Programiranje 2
Podatki

Vladimir Batagelj

Univerza v Ljubljani, FMF

April/maj 2013/2012

Kazalo

Datoteke

Računalniki so stroji za obdelavo podatkov. Podatki so na računalnikih spravljeni v *datotekah*. Datoteke so v bistvu zaporedja bitov. Ena osnovnih nalog operacijskega sistema je podpora dela z datotekami. Delimo jih na: *znakovne* (`TXT`), *prevedeni programi* (`EXE`), *sistemske* in *ostale* – dogovori o zapisu (programi, slike, zvok). Pri predelavi podatkov v računalniško obliko poznamo dva osnovna pristopa

- *posnetki* – vernost; omejena natančnost; veliko prostora – stiskanje;
- *opisi* – uporabljajo pravilnosti v podatkih; manj prostora; omejena uporabnost; natančnost omejena z zmogljivostjo izhodne napave.

Spletne storitve omogočajo prenose datotek in sporočil med računalniki. Prve storitve: prenos datotek (FTP), delo na drugem računalniku (Telnet), izmenjava sporočil (mail).
Arhiviranje in stiskanje: WinZip, TAR, Gzip, 7-zip.

Znakovne datoteke in besedila

Kodiranje je povratno enolična preslikava med znaki abecede in neko drugo diskretno množico. Poznali so ga že stari Grki 2×5 bakle; semafor, Chappe konec 18. st., Morse 1837; telegrafija CCITT #2, 58 znakov, 1931.

Na računalnikih so znaki predstavljeni s števili – kodna tabela: število $\leftrightarrow$ znak.

Na računalniku CDC Cyber je beseda imala 60 bitov, v besedo je bilo mogoče shraniti 10 znakov.

Okrog leta 1960 je ameriška vojska začela zahtevati enotno kodo 7 bitna FIELDATA. EBCDIC, IBM 360, 1964. ASCII (American standard code for information interchange) 7 bitov, 1963 (osnutek, brez malih črk), 1967 $\rightarrow$ ISO 646, 1983. Razširjeni ASCII, 8 bitni; ISO 8859, 1987.

Koda ASCII

ASCII: 1, 2, 3, 4. ISO 8859, The ISO 8859 Alphabet Soup.

```
znaki
  krmilni
  izpisljivi
    števke
    črke (latinske abecede)
      velike
      male
    posebni znaki
      ločila
      operacije
      ostali
```

Microsoft, codepage: 850, MS-DOS Latin-1; cel kup za druge jezike. 1252 – Windows Latin 1, 1250 – Windows Latin 2, 1251 – Windows cirilica, . . .

Razširjeni ASCII – ISO 8859-*

ISO 8859-2 Latin 2

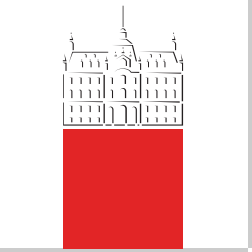
	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
		Ą	˘	Ł	¤	Ľ	Ś	§	¨	Š	Ş	Ť	Ź	-	Ž	Ż
B0	°	ą	˛	ł	´	ľ	ś	ˇ	¸	š	ş	ť	ź	˝	ž	ż
C0	Ŕ	Á	Â	Ă	Ä	Ĺ	Ć	Ç	Č	É	Ę	Ë	Ě	Í	Î	Ď
D0	Đ	Ń	Ň	Ó	Ô	Ő	Ö	×	Ř	Ů	Ú	Ű	Ü	Ý	Ţ	ß
E0	ŕ	á	â	ă	ä	ĺ	ć	ç	č	é	ę	ë	ě	í	î	ď
F0	đ	ń	ň	ó	ô	ő	ö	÷	ř	ů	ú	ű	ü	ý	ţ	˙

ISO 8859-7 Grška abeceda

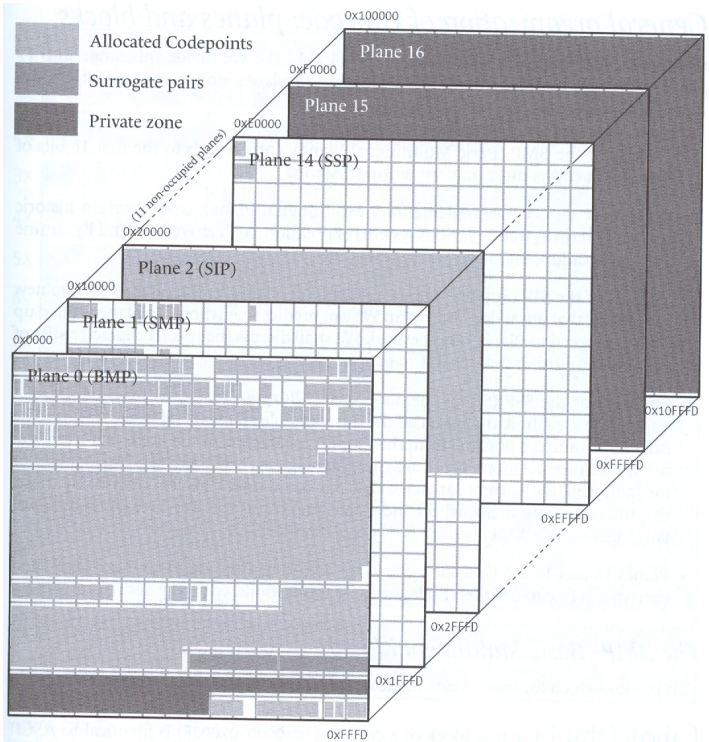
	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
		‘	’	£			¦	§	¨	©		«	¬	-		―
B0	°	±	²	³	΄	΅	Ά	·	Έ	Ή	Ί	»	Ό	½	Ύ	Ώ
C0	ΐ	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο
D0	Π	Ρ		Σ	Τ	Υ	Φ	Χ	Ψ	Ω	Ϊ	Ϋ	ά	έ	ή	ί
E0	ΰ	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
F0	π	ρ	ς	σ	τ	υ	φ	χ	ψ	ω	ϊ	ϋ	ό	ύ	ώ	

Windows 1250

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
80	€ 20AC		‚ 201A		„ 201E	… 2026	† 2020	‡ 2021		‰ 2030	Š 0160	‹ 2039	Ś 015A	Ť 0164	Ž 017D	Ź 0179
90		‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014		™ 2122	š 0161	› 203A	ś 015B	ť 0165	ž 017E	ź 017A
A0	NBSP 00A0	ˇ 02C7	˘ 02D8	Ł 0141	¤ 00A4	Ą 0104	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	Ş 015E	« 00AB	¬ 00AC	- 00AD	® 00AE	Ż 017B
B0	° 00B0	± 00B1	˛ 02DB	ł 0142	´ 00B4	µ 00B5	¶ 00B6	· 00B7	¸ 00B8	ą 0105	ş 015F	» 00BB	Ľ 013D	˝ 02DD	ľ 013E	ż 017C
C0	Ŕ 0154	Á 00C1	Â 00C2	Ă 0102	Ä 00C4	Ĺ 0139	Ć 0106	Ç 00C7	Č 010C	É 00C9	Ę 0118	Ë 00CB	Ě 011A	Í 00CD	Î 00CE	Ď 010E
D0	Đ 0110	Ń 0143	Ň 0147	Ó 00D3	Ô 00D4	Ő 0150	Ö 00D6	× 00D7	Ř 0158	Ů 016E	Ú 00DA	Ű 0170	Ü 00DC	Ý 00DD	Ţ 0162	ß 00DF
E0	ŕ 0155	á 00E1	â 00E2	ă 0103	ä 00E4	ĺ 013A	ć 0107	ç 00E7	č 010D	é 00E9	ę 0119	ë 00EB	ě 011B	í 00ED	î 00EE	ď 010F
F0	đ 0111	ń 0144	ň 0148	ó 00F3	ô 00F4	ő 0151	ö 00F6	÷ 00F7	ř 0159	ů 016F	ú 00FA	ű 0171	ü 00FC	ý 00FD	ţ 0163	˙ 02D9



Unicode



Code point	Byte 1	Byte 2	Byte 3	Byte 4
00000 00000000 0xxxxxxx	0xxxxxxx			
00000 00000yyy yyxxxxxx	110yyyyy	10xxxxxx		
00000 zzzzyyyy yyxxxxxx	1110zzzz	10yyyyyy	10xxxxxx	
uuuuu zzzzyyyy yyxxxxxx	11110uuu	10uuzzzz	10yyyyyy	10xxxxxx

16 bitov, Han, $2^{16} = 65536$, Unicode / charts, tabele.

Windows NT so že od začetka temeljili na Unicodu. Windows/System tools/Character map. MS Office. Na Unicodu temeljijo tudi spletne tehnologije.

Universal Character Set (UCS) določen z ISO/IEC 10646. UCS je nastal v sodelovanju z Unicode. Kodni prostor se razteza od 0 do $10FFFF_{16} = 17 \times 2^{16} = 1114112$ kod. 17 ravni: BMP - 0, basic multilingual plane (osnovni Unicode), SMP - 1, supplementary multilingual plane, SMP - 2, supplementary ideographic plane, 14-16, posebni nameni, industrija.

