteder:
    svar = input("Hva er hovedstaden i " + land + "? ")

    if svar == hovedsteder[land]:
        print("Riktig")
    else:
        print("Feil")
```

Siden rekkefølgen i `hovedsteder` forandrer seg hver gang vi kjører programmet bruker vi ikke `random` i dette eksempelet. Legg merke til hvordan innrykket viser hvilken kode som gjentas for hvert land i `for`-løkken.

En annen type løkke er `while`-løkken. I stedet for å gjentas for hver ting i en liste vil denne gjentas så lenge en betingelse er oppfylt. Vi kan bruke dette til å la programmet fortsette å spørre oss helt til vi finner de riktige svarene.

alle_riktig.py

```
import random

hovedsteder = {"Norge": "Oslo", "Danmark": "København", "Sverige":
               "Stockholm", "Finland": "Helsinki", "Island": "Reykjavik"}

while len(hovedsteder) > 0:
    land = random.choice(list(hovedsteder))
    svar = input("Hva er hovedstaden i " + land + "? ")

    if svar == hovedsteder[land]:
        print("Riktig")
        del hovedsteder[land]
    else:
        print("Feil")
```

Denne gangen sletter vi hovedstedene etterhvert som vi svarer riktig. Det betyr at vi ikke kan bli spurt om dem igjen. Derfor vil programmet bare spørre oss om hovedsteder vi ikke allerede har klart. I `while`-løkken spesifiserer vi at vi vil at programmet skal fortsette å spørre så lenge det finnes hovedsteder igjen å spørre om!

4 Skrive til fil

En viktig funksjon når man jobber med datamaskiner er å lagre informasjon. Den vanligste måten å gjøre dette på er å skrive informasjonen til en fil. La oss se hvordan det gjøres i Python! Og samtidig lage en versjon av hovedstad-quizen vår som vi kan ta med på hytta!

La oss først se om vi klarer å skrive en enkel tekstfil. Prøv det følgende i *REPL*'en:

Python Shell

```
>>> with open("hei.txt", mode="w") as fil:
...     fil.write("Hei fil!")
...
8
```

Dette vil skrive til en fil som heter `hei.txt`. Sjekk at du finner filen og at den har innholdet "`Hei fil!`".

Vi vil nå prøve å skrive ut hovedstadquizen til en tekstfil, slik at vi kan skrive den ut og ta den med på hytta.

quiz_til_fil.py

```
hovedsteder = {"Norge": "Oslo", "Danmark": "København", "Sverige":
               "Stockholm", "Finland": "Helsinki", "Island": "Reykjavik"}

with open("hovedstadquiz.txt", mode="w") as quizfil:
    for nr, land in enumerate(hovedsteder, start=1):
        quizfil.write(str(nr) + ". Hva er hovedstaden i " + land + "?\n\n")
```

Som vanlig er det et par nye ting å legge merke til:

- Vi spesifiserer at `mode` er `w`, det vil si `w`*rite*. Med andre ord at vi vil skrive til filen `hovedstadquiz.txt`. Andre muligheter er *read* og *append*. Førstnevnte bruker vi når vi vil lese en fil, mens *append* brukes for å skrive til uten å overskrive en fil.
- For å legge til spørsmålsnummer bruker vi `enumerate`. Denne gir oss både et tall og en ting fra en liste. Siden Python vanligvis teller fra 0 sier vi at den skal starte å telle på 1.
- Når vi skriver til en fil må vi også fortelle hvor det skal være linjeskift. Dette markeres med den spesielle koden `\n`.

5 Alle hovedsteder i hele verden!

Til slutt skal vi se på hvordan lese informasjon fra filer. I spillet vårt har vi foreløpig skrevet inn hovedstedene selv øverst i spillet. Men det blir kjedelig i lengden!

For å lage et litt realistisk (og dermed litt komplisert) eksempel på hvordan man kan lese en fil vil vi bruke en fil som inneholder mye informasjon om alle verdens land, og bare plukke ut den informasjonen vi trenger.

På siden `http://opengeocode.org/download/cow.php` finnes datasettet *Countries of the World (COW)*. Last ned `Countries of the World - CSV`-filen til datamaskinen din. Dette er en fil som heter `cow.txt`. Pass på at den ligger i samme katalog som pythonprogrammene vi har skrevet. Åpne filen og se på den. Det er mye informasjon her!

La oss først se om vi klarer å lese den i Python. Lag et nytt program:

les_cow.py

```
with open("cow.txt", mode="r") as infofil:
    for linje in infofil:
        print(linje)
```

Dette programmet skal skrive hele filen til skjermen. Om du har problemer med innlesingen kan du prøve å spesifisere at filen er kodet i UTF-8 ved å bruke `open('cow.txt', encoding='utf-8')`. Det nye her er at vi kan behandle filen som om den er en liste, hvor hver linje i filen er en ting i listen.

La oss nå se litt nærmere på selve filen `cow.txt`. Først kan vi legge merke til to ting:

- Øverst i filen er det flere linjer som begynner med tegnet `#` som ikke inneholder informasjon vi er interessert i. Vi bør ignorere disse linjene.
- Hver linje (unntatt linjene som starter med `#`) inneholder informasjon om et land, med hvert informasjonsfelt skilt med semikolon. Vi vil plukke ut noen av disse informasjonsfeltene.

Utvid programmet ditt slik at det ser slik ut:

les_cow.py

```
with open("cow.txt", mode="r") as infofil:
    next(infofil)
    for linje in infofil:
        if linje.startswith("#"):
            continue

        felter = linje.split(";")
        print(felter[4], felter[38])
```

Når du kjører dette programmet vil du få en liste over land og hovedsteder. La oss skjønne hva som skjer!

- Vi kan sjekke om en linje begynner med tegnet `#` ved hjelp av `linje.startswith`. Dersom den gjør det går vi videre til neste linje med `continue`. Det betyr at resten som skjer i løkke blir hoppet over for denne linjen.
- Vi deler hver linje i en liste av felter ved hjelp av `split`, og vi forteller `split` at vi vil bruke semikolon, `;`, som skilletegn mellom feltene.
- Etter litt prøving og feiling finner vi at land og hovedstad ligger som felt 4 og 38 (telt fra 0). Vi skriver derfor ut disse feltene.
- Det er problem med første linje i `cow.txt`-filen. Den ser ut som om den begynner med en `#`, men det ligger et usynlig tegn helt først som gjør at testen vår ikke slår inn. I stedet for å lage en mer avansert test hopper vi bare over første linje ved å si `next(infofil)` rett etter at vi åpner filen.

Om du ser på listen over land og hovedsteder som skrives ut vil du se et par små utfordringer til. Den første linjen som skrives ut er ikke et land, men en forklarende overskrift. Vi vil ikke at denne skal være med i listen vår over hovedsteder. En del land (eller "land") har ingen hovedstad. Disse kan vi heller ikke ta med i quizen. Den neste versjonen av programmet tar også høyde for disse utfordringene, samtidig som dataene leses inn i `dict`en `hovedsteder` i stedet for å skrives til skjerm.

verdensquiz.py

```
import random

hovedsteder = {}

with open("cow.txt", mode="r") as infofil:
    next(infofil)
    for linje in infofil:
        if linje.startswith("#"):
            continue

        felter = linje.split(";")
        if len(felter[0].strip()) == 2 and len(felter[38].strip()) > 0:
            hovedsteder[felter[4].strip()] = felter[38].strip()

with open("verdensquiz.txt", mode="w") as quizfil:
    for nr, land in enumerate(hovedsteder, start=1):
        quizfil.write(str(nr) + ". What is the capital of " + land + "?\n")
        feil_svar = random.sample(list(hovedsteder.values()), 3)
        alternativer = sorted(feil_svar + [hovedsteder[land]])
        quizfil.write("   ".join(alternativer) + "\n\n")
```

Stort sett er dette programmet satt sammen av ting vi har sett tidligere, men noe nytt har også dukket opp.

- I filen `cow.txt` er det en del mellomrom mellom feltene som vi må ta vekk. Det er nettopp dette `strip` gjør, fjerne mellomrom på begynnelsen og slutten av tekststrenger. Det gjør det for eksempel mulig å teste om et land har en hovedstad ved å teste lengden av hovedstad-strengen.
- For å unngå overskriftlinjen tester vi også på om felt 0 har lengde nøyaktig 2, siden dette skal være en 2-bokstavers kode som identifiserer landet.
- For å gjøre quizen litt enklere (særlig om man skal kose seg med den på hytta) vil vi også gi alternativer. Dette gjør vi igjen ved hjelp av `random`-pakken. Funksjonen `random.sample` plukker et gitt antall ting tilfeldig fra en liste. Vi lager alternativene våre ved å sette det riktige svaret sammen tre feil svar (her er det en liten *bug* i programmet vårt, vi er ikke helt sikre på at alle byene i `feil_svar` faktisk er feil). Ved å sortere svaralternativene skjuler vi hvilket svar som er det riktige.

- Funksjonen `join` slår sammen en liste av tekststrenger til en lengre tekststreng.

Dette siste eksempelet kan virke litt komplisert, men samtidig har vi sett hvordan det er bygd opp av mange små biter som hver for seg er ganske forståelige. Slik er det med alle programmer. Store programmer består i dag av tusener og til og med millioner av linjer med kode som det er umulig å ha komplett oversikt over. Men de er alle bygd opp av mindre byggeklosser som ligner de vi har sett her.