input2

August 13, 2023

1 Läsa och skriva filer

Behandlar filer av typen .txt (i regel text) respektive .csv (i regel tal).

1.1 open(), close(), with open() as

Ibland är datamängden stor och arbetet med att skriva in den i listor för att rita en graf är stor. Python kan läsa olika filer med data. På samma sätt kan resultatet av beräkningar i python behöva skrivas ut i en fil i stället för att bara flimra förbi på skärmen.

Vi ska arbeta med 2 olika filtyper, de som har typen .txt och de som har typen .csv (anges ibland i ordbehandlingsprogram som 'text csv'). Vi börjar med filtypen .txt. Ingen extra modul behöver laddas för att hantera filer av denna typ.

Processen kan förstås på följande sätt:

1. Filer måste öppnas innan man kan göra något med dem, vi får ett `open()` kommando. Precis som du måste öppna en bok för att läsa i den.
2. Vi måste naturligtvis ange filen som ska öppnas: `open('data.txt')`. Filen ska ha ändelsen txt, en text-fil. Precis som på bibliotek måste du ange vilken bok du ska läsa.
3. Vi måste ange vad vi ska göra: skriva, läsa, lägga till osv.:`open('data.txt', 'r')` för read. Du måste ange om du ska läsa eller skriva. Det är en säkerhetsgrej. Om du bara ska läsa och öppnar för att läsa kommer python hindra dig från att skriva av misstag.

Några exempel på varianter på read och write

- w innebär att filen öppnas för att skrivas i. Om den redan finns töms den och skrivs på nytt. Om filen inte finns skapas den.
- w+ är både läsa och skriva, annars som w.
- a innebär append. Filen öppnas för skrivning, filen töms inte utan man skriver sist i filen. Filen skapas om den inte redan finns.

`f = open("datafil1.txt","w")` öppnar filen datafil1.txt och förbereder den för skrivning. Filen får också ett enklare namn, den kallas bara f.

Det finns ett kommando som gör både open och close, det heter `with open as`. Det används som ett alternativ i exemplet nedan.

[2]:
```
# Öppnar filen för att skriva och ger den ett enklare namn f.
# Vi utgår från att den inte finns.
f = open("datafil1.txt","w")

# Skriver en massa rader till filen som nu har det korta namnet f
for i in range(10):
    text = "Detta är rad " + str(i+1) + "\n"
    f.write(text)  # Varje gång vi skriver blir det en ny rad

# Vi stänger filen när vi är färdiga
f.close()




# För att leka lite med filen så öppnar vi den igen. a står för append
f = open("datafil1.txt", "a")
f.write("Nu var det slut")
f.close()



# Vi skriver ut filen. Det görs genom att öppna den för läsning och sedan läsa.␣
 ↪r read.
# Vi använder denna gång kommandot 'with open() as' som stänger automatiskt.
with open('datafil1.txt', 'r') as f:
    print(f.read())
# filen stängs denna gång automatiskt
```

```
Detta är rad 1
Detta är rad 2
Detta är rad 3
Detta är rad 4
Detta är rad 5
Detta är rad 6
Detta är rad 7
Detta är rad 8
Detta är rad 9
Detta är rad 10
Nu var det slut
```

Ändra i föregående kodblock som var

```
# För att leka lite med filen så öppnar vi den igen. a står för append
f = open("datafil1.txt", "a")
f.write("Nu var det slut")
f.close()
```

så att det står “w” i stället för “a”. Då kommer filen att skrivas över och endast “Nu var det slut”

kommer att skrivas ut.

Observera således att en fil måste öppnas för 'read' för att kunna skrivas ut.

1.2 Läsa hela eller läsa rader

Koden demonstrerar användningen av `read()` respektive `readlines()` för filtypen .txt.