Kodiranja: UCS-2 (16 bitov), UCS-4 (32 bitov), UTF-8 (zapis spremenljive dolžine, 7-bitni ASCII v enem zlogu), UTF-16, ...

BOM – Byte Order Mark.

Oblike zapisa

Podatki, ki so shranjeni v izbrani obliki zapisa (formatu) so pogosto shranjeni na dvojiških datotekah – sestavin pri branju ni potrebno prekodirati. Za delo (ustvarjanje, urejanje, uporabo) s tovrstnimi podatki potrebujemo posebne programe. Primeri:

- besedila: DOC, PDF, ...
- slike: PNG, JPG, GIF, TIFF, ...
- zvok: WAV, AU, SND, MP3, ...
- video: MPEG, AVI, ...

Metapodatki

Podatki so sestavljeni iz dveh delov:

- *pravih podatkov* o stvareh, ki nas zanimajo, in
- *metapodatkov* – podatkov o podatkih, ki omogočajo razumevanje pravih podatkov

Metapodatki so na primer: oblika zapisa datoteke s podatki; kdo, kje, kdaj, kako in zakaj je podatke zbral; kodiranja (npr. 0 – moški, 1 – ženska); enote, v katerih so merjene posamezne količine; ...
Za večjo prenosljivost in lažje združevanje podatkov so za celo vrsto količin pripravljeni standardi, kako naj se jih zapisuje: datumi (ISO 8601), države, jeziki, ...
Metapodatki so pogosto shranjeni ločeno od datoteke s pravimi podatki. V zadnjem času jih pogosto združujejo v isti datoteki. Datoteke XML, opisi EXIF v datotekah s fotografijami, opisi ID3 v datotekah MP3, ... Tako se izognemo nevarnosti, da bi metapodatke izgubili.

Podatki na znakovnih datotekah

Oglejmo si zapise enot na znakovni datoteki. Zapis je sestavljen iz večih polj, ki vsebujejo vrednosti posameznih na enoti merjenih lastnosti.

- v starejših podatkovnih datotekah so pogosto polja urejena in vrednosti posameznega polja pripada določeno število mest;
- pogosto so tudi polja urejena in vrednosti med seboj ločene z izbranimi ločili
- kadar so poznane samo nekatere vrednosti polj, se večkrat uporablja zapis s pari imePolja=vrednost ločenimi z izbranimi ločili
- kadar so vrednosti v poljih obsežnejše ali celo sestavljene se opis enote razteza čez več (pogosto raznovrstnih) vrstic
- za opise enot s spremenljivo zgradbo se uporabljajo posebni jeziki za opis podatkov, kot sta XML in Jason.

DRY - Don't Repeat Yourself – podatek je shranjen samo na enem mestu.

trees.dat

D	H	V	CD	CH	CDCH	CD2	CH2	X
8.3	70	10.3	-4.9484	-6	29.6903	24.4865	36	4822.30
8.6	65	10.3	-4.6484	-11	51.1323	21.6075	121	4807.40
8.8	63	10.2	-4.4484	-13	57.8290	19.7881	169	4878.72
10.5	72	16.4	-2.7484	-4	10.9935	7.5536	16	7938.00
10.7	81	18.8	-2.5484	5	-12.7419	6.4943	25	9273.69
10.8	83	19.7	-2.4484	7	-17.1387	5.9946	49	9681.12
11.0	66	15.6	-2.2484	-10	22.4839	5.0552	100	7986.00
11.0	75	18.2	-2.2484	-1	2.2484	5.0552	1	9075.00
11.1	80	22.6	-2.1484	4	-8.5935	4.6156	16	9856.80
11.2	75	19.9	-2.0484	-1	2.0484	4.1959	1	9408.00
11.3	79	24.2	-1.9484	3	-5.8452	3.7962	9	10087.51
11.4	76	21.0	-1.8484	0	0.0000	3.4165	0	9876.96
11.4	76	21.4	-1.8484	0	0.0000	3.4165	0	9876.96
11.7	69	21.3	-1.5484	-7	10.8387	2.3975	49	9445.41
12.0	75	19.1	-1.2484	-1	1.2484	1.5585	1	10800.00
12.9	74	22.2	-0.3484	-2	0.6968	0.1214	4	12314.34
12.9	85	33.8	-0.3484	9	-3.1355	0.1214	81	14144.85
13.3	86	27.4	0.0516	10	0.5161	0.0027	100	15212.54
13.7	71	25.7	0.4516	-5	-2.2581	0.2040	25	13325.99
13.8	64	24.9	0.5516	-12	-6.6194	0.3043	144	12188.16
14.0	78	34.5	0.7516	2	1.5032	0.5649	4	15288.00
14.2	80	31.7	0.9516	4	3.8065	0.9056	16	16131.20
14.5	74	36.3	1.2516	-2	-2.5032	1.5665	4	15558.50
16.0	72	38.3	2.7516	-4	-11.0065	7.5714	16	18432.00
16.3	77	42.6	3.0516	1	3.0516	9.3123	1	20458.13
17.3	81	55.4	4.0516	5	20.2581	16.4156	25	24242.49
17.5	82	55.7	4.2516	6	25.5097	18.0762	36	25112.50
17.9	80	58.3	4.6516	4	18.6065	21.6375	16	25632.80
18.0	80	51.5	4.7516	4	19.0065	22.5778	16	25920.00
18.0	80	51.0	4.7516	4	19.0065	22.5778	16	25920.00
20.6	87	77.0	7.3516	11	80.8677	54.0462	121	36919.32

Branje podatkov


```
import sys, os

def cut(row,S):
   l=r=0; L=[]
   for (s, t) in S:
      r=r+s; z = row[l:r].strip(); l=r
      L.append(z)
   return L

os.chdir('C:\\Users\\Batagelj\\test\\python\\2011\\data')
podatki = 'trees.dat'
S = [ (4,'F'), (3,'I'), (5,'F'), (8,'F'), (4,'I'), (9,'F'), (8,'F'), (4,'I'), (9,'F') ]
with open(podatki, 'r') as dat:
   try:
      row = dat.readline(); names = cut(row[:-1],S)
      print(names)
      for row in dat.readlines():
         L = cut(row[:-1],S); R = []
         for (i,(s, t)) in enumerate(S):
            z = L[i]; t = t.upper()
            if t != 'X':
               if   t == 'I': v = int(z)
               elif t == 'F': v = float(z)
               elif t == 'S': v = z
               else: v = z
               R.append(v)
         print(R)
   except Exception as e: sys.exit('file {}: {}'.format(podatki, e))

>>>
['D', 'H', 'V', 'CD', 'CH', 'CDCH', 'CD2', 'CH2', 'X']
[8.3, 70, 10.3, -4.9484, -6, 29.6903, 24.4865, 36, 4822.3]
[8.6, 65, 10.3, -4.6484, -11, 51.1323, 21.6075, 121, 4807.4]
[8.8, 63, 10.2, -4.4484, -13, 57.829, 19.7881, 169, 4878.72]
...
```

CSV – Comma-Separated Values

Ni standard, a v večini izvedb velja:

- vrednosti za posamezno enoto so zapisane v eni vrstici
- polja so urejena, vrednosti so ločene z izbranim ločilom (vejica, podpičje, predelčnik (TAB), . . .)
- dvojne navednice " imajo posebno vlogo. Polja, ki vsebujejo ločilo kot del vrednosti, morajo biti vklenjena v par dvojnih navednic,
- polja, ki vsebujejo znak " morajo biti vklenjena v par dvojnih navednic; posamezna pojavitev znaka " je zapisana s parom ""
- na začetku datoteke je lahko vrstica, ki vsebuje imena polj

Oblika CSV se previloma uporablja za prenos podatkov med različnimi razpredelniškimi programi (Excel).

Datoteke CSV – rdeča vina

http://archive.ics.uci.edu/ml/machine-learning-databases/wine/

```
"fixed acidity";"volatile acidity";"citric acid";"residual sugar";"chlorides";"free sulfur dioxi
7.4;0.7;0;1.9;0.076;11;34;0.9978;3.51;0.56;9.4;5
7.8;0.88;0;2.6;0.098;25;67;0.9968;3.2;0.68;9.8;5
7.8;0.76;0.04;2.3;0.092;15;54;0.997;3.26;0.65;9.8;5
11.2;0.28;0.56;1.9;0.075;17;60;0.998;3.16;0.58;9.8;6
7.4;0.7;0;1.9;0.076;11;34;0.9978;3.51;0.56;9.4;5
7.4;0.66;0;1.8;0.075;13;40;0.9978;3.51;0.56;9.4;5
7.9;0.6;0.06;1.6;0.069;15;59;0.9964;3.3;0.46;9.4;5
7.3;0.65;0;1.2;0.065;15;21;0.9946;3.39;0.47;10;7
7.8;0.58;0.02;2;0.073;9;18;0.9968;3.36;0.57;9.5;7
7.5;0.5;0.36;6.1;0.071;17;102;0.9978;3.35;0.8;10.5;5
6.7;0.58;0.08;1.8;0.097;15;65;0.9959;3.28;0.54;9.2;5
7.5;0.5;0.36;6.1;0.071;17;102;0.9978;3.35;0.8;10.5;5
5.6;0.615;0;1.6;0.089;16;59;0.9943;3.58;0.52;9.9;5
...
6.2;0.6;0.08;2;0.09;32;44;0.9949;3.45;0.58;10.5;5
5.9;0.55;0.1;2.2;0.062;39;51;0.99512;3.52;0.76;11.2;6
6.3;0.51;0.13;2.3;0.076;29;40;0.99574;3.42;0.75;11;6
5.9;0.645;0.12;2;0.075;32;44;0.99547;3.57;0.71;10.2;5
6;0.31;0.47;3.6;0.067;18;42;0.99549;3.39;0.66;11;6
```

Datoteke CSV – rdeča vina


```
import csv, sys, os
# http://docs.python.org/py3k/library/csv.html
os.chdir('C:\\Users\\Batagelj\\test\\python\\2011\\data')
podatki = 'winequality-red.csv'
with open(podatki, newline='', encoding='windows-1250') as dat:
   wineReader = csv.reader(dat, delimiter=';', quotechar='"')
   reds = []
   try:
      head = next(wineReader)
      for row in wineReader:
#         print(', '.join(row))
         row = [eval(v) for v in row]
         reds.append(row)
   except csv.Error as e:
      sys.exit('file {}, line {}: {}'.format(
          podatki, wineReader.line_num, e))
print(len(reds))
print(head)
for i in range(10): print(i,reds[i])
```

```
>>>
1599
['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar', 'chlorides', 'free sulfur
0 [7.4, 0.7, 0, 1.9, 0.076, 11, 34, 0.9978, 3.51, 0.56, 9.4, 5]
1 [7.8, 0.88, 0, 2.6, 0.098, 25, 67, 0.9968, 3.2, 0.68, 9.8, 5]
2 [7.8, 0.76, 0.04, 2.3, 0.092, 15, 54, 0.997, 3.26, 0.65, 9.8, 5]
3 [11.2, 0.28, 0.56, 1.9, 0.075, 17, 60, 0.998, 3.16, 0.58, 9.8, 6]
4 [7.4, 0.7, 0, 1.9, 0.076, 11, 34, 0.9978, 3.51, 0.56, 9.4, 5]
5 [7.4, 0.66, 0, 1.8, 0.075, 13, 40, 0.9978, 3.51, 0.56, 9.4, 5]
6 [7.9, 0.6, 0.06, 1.6, 0.069, 15, 59, 0.9964, 3.3, 0.46, 9.4, 5]
7 [7.3, 0.65, 0, 1.2, 0.065, 15, 21, 0.9946, 3.39, 0.47, 10, 7]
8 [7.8, 0.58, 0.02, 2, 0.073, 9, 18, 0.9968, 3.36, 0.57, 9.5, 7]
9 [7.5, 0.5, 0.36, 6.1, 0.071, 17, 102, 0.9978, 3.35, 0.8, 10.5, 5]
>>>
```

Opis člankov na Web of Science


```
PT J
AU Enomoto, H
   Hirohata, K
   Ota, K
TI Long cycles passing through a specified edge in a S-connected graph
SO JOURNAL OF GRAPH THEORY
LA English
DT Article
AB We prove the following theorem: For a connected noncomplete graph G,
   let tau(G): = min{d(G)(u) + d(G)(v)\d(G)(u,v) = 2} Suppose G is a
   3-connected noncomplete graph. Then through each edge of G there passes
   a cycle of length greater than or equal to min{\V(G)\, tau(G) - 1}. (C)
   1997 John Wiley & Sons, Inc.
RP Enomoto, H, KEIO UNIV,FAC SCI & TECHNOL,DEPT MATH,KOHOKU
   KU,YOKOHAMA,KANAGAWA 223,JAPAN.
CR DIRAC GA, 1978, ANN DISCRETE MATH, V3, P75
   ENOMOTO H, 1984, J GRAPH THEOR, V8, P287
   FAN GH, 1984, J COMB THEORY B, V37, P221
NR 3
TC 3
PU JOHN WILEY & SONS INC
PI NEW YORK
PA 605 THIRD AVE, NEW YORK, NY 10158-0012
SN 0364-9024
J9 J GRAPH THEOR
JI J. Graph Theory
PD MAR
PY 1997
VL 24
IS 3
BP 275
EP 279
PG 5
SC Mathematics
GA WH860
UT ISI:A1997WH86000009
ER
```

GEDCOM – opisi rodovnikov

GEDCOM je standard za opis in izmenjavo rodovniških podatkov.

```
0 HEAD
1 FILE ROYALS.GED
...
0 @I58@ INDI
1 NAME Charles Philip Arthur/Windsor/
1 TITL Prince
1 SEX M
1 BIRT
2 DATE 14 NOV 1948
2 PLAC Buckingham Palace, London
1 CHR
2 DATE 15 DEC 1948
2 PLAC Buckingham Palace, Music Room
1 FAMS @F16@
1 FAMC @F14@
...
...
0 @I65@ INDI
1 NAME Diana Frances /Spencer/
1 TITL Lady
1 SEX F
1 BIRT
2 DATE 1 JUL 1961
2 PLAC Park House, Sandringham
1 CHR
2 PLAC Sandringham, Church
1 FAMS @F16@
1 FAMC @F78@
...
...
```

```
0 @I115@ INDI
1 NAME William Arthur Philip/Windsor/
1 TITL Prince
1 SEX M
1 BIRT
2 DATE 21 JUN 1982
2 PLAC St.Mary's Hospital, Paddington
1 CHR
2 DATE 4 AUG 1982
2 PLAC Music Room, Buckingham Palace
1 FAMC @F16@
...
0 @I116@ INDI
1 NAME Henry Charles Albert/Windsor/
1 TITL Prince
1 SEX M
1 BIRT
2 DATE 15 SEP 1984
2 PLAC St.Mary's Hosp., Paddington
1 FAMC @F16@
...
0 @F16@ FAM
1 HUSB @I58@
1 WIFE @I65@
1 CHIL @I115@
1 CHIL @I116@
1 DIV N
1 MARR
2 DATE 29 JUL 1981
2 PLAC St.Paul's Cathedral, London
```

Pobiranje datotek s spleta


```
# downloading genealogical data
# http://zvonka.fmf.uni-lj.si/netbook/doku.php?id=notes:gendl

import urllib.request, os
from urllib.error import HTTPError,URLError

def downloadFile(fileName,fileMode,baseURL):
   url = baseURL + fileName
   try:
      f = urllib.request.urlopen(url)
      d = open(fileName, "w" + fileMode)
      d.write(f.read()); d.close()
      print("file ",url," downloaded")
      return 0
   except HTTPError as e: print("HTTP Error:",e.code,url); return 1
   except URLError  as e: print("URL Error:",e.reason,url); return 2

os.chdir('C:\\Users\\Batagelj\\test\\python\\2011\\data\\ged')
for ind in range(7,11):
   dirURL = 'http://www.genealogyforum.com/gedcom/gedcom2a/'
   name   = "gedr2" + str(1000+ind)[1:]
   s = downloadFile(name+".htm","b",dirURL)
   s = downloadFile(name+".ged","b",dirURL)
   if s > 0: s = downloadFile(name+".zip","b",dirURL)
>>>
file  http://www.genealogyforum.com/gedcom/gedcom2a/gedr2007.htm  downloaded
file  http://www.genealogyforum.com/gedcom/gedcom2a/gedr2007.ged  downloaded
file  http://www.genealogyforum.com/gedcom/gedcom2a/gedr2008.htm  downloaded
HTTP Error: 404 http://www.genealogyforum.com/gedcom/gedcom2a/gedr2008.ged
file  http://www.genealogyforum.com/gedcom/gedcom2a/gedr2008.zip  downloaded
HTTP Error: 404 http://www.genealogyforum.com/gedcom/gedcom2a/gedr2009.htm
HTTP Error: 404 http://www.genealogyforum.com/gedcom/gedcom2a/gedr2009.ged
HTTP Error: 404 http://www.genealogyforum.com/gedcom/gedcom2a/gedr2009.zip
file  http://www.genealogyforum.com/gedcom/gedcom2a/gedr2010.htm  downloaded
HTTP Error: 404 http://www.genealogyforum.com/gedcom/gedcom2a/gedr2010.ged
```

