Textfiler

Introduktion

Variabler som används i ett program är lagrade i datorns s.k. *arbetsminne*. Variablerna och deras data (värden) existerar bara så länge programmet körs (exekveras).

Program behöver kunna lagra data permanent, exvis på datorns s.k. hårddisk. För detta ändamål används ***filer***.

En fil är en godtyckligt lång följd av s.k bytes.

Om filen innehåller tecken (s.k. Ascii-koder) kallas den ***textfil***. Textfiler kan läsas och innehållet vara begripligt med ett enkelt program typ Notepad.

En annan typ av fil är ***binärfil***, det kan vara t.ex. program (exe-filer) eller Word-filer (doc-filer).

Indata/Utdata

Program kan behöva

- Läsa ***indata*** från fil
- Spara resultat (skriva ***utdata***) på fil

Exempel på program:

- Webbläsaren (indata HTML-fil, en textfil)
- Notepad (indata/utdata textfil)
- Word (indata/utdata word-fil, ej textfil)
- Windows mediaplayer (indata ljudfil, ej textfil)
- Spelprogram

Ex. Spelprogram (t.ex. The Sims)

Programmet börjar med att läsa in data från en fil (indata), dvs ”speldata”, som lagras i programmets variabler.

Programmet kan därmed visualisera speldata med grafik.

Allt vi ser på skärmen är skapat från variablers värden. Massor av variabler.

Programmet kan nu interagera med användaren, dvs ”man kan spela”

När användaren väljer att avsluta programmet sparas alla aktuella speldata på en fil, dvs programmets variabelvärden skrivs på en fil.

Program kan kommunicera via filer

Ett program A skapar utdata → skriver på en fil.

Ett annat program B behöver indata → läser en fil som programmet A skapade.

Exempel programmet Thonny:

I dess editor skriver vi och sparar ett pythonprogram, dvs en pythonfil (utdata är en textfil med pythonsatser)

Run

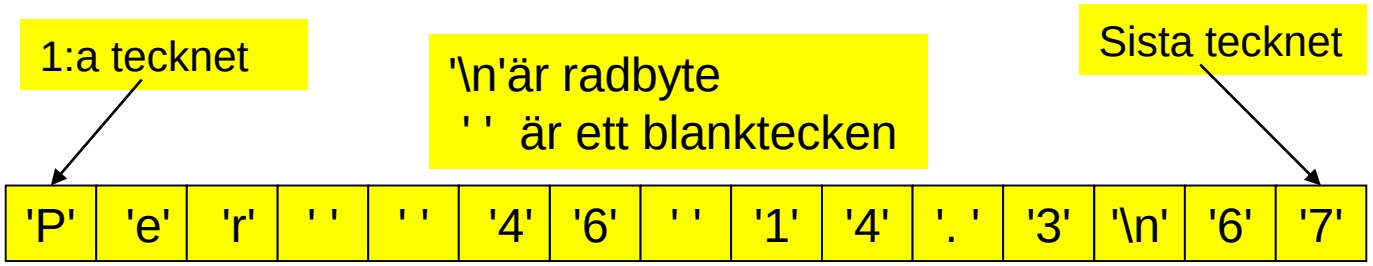
pythoninterpretatorn behöver pythonfilen (`.py`) som indata för att exekvera (köra) programmet.

Textfil = sekvens av tecken

Så här kan en textfil se ut om vi öppnat filen i Notepad. Filen ser ut att bestå av tecken som bildar ord och rader.