Filen som ska läsas ska finnas. Data i en fil med ändelsen .txt ligger separat. Du kan skapa en sådan fil genom att skriva in rader i en fil med tex. Microsofts Word eller LibreOffice Writer eller enkla texthanterare. Om du skriver i MS Word eller LO Writer så måste filen sparas som .txt!

[7]:
```
# Ska läsa en fil som finns
f = open("datafil2.txt", "r")

# Läs hela filen och skriv ut den
content = f.read()
print("Hela filen direkt\n", content)

f.close()



# Vi öppnar filen igen, läser då automatiskt från början

f = open("datafil2.txt", "r")

# Lägg märke till att filen f1 är en lista med varje rad som ett element i␣
 ↪listan.
# Kommandot är också tydligt readlines().
f1=f.readlines()

print("f1 är av datatypen", type(f1))
print("rad 1:", f1[0])
print("rad 2:", f1[1])
print("rad 3:", f1[2])
print("rad 4:", f1[3])

f.close()
```

```
Hela filen direkt
 Första raden
Sedan kommer andra raden
Och näst sist tredje raden
Slut

f1 är av datatypen <class 'list'>
rad 1: Första raden
```

```
rad 2: Sedan kommer andra raden

rad 3: Och näst sist tredje raden

rad 4: Slut
```

Vi skulle kunna säga att filen är en lista: f1=[“Första raden”, “Sedan kommer andra raden”, ”Och näst…]

1.3 Filer med ändelsen .csv

Vi ska nu öppna och läsa en fil med ändelsen (extension) csv (file extension översätts till svenska som filsuffix eller filändelse). csv betyder comma separated values. En fil där data separeras med ett kommatecken (,). SCB benämner dem kommaavgränsad fil. Talen använder decimalpunkt så betydelserna hålls isär. Detta är en sak man måste se upp med då man importerar från olika program. https://fileinfo.com/extension/csv

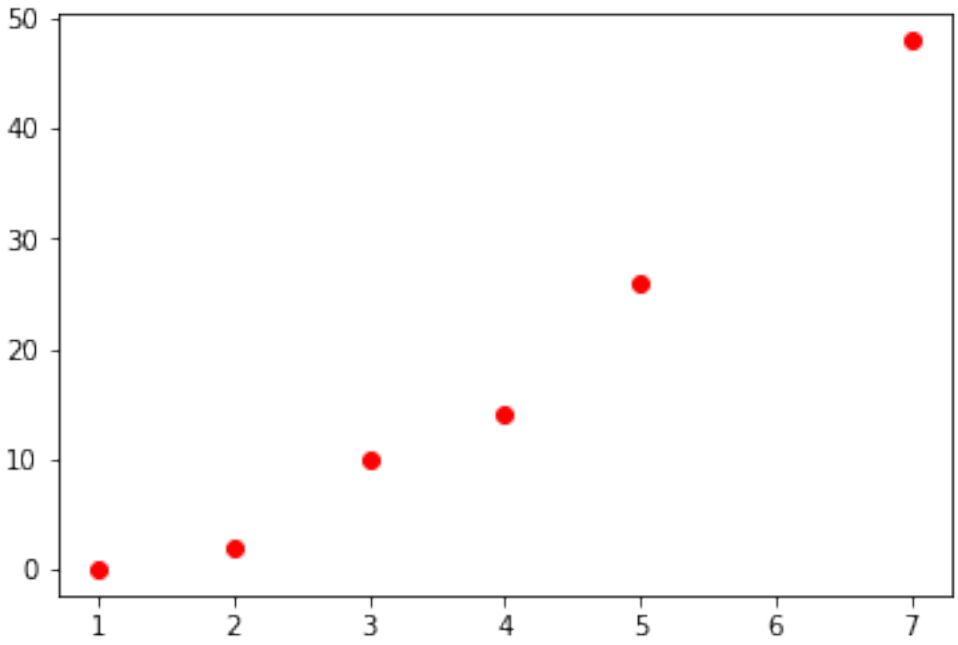
För att kunna läsa denna typ av filer måste en modul importeras som heter just csv.

