Per Sedholm
DD1320 (tilda11)
2013-09-11

Unicode och UTF-8 i Python 2

Unicode i Python-kod

För att allt ska fungera måste man använda både rätt teckenkodning på koden, och rätt datatyp i Python.

Källkodens teckenkodning bestäms av programmet man skriver kod i. De flesta program (inkl. Python och Idle) förstår en kommentar i början av filen på formatet

```
# -*- coding: utf-8 -*-
# -*- coding: iso-8859-1 -*-
# -*- coding: latin-1 -*-
```

och kommer då både att visa bokstäverna korrekt och spara rätt data på fil.

Python har två olika strängtyper, `str` och `unicode`. En vanlig `str`-sträng innehåller bara en sekvens av tecken, Python vet inte ifall de kodar korrekta bokstäver eller inte. Ifall filen är sparad som *Latin-1* kommer strängen `"åäö"` att innehålla tre element, men ifall den är sparad som *UTF-8* kommer den att ha sex element:

```
print len("åäö")          # -*- coding: iso-8859-1 -*-
# 3
```

```
print len("åäö")          # -*- coding: utf-8 -*-
# 6
```

Skriver man ut strängen, så kommer åtminstone den ena att visas som konstiga tecken.

Använder man unicode-strängar, `u"åäö"`, så vet Python hur många bokstäver strängen innehåller (3 st), och de kommer också att visas korrekt vid utskrift.

Unicode-strängar kan skrivas på flera sätt:

```
# Om filens teckenkodning stämmer
data = u"räksmörgås"
data = unicode("räksmörgås", "UTF-8") ## använd filens teckenkodning

# Om man kan bokstävernas code point
data = u"r\u00e4ksm\u00f6rg\u00e5s"

# Om man vet hur teckenkodningen är definierad
data = unicode("r\xc3\xa4ksm\xc3\xb6rg\xc3\xa5s", "UTF-8")
data = unicode("r\xe4ksm\xf6rg\xe5s", "ISO-8859-1")

# Som namngivna bokstäver (ges av 'unicodedata.name(u"ä")')
data = u"r\N{LATIN SMALL LETTER A WITH DIAERESIS}ksm\N{LATIN SMALL LETTER O
WITH DIAERESIS}rg\N{LATIN SMALL LETTER A WITH RING ABOVE}s"
```

Det finns även metoder i sträng-klasserna för att konvertera till (`encode`) och från (`decode`) olika teckenkodningar:

```
u"Достоевский".encode("iso-8859-5")
    →  "\xb4\xde\xe1\xe2\xde\xd5\xd2\xe1\xda\xd8\xd9"

"\xc5\xeb\xeb\xe7\xed\xe9\xea\xdc".decode("iso-8859-7")
    →  "Ελληνικά"
```

Lokaler i Python-kod

För att sortera svensk text korrekt, så måste svenska sorterings-regler användas. För detta sätter man lokalen till *sv_SE.UTF-8* (eller motsvarande för *Latin-1*). Man kan antingen sätta endast sorterings-regler (*LC_COLLATE*), eller alla lokal-regler (*LC_ALL*).

Dessutom måste man skicka med `key` (ett namngivet argument) till sorterings-funktionen. Modulen *locale* innehåller en lämplig funktion, `strxfrm`. Notera att man *inte* får korrekt engelsk sortering när inga lokaler används.

```
# -*- coding: utf-8 -*-

lst = ['Ärlig', 'Caesar', 'Adam', 'Bertil', 'Östen', 'Åke']
lst.sort()
print ", ".join(lst)
# Adam, Bertil, Caesar, Ärlig, Åke, Östen
#                       ^^^^^^^^^^

import locale
locale.setlocale(locale.LC_ALL, 'sv_SE.UTF-8') ## alt. LC_COLLATE
# returns 'sv_SE.UTF-8'
lst = ['Ärlig', 'Caesar', 'Adam', 'Bertil', 'Östen', 'Åke']
# lst.sort(cmp=locale.strcoll) ## Old-style sorting (Python <= 2.4)
lst.sort(key=locale.strxfrm)
print ", ".join(lst)
# Adam, Bertil, Caesar, Åke, Ärlig, Östen

import locale
locale.setlocale(locale.LC_ALL, 'en_US.UTF-8')
lst = ['Ärlig', 'Caesar', 'Adam', 'Bertil', 'Östen', 'Åke']
lst.sort(key=locale.strxfrm)
print ", ".join(lst)
# Adam, Åke, Ärlig, Bertil, Caesar, Östen
```

Bakgrundsinformation

Teckenkodningar

Tecken kan sparas på olika sätt. Ett vanligt sätt är ASCII[1], som t ex säger att ”A” ska representeras av en 8 bitars byte med värdet 65, ”B” av 66, osv. Men ASCII definierar endast amerikanska tecken. För att kunna spara tecken som

á à ä â æ ½ ¼ © ® ...

har man använt olika utökningar, exempelvis:

CP-437 (CodePage 437) eller **CP-850** för MS-DOS

CP-1252 för Windows

Mac OS Roman för Mac OS innan OS X

Alla hade olika sätt att koda gemensamma bokstäver som ÅÄÖ, alla hade några bokstäver som inte fanns i de andra. För de teckenkodningar (för kyrilliska, grekiska, etc) hade ännu färre gemensamma bokstäver.

