CSV Cheatsheet

Biblioteker

Alle eksempler vist er antaget at have følgende to imports:

- Dette importerer biblioteket brugt til at læse og skrive CSV filer

```
import csv
```

- Dette importerer biblioteket brugt til at skrubbe specialkarakterer fra data

```
import re
```

Opbygningen af main

Grunden til at vi vil have en main funktion er for to grunde:

- Det gør koden mere overskueligt, i den forstand at det er nemt at se hvor koden starter
- Hvis man vil gøre mere avancerede ting, såsom at lave et bibliotek som man vil importere senere, gør main funktionen at der ikke er kode der eksekveres så snart biblioteket er importeret

```
def main():
    with open('filename.csv') as csv_file:
        reader = csv.DictReader(csv_file)
        foo(reader)

if __name__ == '__main__':
    main()
```

Læsning og Printning af Alle Rækker

Alle rækker er af formen dictionary, så vi kan kalde på et specifikt værdi fra en række ved at kalde på dens header.

```
def foo(reader):
    for row in reader:
        print(row['id'])
```

Typer af Rækker og deres Indehold

Når man læser læseren en række ad gangen, består hver række af en dictionary.

```
# Eksempel af hvordan en række kunne se ud
row = {'id' : '42', 'name' : 'hello', 'description' : 'world'}
```

Bemærk at id'ets type er en tekststreng, hvilket har den betydning at dens opførsel i nogle situationer kan være uønsket (for eksempel giver `42` + 3 en fejl). For at forhindre dette, kan man tvinge typen om til en ny type, via casting.

```
# Cast fra string til integer (heltal)
print(int('42') + 3)
# Cast fra string ti float (decimaltal)
print(float('1.8') + 3)
```

Indskrivning af CSV-Filer

Det er vigtigt at inkludere 'w' i open, da dette fortæller programmet at vi gerne vil 'write' i filen. Derudover skal
man også, når man initialiserer filen, sige hvilken headers den nye fil skal have. Før man begynder at skrive til filen, skal man huske at køre funktionen 'writer.writeheader()' for at skrive headersne ind øverst i filen. Eksemplet nedenunder kopier alle headersne fra den læste fil.

```
def foo(reader):
    fields = reader.fieldnames
    with open('filename.csv', 'w') as new_file:
        writer = csv.DictWriter(new_file, fieldnames = fields)
        writer.writeheader()
        for row in reader:
            writer.writerow(row)
```

Man kan også indskrive flere rækker ind ad gangen, ved at smide dataen ind i en liste, og bruge funktionen writerows().

```
writer.writerows([{'id' : 42, 'name': 'Hello'},
                  {'id' : 1337, 'name' : 'world!'}])
```

Mulige Problemer og deres Løsninger

Skal Kun Bruge Ét Resultat

Man kan bruge kommandoen break til at afbryde løkken øjeblikkeligt

```
# Dette kodestykker printer rækkens navn hvis id er lig 42,
# og vil derefter terminere
for row in reader:
    if int(row['id']) == 42:
        print(row['name'])
        break
```

Manglende Data

Nogle gange er ikke alt dataen udfyldt - det er derfor vigtigt at håndtere disse situationer. For dette kan man lave en test på dataen, og bruge kommandoen continue til at ignorere rækken hvis den er tom.

```
result = []
for row in reader:
    if row['name'] == '':
        continue
    if row['name'].lower() == 'hot dog':
        result.append(row['id'])
```

Datasættet er Anderledes Formatteret

Hvis datasættet bruger noget andet en kommaer til at adskille dataen, kan man specificere hvilken karakter bruges til at adskille dataen.

```
# Dette åbner en fil som bruger punktum til at adskille dataen
with open('filename.csv') as input_file:
    reader = csv.DictReader(input_file, delimiter='.')
```

Dataen Indeholder Specialtegn

Hvis datasættet indeholder specialtegn, for eksempel kommaer, kan nogle resultater forsvinde grundet specialtegnene, (for eksempel, hvis man søger efter order mustard, kan man risikere sammenligningen 'mustard,' == 'mustard').

For at fikse dette, vil vi bruge sort magi til at fjerne alle specialtegn (til dem der er interesseret, hedder det regex).

```
for row in reader:
    temp = re.sub(r'[^\w]', '', row['name'])
```

Bemærk dog at dette også fjerner mellemrum, så hvis dataen består af en sætning, skal man i stedet for skrubbe hvert ord.

```
for row in reader:
    sentence = row['description']
    for word in sentence:
        new_word = re.sub(r'[^\w]', '', row['description'])
```