Branje spletnih strani

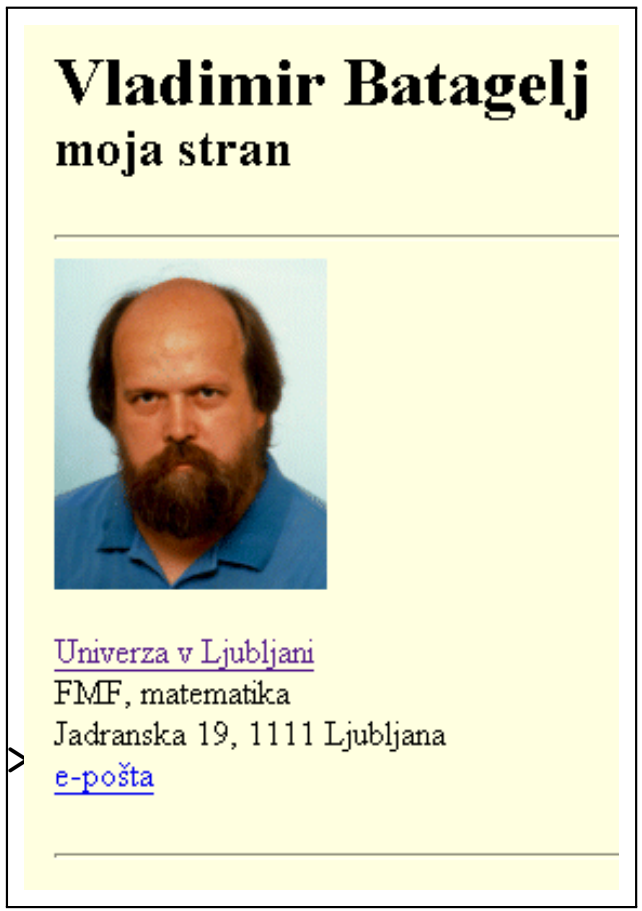

```
# get the code of a given URL as html text string
# Python3 does not read the html code as string but as bytearray

import urllib.request

def extract(text,left,right):
   return text.split(left,1)[-1].split(right,1)[0]

fweb = urllib.request.urlopen("http://www.python.org")
fbytes = fweb.read()
encoding = extract(str(fbytes).lower(),'charset=','"')
print("Encoding type = ",encoding)
if encoding:
   fstr = fbytes.decode(encoding)
   print(fstr[:500])
else: print("Encoding type not found!")
fweb.close()

>>>
Encoding type =  utf-8
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/


<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8" />
  <title>Python Programming Language &ndash; Official Website</title>
  <meta name="keywords" content="python programming language object oriented web free source" />
  <meta name="description" content="      Home page for Pytho
>>>
```

Označevanje – SGML

En od pomembnejših pojmov pri delu s podatki postaja *označevanje*: dele besedila oklenemo z značkami, ki določajo, kaj je dani del besedila in kako naj bo oblikovan.
Čeprav vsi opisi oblikovanega besedila temeljijo na označevanju, predstavlja prelomnico SGML (Standard Generalized Markup Language) sprejet leta 1986. SGML je sestav za pripravo definicij označevalnih jezikov:
HTML – HyperText Markup Language, ISO 12083 – Electronic Manuscript Standard, TEI – Text Encoding Initiative, CALS/ JTA – Computer-aided Acquisition and Logistic Support, NITF – News Industry Text Format, DDI – Data Documentation Initiative, CML – Chemical Markup Language,
SGML je namenjen opisu *zgradbe* podatkov, ki je ločen (pravokoten na) od opisa *oblike* (prikaza). Oblika je določena s *slogi*. V HTMLju so to prekrivni slogi CSS W3C/CSS, W3 schools/CSS, MIRK'04.
Za podporo razvoja rešitev je bilo razvitih več orodij Clark.

Označevanje – HTML

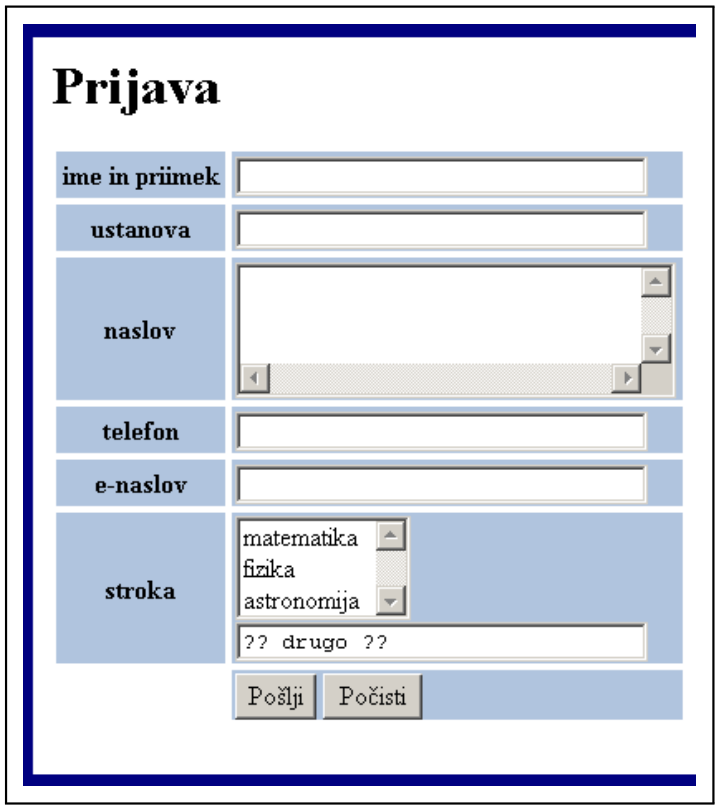

```
<html>
 <head>
  <title>V.B. - moja stran</title>
  <meta http-equiv='content-type'
   content='text/html;charset=utf-8'>
  <meta name="author" content="V.B.">
  <meta name='creation_date'
   content='december 11, 2002'>
 </head>
 <body bgcolor="lightyellow">
  <center><table width="670"><tr><td>
  <h1>Vladimir Batagelj<br>
   <small>moja stan</small></h1>
  <hr><img src="./vlado.gif"><p>
  <a href="http://www.uni-lj.si">
  Univerza v Ljubljani</a><br>FMF,
  matematika<br> Jadranska 19, 1111
  Ljubljana<br><a href=
  "mailto:vladimir.batagelj@uni-lj.si">
  e-po&#353;ta</a></p><hr>
  </td></tr></table></center>
 </body>
</html>
```



Moja stran. Opentype/Unicode. W3 schools, NS guide.

Obrazci


```
<html>
<head><title>Prijava</title>
 <meta http-equiv="Content-Type"
  content="text/html;charset=UTF-8"></head>
<body bgcolor=navy><center>
<table bgcolor='white' width=600 cellpadding=10>
<tr><td><h1>Prijava</h1>
 <form method='post' enctype='text/plain'
    action='mailto:vladimir.batagelj@uni-lj.si'>
  <table bgcolor='lightsteelblue'>
   <tr><th width=100>ime in priimek</th><td>
    <input type='text' name='name' size=30>
   </td></tr><tr><th>ustanova</th><td>
    <input type='text' name='inst' size=30>
   </td></tr><tr><th>naslov</th><td>
    <textarea rows=3 name='addr' cols=30></textarea>
   </td></tr><tr><th>telefon</th><td>
    <input type='text' name='coun' size=30>
   </td></tr><tr><th>e-naslov</th><td>
    <input type='text' name='emai' size=30>
   </td></tr><tr><th>stroka</th><td>
    <select name='acti' size=3 multiple>
     <option>matematika   <option>fizika
     <option>astronomija
    </select><br>
    <input type='text' name='acti' size=30
     value='?? drugo ??'>
   </td></tr><tr><td></td><td>
    <input type='submit' value='Po&#353;lji'>
    <input type='reset'  value='Po&#269;isti'>
 </td></tr></table></form></td></tr></table>
 </center></body>
</html>
```



Prijava; Prijava/CGI, Prijavljeni; Prijava/CGI-Excel, Prijavljeni(csv);

Program dodaj.

Slogi – CSS


```
<head>
 ...
 <style type="text/css">
  h1, small {color: darkred;
    font-family: Comic Sans MS;}
  h1 {font-size: 30pt;}
  small {font-size: 20pt;}
  p {font-family: Arial;
     font-size: 15pt;
     color: navy;}
  a {text-decoration: none;}
  a.text {color: red;}
 </style>
</head>
```

Moja stran.

XML – eXtensible Markup Language

Z razvojem spleta se je pojavila potreba, da lahko uporabnik razširi HTML s svojimi oznakami. Prirejena izpeljanka SGMLja je XML – eXtensible Markup Language. XML omogoča hranjenje, izmenjavo in lažjo obdelavo podatkov.
XML je ohranil pomembnejše zmogljivosti in glavne značilnosti SGMLja. Predvsem je z odpravo 'potuh' sestav precej poenostavil.
XML se vse bolj uveljavlja tudi pri uporabah informacijske tehnologije v izobraževanju.

Značke

Kakor v HTMLju se značke pojavljajo v parih <*ime*> in </*ime*> – *oklepajne* značke. Tudi *samostojne* značke so v bistvu oklepajne – imajo obliko <*ime*></*ime*> ali okrajšano <*ime* />.
Del opisa, ki ga nek par značk oklepa, je *vsebina* te značke. Ta je lahko *prazna* (za samostojne znacke), *enostavna* (niz znakov, ki ne vsebuje drugih značk) ali pa je *sestavljena*.
V imenih značk se velikost črk upošteva. XML podpira *samoopisnost* imen. *Ime*na značk so nizi znakov, ki ne vsebujejo presledkov ali dvopičij in se ne začnejo s številko, ločilom ali podnizom `xml` (`XML`, `Xml`, ...).
Značka ima lahko lastnosti <*ime* l_1="v_1" l_2="v_2" ... l_k="v_k" > .
Vrednosti lastnosti so v navednicah " ali ’.
Večkratni presledki se ne skrčijo v enega; nova vrsta je predstavljena z znakom `LF`. *Pojasnila* so kot v SGML obdana z <!-- ... --> .

Dobro oblikovani opisi

Opis v nekem označevalnem jeziku *jezik* nad XML je *dobro oblikovan* (well formed), če

- vsak opis oklepa *glavna* (korenska) značka <*jezik*> ;
- v opisu podatkov so ta vsebini poljubnih dveh značk ali ločeni ali pa je ena vsebovana v drugi (*gnezdenje*) – zaporedje <*a*> . . . <*b*> . . . </*a*> . . . </*b*> ni dovoljeno;

Dobro oblikovani opis lahko predstavimo kot drevo – DOM (Document Object Model).

. . . Dobro oblikovani opisi

Podatke lahko v opisu predstavimo kot vsebino ali vrednost neke lastnosti. Priporočilo: prave podatke predstavimo kot vsebino; lastnosti povedo podatke o podatkih.
Novejše izdaje spletnih pregledovalnikov Internet Explorer (vsaj 5.5) in Mozilla/ Netscape (vsaj 7) omogočajo preverjanje dobre oblikovanosti opisov in ličen (določen s prekrivnimi slogi) prikaz dobro oblikovanih opisov.
S slogi lahko določimo prikaz dobro oblikovanih opisov. CD.xml, CD.css, CD.css+xml.

Primer: zbirka knjig – opis podatkov v XML


```
<?xml version="1.0" encoding="Windows-1250" ?>
<?xml-stylesheet type="text/css" href="zbirka.css" ?>
<!-- 3. december 2002 -->
<zbirka>
 <knjiga jezik="slo" vezava="trd">
  <avtor>Janez Novak</avtor>
  <naslov>Gojenje glist</naslov>
  <zalozba>DZS</zalozba>
  <kraj>Ljubljana</kraj>
  <leto>1995</leto>
  <opomba>Zelo zanimivo</opomba>
 </knjiga>
 <knjiga jezik="slo" vezava="meh">
   <avtor>Peter Škafar</avtor>
   <naslov>Razmišljanja</naslov>
   <zalozba>MK</zalozba>
   <kraj>Ljubljana</kraj>
   <leto>2001</leto>
   <opomba>Za vztrajne.</opomba>
 </knjiga>
</zbirka>
```

Primer: zbirka knjig - prekrivni slogi


```
zbirka { font-size: 20pt;
   font-family: Courier New;
   background-color: Wheat;
   width: 400;}
knjiga { display: block;
   margin-bottom: 30pt;
   margin-left: 0;
   background-color: Cornsilk;}
avtor { font-family: Arial;}
naslov { font-family: Impact;
   display: block;
   color: Maroon;
   margin-left: 20pt;}
zalozba { display: block;
   margin-left: 20pt;}
kraj { margin-left: 20pt;}
leto { }
opomba { display: block;
   margin-left: 20pt;
   color: DarkGreen;
   font-size: 12pt;}
```

Janez Novak
Gojenje glist
DZS
Ljubljana 1995
Zelo zanimivo

Peter Škafar
Razmišljanja
MK
Ljubljana 2001
Za vztrajne.

Pravilni opisi

Zaenkrat še nismo povedali, kateri opisi so 'slovnično' pravilni – v kakšnih medsebojnih odnosih so lahko posamezne značke, katere lastnosti imajo, . . . ? Slovnico označevalnega jezika določimo z DTD (Document Type Definition), s katerim povemo, kako so posamezne značke med seboj povezane. Obstaja tudi opis slovnice s shemami.
Opisi, ki zadoščajo dani slovnici so *pravilni* (valid).
Pri zapisu slovnice uporabljamo naslednje stavke:

Ukaz ELEMENT

Značko vpeljemo z ukazom

<!ELEMENT ime sestava >

kjer je *ime* ime vpeljane značke in je opis *sestave*:

- `EMPTY` – samostojna značka;
- `ANY` – poljubno zaporedje znakov;
- `(#PCDATA)` – zaporedje znakov (brez značk);
- `(#PCDATA|`*ime*`...)*` – mešano zaporedje;
- (*členi*) *določilo* – členi sestavljajo ali *zaporedje*, če so ločeni z vejico , (tudi v opisu se morajo pojaviti v istem vrstnem redu); ali *izbiro*, če so ločeni s črtico | (v opisu se mora pojaviti le en izmed njih). določilo je ali prazen znak ali en izmed znakov ?, + in *.

. . . ukaz ELEMENT

Posamezni *člen* je lahko:

- *ime* – ime podrejene značke (otroka);
- *ime*? – največ ena pojavitev;
- *ime*+ – vsaj ena pojavitev;
- *ime*∗ – poljubno (tudi 0) pojavitev;
- (*členi*) *določilo* .

Posamezni znački ustreza natanko en ukaz ELEMENT.

Ukaza ATTLIST in CDATA

Lastnost *lastnost* značke *ime* opredelimo z ukazom

<!ATTLIST ime lastnost zvrst vrednost>

kjer je *zvrst* opis zvrsti vrednosti lastnosti *lastnost* in *vrednost* določa vgrajeno vrednost lastnosti.
Podrobno o vrednostih lastnosti.

Z ukazom

<![CDATA[vsebina]]>

zahtevamo, da se *vsebina* upošteva dobesedno.

Ukaz ENTITY

Najpogostejše oblike vpeljave delca *ime* so okrajšave

<!ENTITY ime "niz_znakov" >

in povezave na datoteke

<!ENTITY ime vrsta "naslov" >

Pri tem je *vrsta* ali `SYSTEM` (lastna datoteka) ali `PUBLIC` (poskus uporabe javne datoteke).
Pri kodiranih datotekah na koncu ukaza `ENTITY` dodamo še par `NDATA` *oblika* in jo z ukazom

<!NOTATION oblika SYSTEM "program" >

povežemo z ustreznim prikazovalnim *program*om.

Primer: DTD za zbirka


```
<!ENTITY DZS "Drzni znanilci sprememb">
<!ENTITY MK  "Mladinska knjiga">
<!ELEMENT zbirka  (knjiga+)>
<!ELEMENT knjiga  (avtor+,naslov,zalozba?,kraj?,leto?,opomba*)>
<!ELEMENT avtor   (#PCDATA)>
<!ELEMENT naslov  (#PCDATA)>
<!ELEMENT zalozba (#PCDATA)>
<!ELEMENT kraj    (#PCDATA)>
<!ELEMENT leto    (#PCDATA)>
<!ELEMENT opomba  (#PCDATA)>
<!ATTLIST knjiga
  jezik  (slo|ang|nem|shr|ita|ost) "slo"
  vezava (trd|meh|raz|dru) "dru" >
```

Sheme in imenski prostori

Slovnico lahko opišemo tudi s shemami XML Schema, ki same temeljijo na XMLju.
Pri hkratni uporabi večih označevalnih jezikov lahko pride do prekrivanja – uporabe istih imen značk. Problem je razrešen z uvedbo imenskih prostorov (namespace).

<značka xmlns:prostor='naslov'>

Polno ime značke ima obliko *prostor* : *ime* .
Označevanje in XML (MIRK'04)

Prepletanje XML in XHTML


```
<?xml version="1.0" encoding="Windows-1250" ?>
<?xml-stylesheet type="text/css" href="zbirka.css" ?>
<!DOCTYPE zbirka [
 <!ENTITY DZS "Drzni znanilci sprememb">
 <!ENTITY MK "Mladinska knjiga">
]>
<zbirka xmlns:h="http://www.w3.org/1999/xhtml">
 <h:html><h:head>
  <h:title>Knjige</h:title></h:head>
  <h:body><h:h1>Knjige</h:h1>
  <h:img src="./surfer2.gif" width="120" />
 </h:body></h:html>
 <knjiga jezik="slo" vezava="trd">
  <avtor>Janez Novak</avtor>
  <naslov>Gojenje glist</naslov>
  <zalozba>&DZS;</zalozba>
  <kraj>Ljubljana</kraj>
  <leto>1995</leto>
  <opomba>Zelo zanimivo</opomba>
 </knjiga>
...
```

Viri: XML, DTD, libxml2

XML in Python

- paket xml
- DOM: xml.minidom
- SAX: xml.sax
- py: xml.etree.ElementTree
- lxml: nadgradnja etree, dostop do najboljših knjižnic

etree

Knjižnica `xml.etree` je preprosta knjižnica, ki omogoča delo z dobro oblikovanimi opisi XML. docs.
Drevo opisa je sestavljeno iz vozlišč – objektov ElementTree, ki omogočajo opis hierarhično urejenih podatkov
Ukaz ElementTree([značka[,vir]]) ustvari primerek objekta ElementTree s korenom vrste značka. Vsebina primerka je vzeta iz datoteke vir.

Naj bo tree primerek objekta ElementTree.
tree._setroot(značka) – postavi značko za koren.
tree.find(pot) – določi najvišje vozlišče, ki ustreza poti. Opisi poti:
'značka' – določi vrhnjo pojavitev značke; gnezdenih pojavitev ne upošteva;
'predniki/značka' – določi pojavitev značke, ki ima zahtevane prednike;
'*' – določi vse otroke; '*/značka' – vsi vnuki z dano značko;
'.' – tekoče vozlišče;
'//' – vsa podvozlišča; './/značka' – vsa vozlišča z dano značko

etree

tree.findall(pot) – določi vsa vozlišča, ki ustrezajo poti, in jih vrne kot ponovnik.
tree.findtext(pot[,privzeto]) – vrne besedilo, ki pripada najvišjemu vozlišču za dano pot; če ga ni, vrne privzeto.
tree.Element.iter([značka]) – vrne ponovnik čez vse pojavitve značke; če je opuščena, čez vsa vozlišča.
tree.getroot() – vrne koren.
tree.parse(vir) – vrne razčlenjen opis z datoteke vir.
tree.write(datoteka[,koda]) – izpiše drevo na datoteko v dani kodi.
Nad vozliščem v lahko opravimo naslednje operacije:
len(v) – število otrok vozlišča v.
v[n] – n-ti otrok vozlišča v.
v[n] = primerek – zamenjaj n-tega otroka s primerkom.
del v[n] – odstrani n-tega otroka.
v.tag – značka vozlišča.
v.text – podatki, ki jih značka oklepa.
v.tail – niz, ki sledi zaklepajni znački.
v.attrib – slovar lastnosti značke.

etree

Metode:
v.append(u)
v.clear()
v.find(pot)
v.findall(pot)
v.findtext(pot)
v.get(lastnost[,privzeto])
v.list()
v.Element.iter([značka])
v.insert(n,u)
v.items()
v.keys()
v.remove(u)
v.set(lastnost,vrednost)
dump(v)
tostring(v)

Zbiranje podatkov s spleta

Knjižnica lxml je nadgradnja knjižnice etree in omogoča tudi zahtevnejše obdelave.
Knjižnica lxml vsebuje posebna razčlenjevalnika za XML in HTML – etree.XML(opis) in etree.HTML(opis). Namesto ukaza tree.findall(pot) pa uporabljamo zmogljivejši ukaz tree.xpath(pot).
Za primer si bomo pogledali uporabo pri zbiranju podatkov iz spletnih strani – npr. o knjigah z Amazona.

Zgledi


```
>>> # import xml.etree.ElementTree as etree
>>> from lxml import etree
>>> import os, sys
>>> os.chdir('C:\\Users\\Batagelj\\test\\python\\2011\\XML')
>>> T = etree.ElementTree(file="zbirka.xml")
>>> r = T.getroot()
>>> r.tag
'zbirka'
>>> r.attrib
{}
>>> k = T.findall('knjiga')
>>> k
[<Element knjiga at 0x36faaa8>, <Element knjiga at 0x36faa58>]
>>> for b in k: print(b.attrib)

{'vezava': 'trd', 'jezik': 'slo'}
{'vezava': 'meh', 'jezik': 'slo'}
>>> print(etree.tostring(k[1]).decode('utf8'))
<knjiga jezik="slo" vezava="meh">
      <avtor>Peter &#352;kafar</avtor>
      <avtor>Micka Kova&#269;</avtor>
      <naslov>Razmi&#353;ljanja</naslov>
      <zalozba>Mladinska knjiga</zalozba>
      <kraj>Ljubljana</kraj>
      <leto>2001</leto>
      <opomba>Za vztrajne.</opomba>
   </knjiga>
```

Preverjanje pravilnosti datotek XML


```
from lxml import etree

def isValid(XMLfile,DTDfile=None):
    if DTDfile is None:
        parser = etree.XMLParser(dtd_validation=True)
    else:
        dtd = etree.DTD(DTDfile)
        parser = etree.XMLParser(load_dtd=True)
    try:
        tree = etree.parse(XMLfile, parser)
        if DTDfile is None:
            print("{} is a valid XML file".format(XMLfile))
        elif dtd.validate(tree):
            print("{} is a valid XML file".format(XMLfile))
        else:
            print("ERROR",dtd.error_log.filter_from_errors()[0],
            "\n*** {} is not a valid XML file".format(XMLfile))
            error = dtd.error_log[0]
            print(error.message)
            print('line:',error.line,'  column:',error.column)
    except Exception as err:
        print("ERROR {}\n*** {} is not a well-formed XML file".\
              format(err,XMLfile))
```

Preverjanje dobre oblikovanosti datotek XML


```
from xml.sax.handler import ContentHandler
import xml.sax
def isWellFormed(file):
    xmlparser = xml.sax.make_parser()
    xmlparser.setContentHandler(ContentHandler())
    try:
        xmlparser.parse(file)
        print("{} is a well-formed XML file".format(file))
    except Exception as err:
        print("ERROR {}\n*** {} is not a well-formed XML file".\
              format(err,file))
```

Amazon – podatki o knjigi


```
from lxml import etree
import urllib.request

bookURL = "http://www.amazon.com/dp/0521387078/"
# bookURL = "http://www.amazon.com/dp/0761963391/"
# bookURL = "http://www.amazon.com/dp/0521602629/"
f = urllib.request.urlopen(bookURL)
book = f.read().decode('iso-8859-1')
f.close()
html = etree.HTML(book)

# title
t = html.xpath('//div/h1/span')[0].text
print(t)
# authors
a = html.xpath('//meta')
for i in range(len(a)):
    if 'name' in a[i].attrib.keys() :
        if a[i].attrib['name']=='title' :
            authors = a[i].attrib['content'].split(':')[-2].strip()
            break
print(authors)
# Product Details
r = html.xpath('//li/b')
# binding
bind = r[0].text.strip()
pages = r[0].tail.strip().split(' ')[0]
print(bind, pages)
# publisher
R = r[1].tail.split('(')
publisher = R[0].strip()
j = R[1].rfind(',')
if j < 0: j = 0
year = R[1][j+1:-1].strip()
print(publisher,year)
```

. . . Amazon – podatki o knjigi


```
# Prices
p = html.xpath('//table/tr/td/span')
for i in range(len(p)):
    if 'class' in p[i].attrib.keys():
        if p[i].attrib['class']=="listprice":
            listprice = p[i].text[1:]
            break
print(i,'listprice =',listprice)
for i in range(len(p)):
    if 'class' in p[i].attrib.keys():
        if p[i].attrib['class']=="price":
            price = p[i].text[1:]
            break
print(i,'price =',price)

# Customers Who Bought This Item Also Bought
b = html.xpath('//div/ul/li/div/a')
for i in range(len(b)):
    j = b[i].attrib['href'].rfind('/')
    amazonID = b[i].attrib['href'][j+1:]
    print(i,amazonID)
```

Amazon – pregledovanje knjig


```
from lxml import etree
import urllib.request, datetime

Q = ["0521387078", "0761963391", "0521602629"]     # nepregledane knjige
K = { k:(i+1) for (i,k) in enumerate(Q)}; num = len(K) # že znane knjige
while len(Q) > 0:
  amazonID = Q.pop(0); ind = K[amazonID]
  if ind > 10: break
  time = datetime.datetime.now()
  print(ind,amazonID,time.ctime(),len(Q),len(K))
  bookURL = "http://www.amazon.com/dp/"+amazonID+"/"
  with urllib.request.urlopen(bookURL) as f:
    book = f.read().decode('iso-8859-1','replace')
  html = etree.HTML(book)
# process html  ......
# Customers Who Bought This Item Also Bought
  b = html.xpath('//div/ul/li/div/a')
  L = []
  for i in range(len(b)):
    j = b[i].attrib['href'].rfind('/')
    amID = b[i].attrib['href'][j+1:]
    if len(amID)==10:
      if amID in K.keys(): jnd = K[amID]
      else:
        num += 1; jnd = num; K[amID] = num; Q.append(amID)
      L.append(amID); L.append(jnd)
  print("***",ind,len(Q),len(K),amazonID,L)

>>>
1 0521387078 Tue May 15 08:55:09 2012 2 3
*** 1 7 8 0521387078 ['1412927498', 4, '0761963391', 2, '0199206651', 5, '0123822297', 6, '05216
2 0761963391 Tue May 15 08:55:19 2012 6 8
*** 2 9 11 0761963391 ['1412927498', 4, '0521387078', 1, '0123822297', 6, '0195379470', 9, '1412
3 0521602629 Tue May 15 08:55:25 2012 8 11
*** 3 10 13 0521602629 ['1412927498', 4, '0521387078', 1, '0123822297', 6, '0521600979', 7, '159
4 1412927498 Tue May 15 08:55:31 2012 9 13
```

Amazon 1


```
# Amazon book data harvesting
# Vladimir Batagelj, May 2012

# bookURL = "http://www.amazon.com/dp/0521387078/"

from lxml import etree
import urllib.request
import os, sys, csv, datetime, time
class inspectError(RuntimeError): pass

def harvest():
  global Q, K, num, ind, dat

  def inspect(book):
    global Q, K, num, ind, dat
    html = etree.HTML(book)
# title & authors
    tit = html.xpath('//title')
    try:
      titext = tit[0].text.strip()
      if not titext.endswith("Books"): raise inspectError()
      b = titext.split(':')
      if titext.startswith("Amazon"):
        authors = b[-2].strip()
        i = titext.find(":")+2; j = titext.rfind(":",0,len(titext)-8)
        title = titext[i:titext.rfind("(",0,j)].strip()
      else:
        authors = b[-4].strip()
        title = titext[:titext.find(": "+authors[:8])]
    except Exception:
      print('*** ERROR title:',ind,amazonID); raise inspectError()
#   print(ind,title)
# Product Details
    r = html.xpath('//li/b'); i = 0
```

Amazon 2


```
# binding
    try:
      binding = r[0].text.strip()
      if binding.startswith("Read"):
        i = 1; binding = r[i].text.strip()
      if binding.endswith(":"): binding = binding[:-1]
    except:
      binding = '*** Unknown'
      print('*** ERROR binding:',ind,amazonID)
    try: pages = int(r[i].tail.strip().split(' ')[0])
    except Exception:
      pages = 0
      if not (binding in ["MP3 CD", "Audio CD"]):
        print('*** ERROR pages:',ind,amazonID)
# publisher and year
    try:
      if r[i+1].tail is None:
        publisher = '*** Unknown'; year = 0
        print('*** ERROR year 0:',ind,amazonID)
      else:
        j = r[i+1].tail.rfind('(')
        if j < 0:
          publisher = r[i+1].tail.strip(); year = 0;
          print('*** ERROR year 1:',ind,amazonID)
        else:
          publisher = r[i+1].tail[:j].strip()
          rest = r[i+1].tail[j+1:-1]
          j = rest.rfind(',')
          if j < 0: j = 0
          year = rest[j+1:].strip()
          if len(year)>4:
            j = year.rfind(' '); year = year[j+1:]
```

Amazon 3


```
            try: year = int(year)
            except:
              year = 0; print('*** ERROR year 2:',ind,amazonID)
        except Exception:
          publisher = '*** Unknown'; year = 0
          print('*** ERROR year 3:',ind,amazonID)
    # Prices
        price = listprice = -1
        try:
          pf = book.find('class="priceLarge">')
          if pf > 0:
            temp = book[pf+20:pf+30]
            price = float(temp[:temp.find('<')])
          pf = book.find('class="listprice">')
          if pf > 0:
            temp = book[pf+19:pf+29]
            listprice = float(temp[:temp.find('<')])
        except Exception: pass
    # Look for Similar Items by Subject
        d = html.xpath('//div/div/form/input')
        L = []
        for i in range(len(d)):
          if 'name' in d[i].attrib:
            if d[i].attrib['name'] == "field+keywords":
              L.append(d[i].attrib['value'])
        subjects = "§".join(L)
```

Amazon 4


```
# Customers Who Bought This Item Also Bought
    b = html.xpath('//div/ul/li/div/a')
    L = []
    for i in range(len(b)):
      j = b[i].attrib['href'].rfind('/')
      amID = b[i].attrib['href'][j+1:]
      if len(amID)==10:
        if amID in K.keys(): jnd = K[amID]
        else:
          num += 1; jnd = num; K[amID] = num; Q.append(amID)
        L.append(jnd)
    dat.writerow([ind,len(Q),len(K),amazonID,authors,title,
      publisher,year,binding,pages,listprice,price,subjects] + L)

## harvesting
  while Q:
    amazonID = Q.pop(0); ind = K[amazonID]
    if ind % 50 == 0:
      print(ind,amazonID,datetime.datetime.now().ctime(),len(Q),len(K))
    bookURL = "http://www.amazon.com/dp/"+amazonID+"/"
    step = 0; delay = 60 # seconds
    while step < 3:
      try:
        with urllib.request.urlopen(bookURL) as f:
          book = f.read().decode('iso-8859-1','replace'); break
      except Exception:
        step += 1; print('*** RETRY open:',step,ind,amazonID)
        time.sleep(delay); delay = 10*delay
    if step>=3:
      print('*** ERROR open:',ind,amazonID); continue
    if len(book) < 500:
      print('*** ERROR length:',ind,amazonID); continue
    try: inspect(book)
    except inspectError: continue
```

Amazon 5


```
def amazon(csvName,saveName,visited):
  """Program amazon(csvName,saveName,visited) is harvesting the book data
from the Amazon. The collected data are stored on csv file csvName.
To interrupt it use Ctrl-C. The data for resuming collection starting
at book visited are stored on file saveName.

V. Batagelj, May 2012"""
  global Q, K, num, ind, dat
  print("Amazon book harvesting", datetime.datetime.now().ctime())
  try:
    save = open(saveName,'r')
    K = {}; Q = []
    for line in save.readlines():
      L = line.strip().split(" "); bk = L[1][1:-1]; bi = int(L[0]); K[bk] = bi
      if bi >= visited: Q.append(bk)
    save.close()
    print(visited,"/",len(K),"book IDs read\n")
  except:
    Q = ["0521387078", "0312569378", "1451648537", "1452101248", "0385343841",
     "1451627289", "140123206X", "0307408841", "0425243214", "0553801473" ]
    K = { k:(i+1) for (i,k) in enumerate(Q)}
  num = len(K)
  datFile = open(csvName,'w',newline='',encoding='utf-8')
  dat = csv.writer(datFile,delimiter=';',quotechar='"',
    quoting=csv.QUOTE_NONNUMERIC)
  try:
    harvest()
    print('\nHarvesting finished', datetime.datetime.now().ctime())
  except:
    print("\nHarvesting interrupted", datetime.datetime.now().ctime(),
      "\n at book:", ind, "/", num)
  save=open(saveName,"w")
  for x in sorted(K,key=K.get): save.write(str(K[x])+' "'+x+'"\n')
  save.close()
  datFile.close()
```

Amazon 6


```
print(amazon.__doc__)
if __name__ == '__main__':
  if len(sys.argv)==4: amazon(sys.argv[1],sys.argv[2],sys.argv[3])
  else:
    os.chdir('C:\\Users\\Batagelj\\test\\python\\2012\\amazon')
    amazon('books.csv','save.dat',20)
else:
  amazon('booksX.csv','saveX.dat',0)
```

- izvajanje programa lahko prekinemo s Ctrl-C. Zato imajo notranji prestrezni stavki obliko `except Exception:`, ki ne prestreza prekinitev; glavni prestrezni stavek v metodi `amazon` pa obliko `except:`, ki prestreže vse.
- vgrajena je možnost za nadaljevanje zbiranja po prekinitvi: podatki potrebni zanj se shranijo na datoteko `save.dat`. Ob zaključku programa se izpiše tudi številka knjige, pri kateri je prišlo do prekinitve, `visited`, ki jo potrebujemo pri ponovnem zagonu zbiranja.
- v začetni različici je bil na več mestih, sedaj v metodi `inspect`, uporabljen stavek `continue` na glavno zanko, sedaj v metodi `harvest`. Pri razbitju na dve metodi je bila vpeljana uporabniška izjema `inspectError`. Stavki `continue` so bili nadomeščeni s stavki `raise inspectError()`.

The R Project for Statistical Computing

```
> 3+5
[1] 8
> a <- c(3,1,5); c(2,4,1) -> b
> a
[1] 3 1 5
> b
[1] 2 4 1
> a+b
[1] 5 5 6
> a*b
[1] 6 4 5
> sqrt(a)
[1] 1.732051 1.000000 2.236068
> c <- c(2,3)
> c
[1] 2 3
> a+c
[1] 5 4 7
Warning message:
In a + c : longer object length is not a multiple of shorter object length
> a+1
[1] 4 2 6
> help(sqrt)
starting httpd help server ... done
> a < 4
[1]  TRUE  TRUE FALSE
> a[2]
[1] 1
> a
[1] 3 1 5
> a[a<4]
[1] 3 1
> koren <- function(x) return(x**0.5)
> koren(4)
[1] 2
```

R/Amazon – branje razpredelnice


```
> help(read.csv)
> getwd()
[1] "C:/Users/Batagelj/test/python/2012/amazon"
> setwd("C:/Users/Batagelj/test/python/2012/amazon")
> dat <- read.csv2("booksT.csv",header=FALSE,stringsAsFactors=FALSE)
> dim(dat)
[1] 16804    23
> names(dat)
 [1] "V1"  "V2"  "V3"  "V4"  "V5"  "V6"  "V7"  "V8"  "V9"  "V10" "V11" "V12" "V13" "V14" "V15"
[16] "V16" "V17" "V18" "V19" "V20" "V21" "V22" "V23"
> dat[c(3,7),]
  V1 V2 V3          V4                                          V5                    V6
3  3 30 33 1451648537                              Walter Isaacson                Steve Jobs
7  7 53 60 140123206X Scott Snyder, Jock, Francesco Francavilla Batman: The Black Mirror
                                                    V7   V8        V9 V10   V11   V12
3 Simon & Schuster; First Edition ~1st Printing edition 2011 Hardcover 656  35.0 16.85
7                                          DC Comics 2011 Hardcover 304 29.99  16.8
                                 V13
3 Biography/Autobiography§1955-2011§Biography§Businessmen§Computer engineers§Jobs, Steve,United
7 Comic books, strips, etc§Graphic novels§Comics & Graphic Novels§Comics & Graphic Novels / Supe
  V14 V15 V16 V17 V18 V19 V20 V21 V22 V23
3  26  27  28  29  30  31  27  32  26  33
7  54  55  56  57  58  59  54  55  56  60
>
------------------------
V1   index    V4   AmazonID    V7   publisher     V10  pages            V13  subjects
V2   lenQ     V5   authors     V8   year          V11  listPrice     V14-V23  neighbors
V3   lenK     V6   title       V9   binding       V12  price
------------------------
> year <- dat$V8
> summary(year)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
      0    2002    2008    1970    2011    2013      17
```