Standardisering gjordes med ISO/IEC 8859, och idag används teckenkodningar som ISO-8859-1 (Latin-1) eller ISO-8859-15 (Latin-15)[2] för att spara västerländska bokstäver som 8-bitars tecken. Andra (ISO-8859-5) sparar kyrilliska eller (ISO-8859-7) grekiska bokstäver som 8-bitars tecken. De är alla en del av Unicode[3]. Ett Unicode-tecken kallas en *code point*, och skrivs `U+XXXX` med 4 hexadecimala siffror. Vissa har i stort sett samma utseende, t ex ser både U+2044 (”fraction slash”) och U+2215 (”division slash”) ut som U+002F (”solidus”): /

För att kunna använda fler än de 256 tecken som ryms inom 8 bitar sparar man med kodningen **UTF-8**[4]. Den använder mellan 1 och 6 byte för att spara varje tecken. (I praktiken sparas alla ASCII-tecken med 1 byte, och alla Latin-1–tecken med 2.)

Det datorprogram som läser filen måste veta vilken teckenkodning som används. Utan det är det omöjligt att veta ifall någon skrivit ett ”Ö” kodat i UTF-8 eller ”Ã¶” (versalt A med tilde följt av tecknet alinea, *pilcrow sign*). Bägge saker representeras av exakt samma binära data.

Lokaler

Teckenkodning ställs (ofta) in som en del av en *lokal*. Lokaler berättar vilket språk, format o dyl som ska användas. Exempel är:

`sv_SE.UTF-8` svenska, Svensk, kodat med UTF-8

`en_AU.ISO-8859-1` engelska, Australisk, kodat med Latin-1

[1] *American Standard Code for Information Interchange*

[2] Skillnaden är Euro-tecken, Œ-ligaturer, och några till

[3] Närmare bestämt utgör de olika *block* av ett av Unicodes *planes*, kallat *Basic Multilingual Plane*, som innehåller 65 536 av Unicodes *code points* (U+0000 t o m U+FFFF). Latin-1 innehåller blocken ”Basic Latin” (U+0000 – U+007F) och ”Latin-1 Supplement” (U+0080 – U+00FF).

[4] Unicode Transformation Format – 8-bit

För att byta sätter man en av de miljövariabler som definierar lokaler:

```
$ LC_TIME=en_US.UTF-8 date
Mon Sep 6 12:16:50 CEST 2010

$ LC_TIME=sv_SE.UTF-8 date
mån 6 sep 2010 12.16.54 CEST

$ LC_TIME=sv_SE.ISO-8859-1 date
m?n 6 sep 2010 12.17.09 CEST
```

Man kan i princip säga att alla datum ska använda *da_FO.KOI8-R* (Färöisk danska, en kyrillisk teckenkodning) men sorteras som *sv_FI.UTF-8* (finlandssvenska). I praktiken beror det på vilka lokaler som är installerade på systemet[5]. Man kan också (liksom ovan) se att det program som visar texten måste förstå teckenkodningen, annars visas oftast något konstigt tecken.

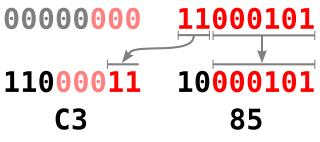
Det finns givetvis program som *iconv*, som kan konvertera filer.

Detaljerad beskrivning av UTF-8

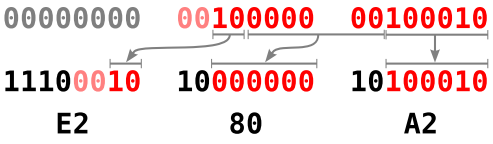
UTF-8 är konstruerad så att alla ASCII-tecken (mellan 0 och 127) kan sparas med en byte. Binärt börjar dessa siffror med en 0:a. För alla andra inleds UTF-8–tecknet första byte med så många 1:or som antalet byte i hela tecknet, följt av en 0:a. För alla tecken i Latin-1 betyder det två ettor, en nolla.

Code point	Byte 1	Byte 2	Byte 3	bitar till tecknet
U+0000 – U+007F	`0xxxxxxx`			7
U+0080 – U+07FF	`110xxxxx`	`10xxxxxx`		11
U+0800 – U+FFFF	`1110xxxx`	`10xxxxxx`	`10xxxxxx`	16

Binärt ser det ut som så för tecken med två resp. tre byte:



$Å = 197_{10} = C5_{16}$
$= 11000101_2$
$\rightarrow C3\ 85_{16}$



- $= \backslash N\{bullet\} = 2022_{16}$
$= 00100000\ 00100010_2$
$\rightarrow E2\ 80\ A2_{16}$

Referenser (Wikipedia)

- Latin-1
- UTF-8
- Unicode
- Om lokaler
- Mojibake (när det blir fel)

[5] Ifall man inte säger något annat använder många program lokalen *C*, dvs ”så som programspråket C skulle göra ifall lokaler inte används”.