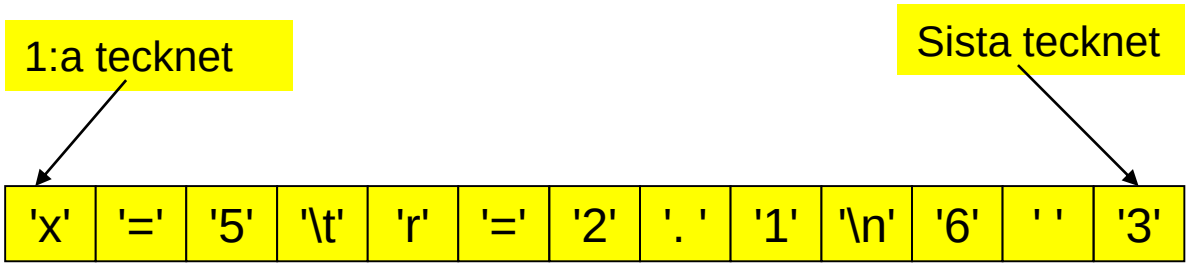
```
Per  46 14.3
67
```

Filen är egentligen en sekvens av tecken, där vissa tecken syns. Andra tecken är specialtecken som TAB och radbyte som t.ex. Notepad använder för att formattera texten, dvs så det ser bra ut på skärmen. Sekvensen av tecken är lagrad permanent i minnet (t.ex. hårddisken) som så här:



Denna sekvens av tecken kan betraktas som en **ström** av tecken.

Om textfilen består av följande **ström** av 13 st tecken:



'\t' är TAB
'\n' är radbyte
' ' är ett blanktecken

Om vi öppnar textfilen i Notepad så ser den ut så här

```
x=5     r=2.1
6 3
```

Notepad har formatterat filen.

Öppna och stänga filer

Vi skall nu se hur kan skapa egna strömmar och koppla dem till filer och att man läsa och skriva till dessa strömmar. Öppna en fil för läsning:

```
f = open('data.txt', 'r')
```

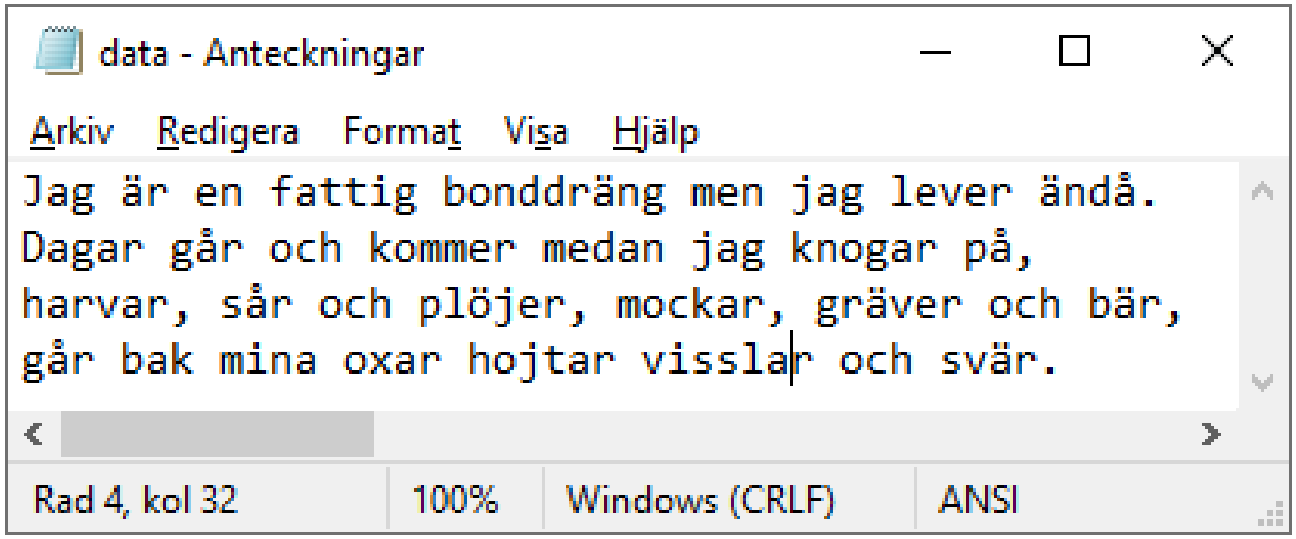
- Funktionen `open` skapar (returnerar) en referens (`f`) till ett filobjekt och `f` används för att läsa filen `data.txt`
- `'r'` betyder vi öppnar filen för läsning

När vi kör programmet fås en felutskrift:

```
No such file or directory: 'data.txt'
```

Det beror på att filen `data.txt` ej finns i den mapp där pythonfilen (programmet) finns.

Vi skapar en fil i Notepad, som ser ut så här:



Spela upp

Döper filen till `data.txt` och spar den i samma mapp som python-filen

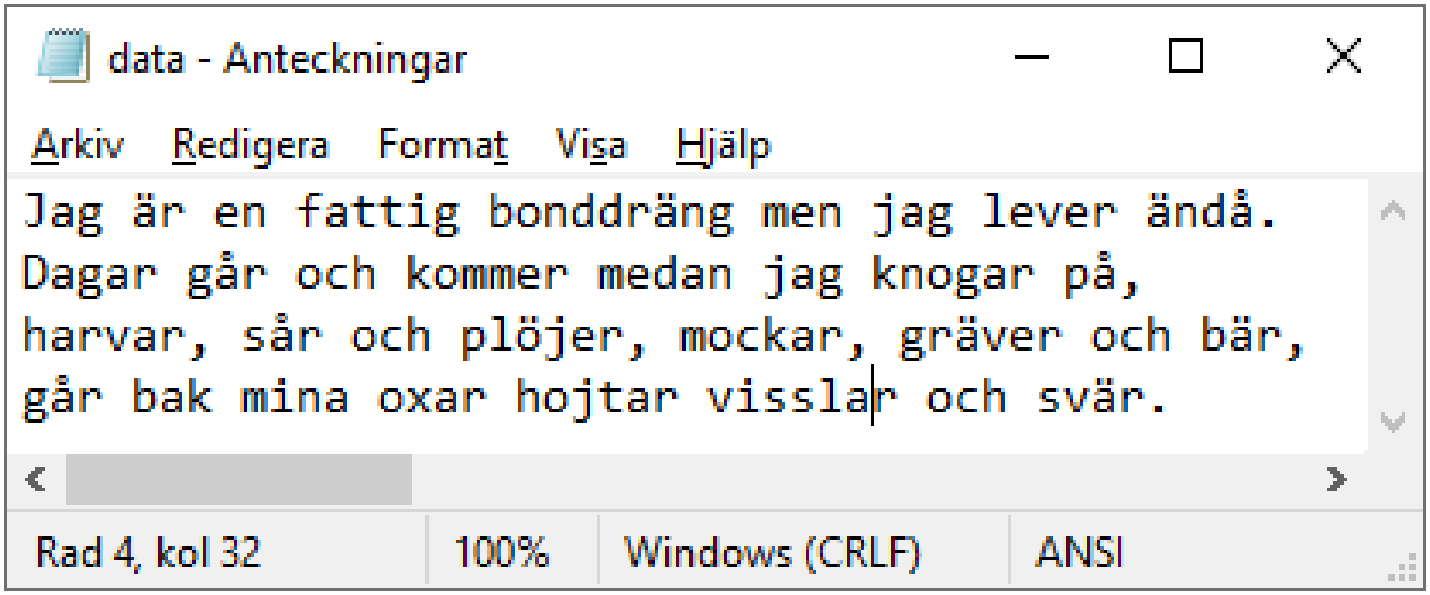
Nu går det att köra programmet utan det tidigare felet.

Men behöver lägga till satser för att läsa filen →

Vi skall läsa filen och skriva ut innehållet på skärmen:

```
# filEx1.py
f = open('data.txt', 'r') # öppna filen
for rad in f: # för varje rad i filen
    print(rad, end='')
f.close()
```

- Börjar med att öppna filen för läsning. En ”filpekare” är nu högst upp i filen och ”pekar” på det första tecknet i den första raden i filen.
- Loopen itererar så många ggr som det finns rader i filen, dvs tills filen är slut.
- Variabeln `rad`, tilldelas den sträng av text som är aktuell rad i filen. Första varvet i loopen består strängen av:
  `Jag är en fattig bonddräng men jag lever ändå.\n`
- `end=''` för att slippa dubbla radbyten
- Funktionen `close` stänger filen efter loopen.