R/Amazon – čiščenje in prikazi podatkov


```
> year <- dat$V8; pages <- dat$V10; binding <- dat$V9; price <- dat$V12
> isNA <- which(is.na(year)|is.na(pages)|is.na(binding)|is.na(price))
> year <- year[-isNA]; pages <- pages[-isNA]; binding <- binding[-isNA]
> typeof(price)
[1] "character"
> price <- as.numeric(price[-isNA])
> OK <- (0<pages)&(pages<2050) & (1900<year)&(year<2013) & (0<price)&(price<2000)
> table(OK)
OK
FALSE  TRUE
 1759 15028
> pages <- pages[OK]; binding <- binding[OK]; year <- year[OK]; price <- price[OK]
> bind <- rep(3,length(binding))
> bind[binding %in% c("Paperback", "Perfect Paperback", "Mass Market Paperback")] <- 1
> bind[binding %in% c("Hardcover", "Bonded Leather", "Leather Bound", "Hardcover-spiral")] <- 2
> table(bind)
> plot(density(pages))
> plot(density(year))
> plot(density(price[(0<price)&(price<60)]))
> plot(pages,price,col=c("red","blue","green")[bind],pch=16,cex=0.1)
```

R/Amazon – predelava v Pajkovo omrežje


```
> setwd("C:/Users/Batagelj/test/python/2012/amazon")
> dat <- read.csv2("books.csv",header=FALSE,stringsAsFactors=FALSE)
> dim(dat)
[1] 159494     23
> year <- dat$V8; pages <- dat$V10; binding <- dat$V9; price <- dat$V12
> isNA <- which(is.na(year)|is.na(pages)|is.na(binding)|is.na(price))
> year <- year[-isNA]; pages <- pages[-isNA]; binding <- binding[-isNA]
> price <- as.numeric(price[-isNA])
> OK <- (0<pages)&(pages<2050) & (1900<year)&(year<2014) & (0<price)&(price<2000)
> table(OK)
OK
 FALSE   TRUE
 26196 133258
> pages <- pages[OK]; binding <- binding[OK]; year <- year[OK]; price <- price[OK]
> bind <- rep(3,length(binding))
> bind[binding %in% c("Paperback", "Perfect Paperback", "Mass Market Paperback")] <- 1
> bind[binding %in% c("Hardcover", "Bonded Leather", "Leather Bound", "Hardcover-spiral")] <- 2
> save(pages,year,bind,price,file="sprem.Rdata")
> nod <- read.csv("save.dat",header=FALSE,stringsAsFactors=FALSE,sep=" ")
> n <- nrow(nod)
> m <- nrow(dat)
> net <- file("amazon.net","w"); cat("*vertices",n,"\n",file=net)
> ind <- nod$V1; amazonID <- nod$V2
> for(i in 1:n) cat(ind[i],' "',amazonID[i],'"\n',sep='',file=net)
> cat("*arcslist\n",file=net)
> for(i in 1:m) cat(ind[i],as.vector(na.omit(unlist(dat[i,14:23]))),'\n',file=net)
> close(net)
> nam <- file("amazon.nam","w"); cat("*vertices",n,"\n",file=net)
> authors <- dat$V5; tit <- dat$V6; mi <- ind[m]
> for(i in 1:n) {
+   if (ind[i]>mi) title <- amazonID[i] else
+     title <- paste(substr(authors[i],1,12),'\\n',substr(tit[i],1,12),sep='')
+   cat(ind[i],' "',title,'"\n',sep='',file=nam)}
> close(nam)
```

Pajek in 3d

Pajek / wiki
Pajek / download

d3.js – Data-Driven Documents
Force-directed graph drawing

Json

JSON.org
Wikipedia
W3schools
rJson
Json / Python
primer

```
{"nodes":[{"name":"Myriel","group":1},{"name":"Napoleon","group":1}, ...
{"name":"Brujon","group":4},{"name":"Mme.Hucheloup","group":8}],
"links":[{"source":1,"target":0,"value":1},{"source":2,"target":0,"value":8}, ...
{"source":76,"target":48,"value":1},{"source":76,"target":58,"value":1}]}
```

Pretvorba Pajkovih datotek NET in CLU v JSON

```
setwd("C:/Users/Batagelj/test/python/2012/amazon")
library(rjson)

net2json <- function(netF,cluF,jsonF){
  net <- file(netF,"r");  clu <- file(cluF,"r")
  b <- unlist(strsplit(readLines(net,n=1)," "))
  n <- as.integer(b[length(b)])
  N <- readLines(net,n=n);  nam <- character(n)
  for(i in 1:n) nam[i] <- unlist(strsplit(N[i],'"'))[2]
  skip <- readLines(clu,n=1);  C <- as.integer(readLines(clu,n=n))
  skip <- readLines(net,n=1);  L <- readLines(net,n=-1)
  M <- matrix(as.integer(unlist(strsplit(sub('^\\s+','',L),'\\s+'))),ncol=3,byrow=TRUE)
  nods <- vector('list',n)
  for(i in 1:n) nods[[i]] <- list(name=nam[i],group=C[i])
  m <- nrow(M); lnks <- vector('list',m)
  for(i in 1:m) lnks[[i]] <- list(source=M[i,1]-1,target=M[i,2]-1,value=M[i,3])
  data <- list(nodes=nods,links=lnks)
  jstr <- toJSON(data)
  json <- file(jsonF,"w");  cat(jstr,file=json)
  close(json);  close(net); close(clu)
}

net2json("islands.net","islands.clu","islands.json")
```

primer

ID3 / MP3


```
# MP3 - metadata
#    http://en.wikipedia.org/wiki/ID3
#    http://www.id3.org/id3v2.3.0
# Vladimir Batagelj, April 2011

def getHead(s):
   return { "version" : 'ID3.2.'+str(s[3])+'.'+str(s[4]),
      "flags" : bin(s[5]),
      "size" : ((int(s[6])*128+int(s[7]))*128+int(s[8]))*128+int(s[9]) }

def getFrame(s,si):
   frame = { "Id" : s[si:si+4].decode('ascii'), "size" :
      ((int(s[si+4])*256+int(s[si+5]))*256+int(s[si+6]))*256+int(s[si+7]),
      "Flags1" : bin(s[si+8]), "Flags2" : bin(s[si+9]) }
   fId = frame['Id']
   fEnc = int(s[si+10]); frame['enc'] = fEnc
   st = si+11
   if fId == 'COMM':
      frame['lang'] = s[si+11:si+14].decode('ascii')
      st = s.find(b'\xfe\xff',st)
   if s[st:st+2] == b'\xfe\xff':
      frame['data'] = s[st:si+10+min(200,frame['size'])].decode('utf16')
   else:
      frame['data'] = s[st:si+10+min(200,frame['size'])]
   return frame
```

ID3 / MP3


```
import os
os.chdir('C:\\Users\\Batagelj\\test\\python\\2011\\png')

# files = ["./device.mp3"]
files = [f for f in os.listdir('.') if f.endswith('.mp3')]
for file in files:
   print("\nFile:",file)
   mp3 = open(file,'rb')
   s = mp3.read()
   mp3.close()
   print('file length =', len(s))
   head = getHead(s)
   print(head)
   si = 10
   hsize = head['size']
   while si < hsize:
      print('start:', si)
      frame = getFrame(s,si)
      if frame['Id'] == '\x00\x00\x00\x00': break
      print(frame)
      si = si+10+frame['size']
```

Spletni naslovi

Alphabet soup: * * *
Unicode: * * *
Kode: čas, države, jeziki * * * *
CSV: * * *
netCFD: *
EXIF: *
JPEG: * *
Web of knowledge (deluje na univerzi) *
Gedcom: * *
XML * * * *
JSON: * * * *

Spletni naslovi

DataExpo: *
DASL: *
UCI MLR: * * *
Kaggle *
SURS: * *
The Internet Movie Database (IMDb) * *
Amazon: *
Šah: *
Nogomet: *
Mineralne vode: *

Viri podatkov

http://stat-computing.org/dataexpo/
http://www.unidata.ucar.edu/software/netcdf

Viri besedil

Official and Original Project Gutenberg Web Site and Home Page,
The Online Books Page,
Zbirka slovenskih leposlovnih besedil,
Beseda, Wikivir.