Först en bild på den datafil som vi ska läsa. På replit.com skapar du en ny fil och skriver data direkt in. På 3:e raden ligger 2 tal, talet 3 och talet 9. De är separerade med ett kommatecken. Kommatecknet är inte ett decimaltecken i detta sammanhang. Radnummret ingår inte i filen.

[1]:
```
# Vi behöver importera en modul för att läsa .csv. En fil exporterad som 'text␣
 ↪csv'
# från LibreOffice läses.

import csv

with open('testdata.csv','r') as f:
  data = csv.reader(f)  # Obs reader för csv, ej read!
  print("radutskrift")
  for row in data:  # Behöver ej heta row, men det är suggestivt
        print(row)  # Observera att elementen i listan är strängar
# Automatiskt close i och med vi använt with. Så nästa gång vi använder with␣
 ↪börjar vi från början
# i filen.



import matplotlib.pyplot as plt
import numpy as np


# Nu ska vi läsa in dessa som x och y listor.
# Med rader ungefär som ovan läses varje rad som en lista med två 'tal'; de är␣
 ↪lagrade som strängar.
```

```
# Första elementet i raden är x och det andra y.

x=[]; y=[]  # Två tomma listor att starta med

with open('testdata.csv','r') as f:
  data = csv.reader(f)

  for row in data:
        x.append(float(row[0]))  # Omvandlar strängarna till float och lägger i
 ↪en annan lista
        y.append(float(row[1]))

print("x och y i listor")
print("x",x)
print("y",y)


plt.plot(x, y, 'ro')
#red circle

plt.savefig('plot.png')
plt.show()
```

```
radutskrift
['1', '0']
['2', '2']
['3', '10']
['4', '14']
['5', '26']
['7', '48']
x och y i listor
x [1.0, 2.0, 3.0, 4.0, 5.0, 7.0]
y [0.0, 2.0, 10.0, 14.0, 26.0, 48.0]
```



Vi tittar i detalj på två centrala delar av koden:

Del 1

```
with open('testdata.csv','r') as f:
  data = csv.reader(f)
  print("radutskrift")
  for row in data:  # Behöver ej heta row, men det är suggestivt
        print(row)
```

Metoden reader stöder iteration, därför kan vi använda det vi fått från reader, data, nästan som en lista, det är dock inte en lista.

`for row in data:` behöver variabeln inte heta `row`, den kan heta vad som helst men det är suggestivt med row.

Del 2

```
with open('testdata.csv','r') as f:
 data = csv.reader(f)
  for row in data:
        x.append(float(row[0]))
        y.append(float(row[1]))
```

Till stor del samma som föregående, row går igenom data rad för rad. Men varje rad består av en sträng, sedan kommatecken och sedan en sträng till. Den första strängen på raden har index 0, den andra index 1, den tredje index 2 osv. Kommatecknet ingår inte i strängarna.