Kör programmet igen, vilket ger följande utskrift:

```
Jag är en fattig bonddräng men jag lever ändå.
Dagar går och kommer medan jag knogar på,
harvar, sår och plöjer, mockar, gräver och bär,
går bak mina oxar hojtar visslar och svär.
```

```
# filEx2.py
f = open('data.txt', 'r')
rader = 0
tecken = 0
for rad in f:
    rader += 1            # Radräknare
    tecken += len(rad)    # Teckenräknare
f.close()
print('#rader, #tecken = ', rader, tecken)
```

- För varje rad i filen läser programmet raden från filen och placerar den inlästa raden (texten, dvs strängen) i variabeln `rad`, som därmed en vanlig strängvariabel.
- for-satsen innebär en iteration för varje rad i filen.

Programkörning:

```
#rader, #tecken =  4   179
```

Läsa teckenvis med `read`

```
f = open('data.txt', 'r')
# Läs filens 5 första tecken
s = f.read(5)  # s är en sträng med 5 tecken
print(s)
```

```
Jag ä
```

Läsa radvis med `readline`

```
f = open('data.txt', 'r')
f.readline() # Läs förbi första raden
rad = f.readline() # Läs och spara nästa rad
print(rad) # rad inkluderar '\n')
```

```
Dagar går och kommer medan jag knogar på,
```

Läsa heltal från en fil, `heltal.txt`, lagra dem i en lista

Antag att man vet att filen innehåller ett tal per rad

```
# filEx3.py
f = open('heltal.txt', 'r')
# Spara talen i en lista
talList = []
for rad in f: # För varje rad i filen
    talList.append(int(rad)) #Omv tecken->heltal
f.close()
print(talList)
```

Ger:

```
[413, 410, 405, 43, 393, 394, 390, 417, 39, 404]
```

Läsa heltal från en fil, heltal2.txt, lagra i en lista

Antag antalet heltal per rad är okänt, mellanslag mellan talen

```
# filEx4.py
f = open('heltal.txt', 'r')
talList = [] # Spara talen i en lista
for rad in f: # Iterera över alla rader
    # Splitta orden i raden till en lista
    ordList = rad.split() # mellanslag
    # Gör om str-listan till en int-lista
    # Lägg ihop med talList
    talList = talList +  \
              [int(ord) for ord in ordList]
f.close()
print(talList)
```

\ satsen fortsätter nästa rad

```
[413, 410, 405, 43, 393, 394, 390, 417, 39, 404]
```

Problem med de svenska tecknen åäöÅÄÖ

Tecknen i textfiler kan vara kodade enligt olika principer, exvis `utf-8`, `ansi`. Detta kan ställa till problem.

Läs mer om hur detta kan lösas i minilektionen Läsa och skriva på kurswebben.

Fundera på: Vi har en fil `register.txt` med följande innehåll:

```
Medlemmmar 2020-10-01
kim        23
saga       18
juno       29
alex       27
robin      14
```

Första raden i filen är en rubrik.

Kolumnerna betyder namn och ålder.

Om vi skall beräkna värden på åldern: min, max och median.

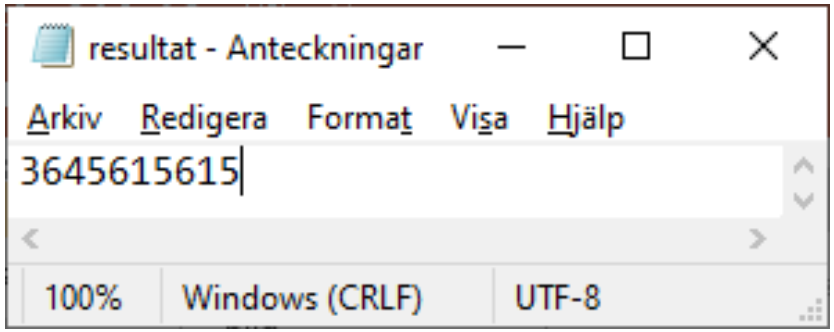
Hur kan filen lämpligen läsas?

Skriva på fil.

Ex. Scriptet `filEx5.py` skriver ut 10 st tärningsvärden

```
f = open('resultat.txt', 'w') # Skapa en fil
for i in range(1,11):
    t = random.randint(1,6)
    f.write(str(t))
    # alt: print(t, end='', file=f)
f.close()
```

Körning skapar en fil `resultat.txt` i samma mapp som pythonfilen. Vi kikar på filen mha Notepad

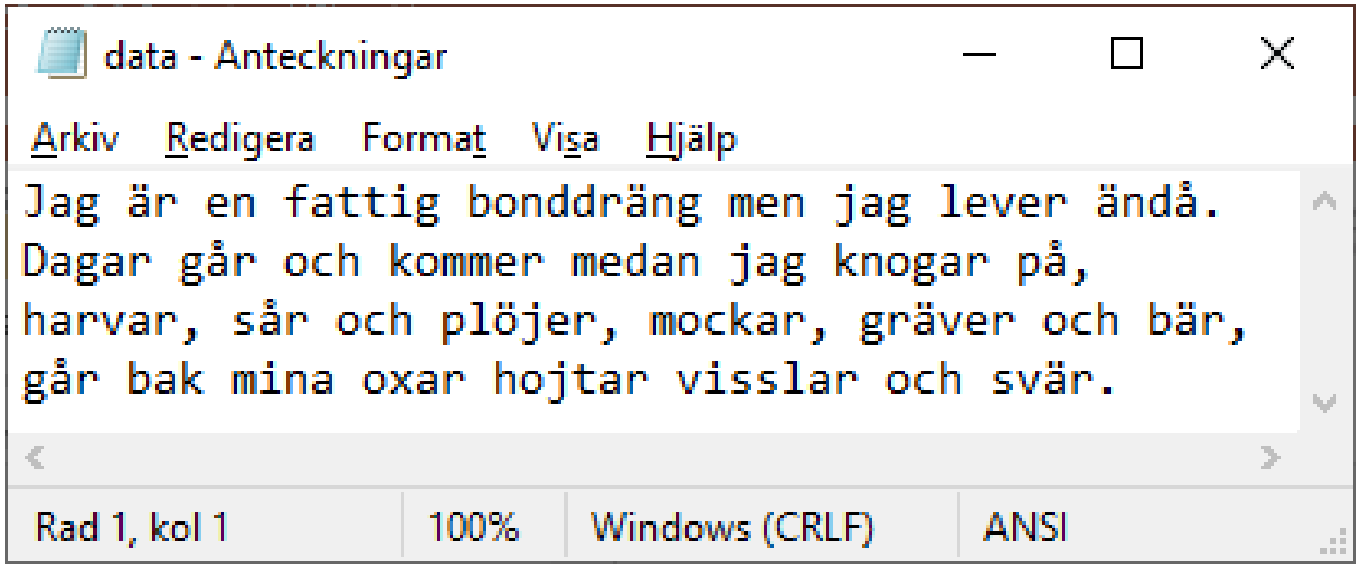


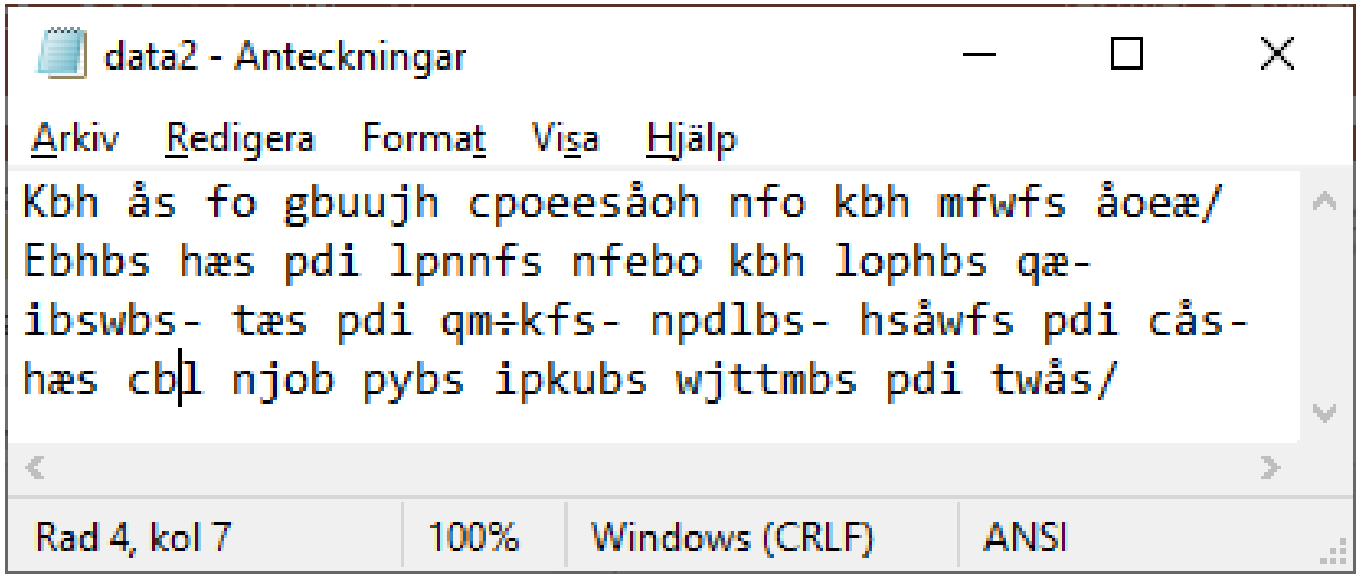
Metoden `write` skriver ut en sträng. Bättre utskrift med:

```
f.write(str(t) + ' ')
```

Ex. Scriptet filEx6.py, ”Krypterar” en textfil

```
fin = open('data.txt', 'r') #Fil att läsa från
fut = open('data2.txt', 'w')#Fil att skriva på
special = [' ', '\t', '\n']
for rad in fin:
    s = ''
    for ch in rad:
        if ch in special:
            s = s + ch
        else:   # addera ett i ascii-koden
            s = s + chr(ord(ch)+1)
    fut.write(s)
fut.close()
fin.close()
```





Kontrollera i Ascii-tabellen, å,ä,ö är knepiga

with, ett alternativ till open.

Exvis

```
with open('data.txt', 'r') as f:
    for rad in f:
        …
```

Filen stängs per auto

Allmänt: **`f = open(filnamn, mode)`**

`mode` anger hur filen skall användas:

`r`	läsning
`w`	Skrivas, om den finns skrivs den över, annars skapas ny
`x`	Skrivas, får ej finnas tidigare
`a`	som `w`, men lägger till från slutet om filen redan finns

Sammanfattning

`f.read(n)`
Läser `n` st tecken från filen `f`. Om `n` är negativ, läses hela filen. Returnerar `None` vid filslut.

`f.readline()`
Läser en rad i filen. Returnerar `None` vid filslut.

`for rad in f:`
Läser en rad i taget till rad. Som upprepade `readline`

`f.readlines()`
Läser alla raderna i filen. Ger en lista med alla raderna, dvs elementen är resp rad.

`f.write(s)`
Skriver ut `s` till filen `f`. Returnerar antal skrivna tecken.

`print(…, file=f).`
Som vanliga `print` på skärmen, men skriver till filen `f`.