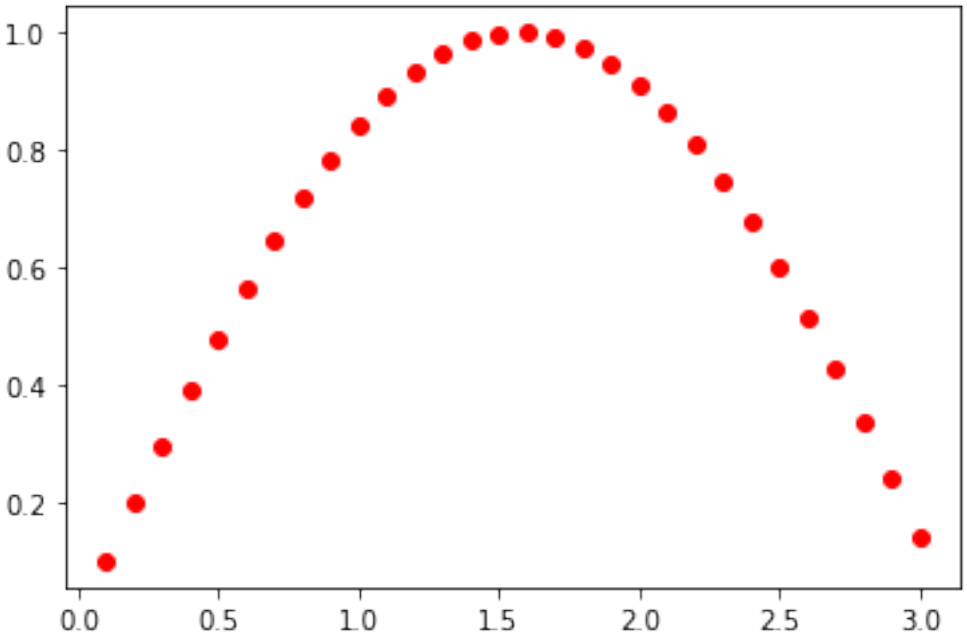
`row[0]` är den första strängen på den aktuella raden. row itererar sig fram genom raderna.

1.4 Rita med data från kalkylark

Vi kombinerar med en större fil för att rita en graf.

Vi använder att metoden reader för csv filer stöder iteration. Filen testdata2.csv är skapad i LO Calc som är ett program för kalkylark. Se till att decimaltecknet är en punkt, inte kommatecken. File->Save as-> Text csv.

[3]:
```
import csv

import matplotlib.pyplot as plt
import numpy as np


x=[]; y=[]
with open('testdata2.csv','r') as f:
  data = csv.reader(f)
  for row in data:
        x.append(float(row[0]))
        y.append(float(row[1]))

#print("x",x)
#print("y",y)

#plt.axis([0,3,0,1])

plt.plot(x, y, 'ro')
#red circle

plt.savefig('plot.png')
plt.show()
```



1.5 Sammanfattning

Du måste öppna en fil innan du ska använda den, du måste stänga den när du är klar. Du bör också ange vad du ska göra: läsa, skriva. Vi har behandlat 2 filformat: txt och csv. Vi behandlar inte kommersiella format. Alla operativsystem kan spara i txt och csv.

.txt 1. open() 2. close() 3. `with open('datafil1.txt', 'r') as f:` automatisk close(); f är nu kortnamnet för filen du ska arbeta med.
* r, w, w+ 4. f.read när du ska läsa, f.write när du ska skriva 5. f.readlines för att läsa en rad åt gången
6. Rader kan adresseras med index

.csv 1. Kalkylark. Se upp för användningen av decimaltecken; kommatecken eller punkt. 2. Typexempel

```
import csv
with open('testdata.csv','r') as f:
    data = csv.reader(f)
    print("radutskrift")
    for row in data:
        print(row)
```

3. Typexempel inför plot; skapa 2 listor som sedan kan ritas

```
x=[]; y=[] #Två tomma att starta med

with open('testdata.csv','r') as f:
  data = csv.reader(f)
```

```
for row in data:
      x.append(float(row[0]))
      y.append(float(row[1]))
```

1.6 Uppgifter

Uppgift 1

Ladda ner filen, öppna den, läs den och plotta den: uppgift1_fil.csv

Lösningsförslag

Uppgift 2

Hämta en kommaavgränsad fil från statistiska centralbyrån eller ladda ner uppgift2_fil.csv (hämtad från scb 220618) och plotta datamängden. Filen innehåller "Eltillförsel netto, GWh efter månad och produktionsslag". Kommandot `next(data)` gör att en rad hoppas över. Detta kan vara bra då första raden ofta används för rubrik. Koden nedan illustrerar användningen av next().

```
with open('filnam','r') as f:
  data = csv.reader(f)
  next(data)  # ! Hoppa en rad
  for row in data:
        x.append(...)
        y.append(...)
```

Lösningsförslag

[ ]: