VEŽBA 9

Pregled

XML – eXtensible Markup Language – je jezik za skladištenje i prenos podataka koji već godinama predstavlja veoma važan standard na polju informacionih tehnologija, naročito u oblasti web aplikacija. XML svoju popularnost može da zahvali činjenici da su dokumenti napisani njegovim korišćenjem čitljivi i ljudima i mašinama. XML omogućava opisivanje strukture podataka u obliku višestruko razgranatog stabla, a i sami podaci se skladište zajedno sa opisom strukture u obliku običnog teksta.

U jeziku Python postoji podrška za čitanje i pisanje XML dokumenata. Od raznih rešenja koja postoje u ovim vežbama biće korišćen modul pod nazivom *ElementTree*.

XML

Jezik XML nije programski jezik. On služi za skladištenje podataka umerenog obima na strukturiran način, a ta struktura je stablo. XML dokument je skup elemenata od kojih je svaki imenovan odgovarajućom oznakom – *tag*-om. Jedan jedini element je u na najvišem mestu i hijerarhiji i on se naziva korenom (*root*). Svaki element može da skladišti tekstualni sadržaj i može imati podređene elemente u strukturi stabla – *decu*, a on je tim elementima *roditelj*. Element se identifikuje svojim tag-om, a koristeći tagove nadređenih elemenata može se formirati putanja od korenog elementa do proizvoljnog elementa u strukturi (apsolutna putanja – path). Putanja može biti i relativna u odnosu na neki element koji je niže u hijerarhiji. Osim tag-a, svaki element može imati i proizvoljan broj pridruženih *atributa* od kojih svaki može imati proizvoljan string kao vrednost.

Sam jezik XML je relativno jednostavan. Svaki element počinje navođenjem tag-a unutar znakova < i > (otvarajući), a završava se navođenjem istog tag-a sa znakom / ispred sebe (zatvarajući tag). Otvarajući tag može da sadrži i atribute u obliku parova `atribut="vrednost"`. Unutar elementa (između otvarajućeg i zatvarajućeg tag-a) navodi se sadržaj i proizvoljan broj elemenata nižih u hijerarhiji.

Modul ElementTree

Ovaj modul definiše dve klase koje su od velike važnosti za rad sa XML dokumentima: ElementTree i Element. U Python programe se uključuje importovanjem modula `xml.etree.ElementTree`. (U starijim verzijama sistema modul koji treba importovati je `elementtree.ElementTree`.)

Klasa `Element`

Objekti ove klase predstavljaju elemente stabla. Jedan dokument obično sadrži veliki broj objekata ove klasa, a među objektima postoji hijerarhija – svaki ima jedan element roditelj i neograničen broje elemenata dece.

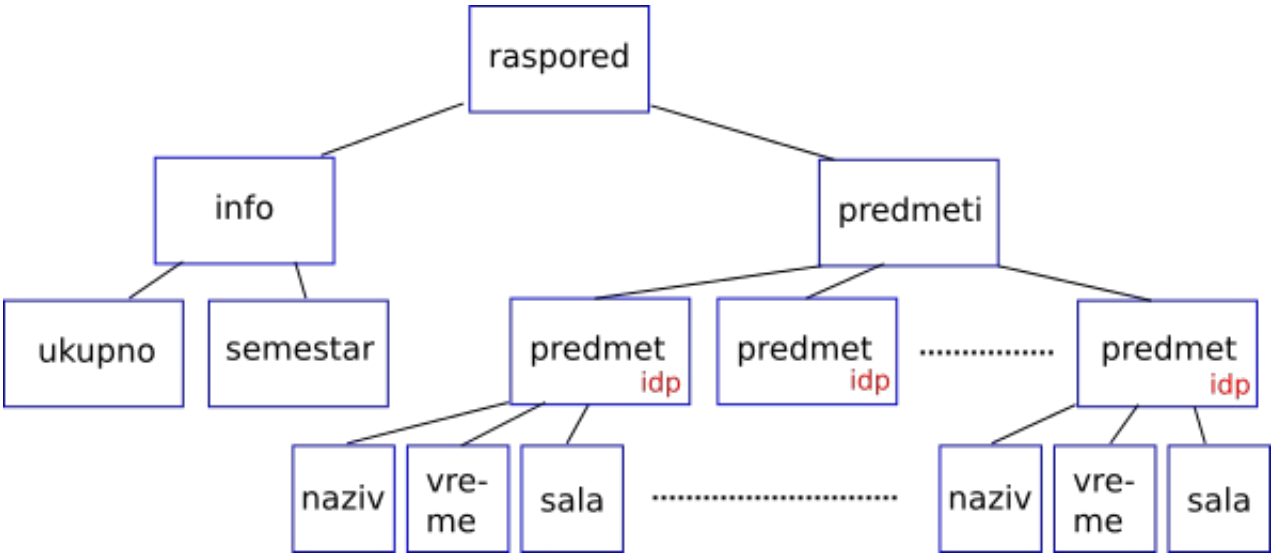
Klasa `ElementTree`

Objekat ove klase predstavlja ceo XML dokument. Povezuje se sa korenim elementom stabla koji je

pripada klasi `Element`. Pozivanjem metoda objekata ove klase moguće je na osnovu XML dokumenta formirati stablo u memoriji, a takođe je moguće konvertovati stablo iz memorije u tekstualni oblik i zapisati ga u fajl – XML dokument.

Zadaci

1. Napisati ručno u text editoru XML dokument koji opisuje situaciju sa slike 1. Kao podatke koristiti predmete koje slušate u tekućem semestru. Testirati ispravnost fajla njegovim učitavanjem u web pretraživač. Pretraživač će prijaviti tačnu lokaciju prve greške u XML dokumentu ukoliko on sadrži sintaksnu grešku (engl. *not well formed*). Crvena oznaka u polju zači da to polje treba da ima atribut pod tim nazivom, a vrednost treba da bude broj sa pet cifara i obaveznim vodećim nulama koji broji od 0 naviše.



Slika 1

2. Otvoriti dokument `consoles.xml`[1] u tekst editoru ili web pretraživaču i proučiti njegov format. Na papiru nacrtati strukturu (stabla) radi lakšeg snalaženja.

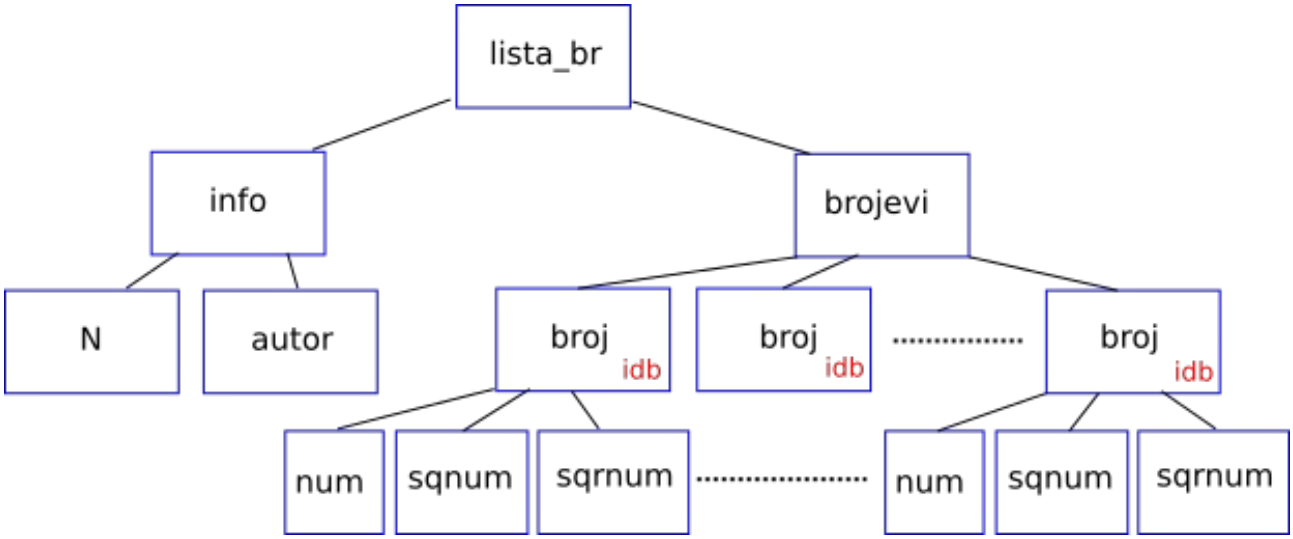
   Napisati program koji na osnovu podataka u dokumentu `consoles.xml` ispisuje listu imena svih proizvođača konzola.

3. Proširiti program iz prethodnog zadatka tako da se za svakog pronađenog prozivođača ispisuje i zemlju porekla, kao i vrednost pripadajućeg `id` atributa u formatu kao u primeru ispod:

   ```
   Sony, Japan, id: 0004
   ```

4. Napisati program u Python-u koji formira stablo koje je u skladu sa šemom na slici 2. Korisnik preko terminala unosi N proizvoljnih brojeva $n_i, n=1..N$, a XML dokument se formira u skladu sa tim brojevima. Podatak `<num>` je sam broj n_i , `<sqnum>` je n_i^2 , a `<sqrnum>` je $\sqrt{n_i}$. Crvena oznaka u polju zači da to polje treba da ima atribut pod tim nazivom, a vrednost treba da bude broj sa pet cifara i obaveznim vodećim nulama koji broji od 0 naviše. Stablo proslediti u XML dokument korišćenjem modula ElementTree. Generisani XML dokument proveriti učitavanjem u web pretraživač.

1 Fajl sa ovim dokumentom može se skinuti sa iste web stranice gde se nalazi i praktikum za laboratorijske vežbe



Slika 2

5. **Dodatni (neobavezni) zadatak:** Napisati program u Python-u koji učitava XML dokument `consoles.xml` koji sadrži podatke o raznim proizvođačima konzola za video igre kao i o samim konzolama za video igre. Na osnovu podataka u učitanom dokumentu formirati izveštaj koji navodi sve proizvođače konzola, a ispod imena svakog od njih navodi listu njihovih najpoznatijih konzola. Sledi primer za jednog proizvođača, a na isti način treba navesti i ostale:

```
...
Manufacturer: Sega
 * Sega Master System (1986), generation 3
 * Sega Mega Drive/Genesis (1989), generation 4
 * Sega Saturn (1995), generation 5
 * Sega Dreamcast (1999), generation 6
--------------------
...
```

Dodatak

Metode klase `ElementTree`

Sledi izbor najvažnijih metoda:[2]

- `ElementTree([root_el])` – konstruktor. Ako se argument `root_el` ne navede stvara se prazno stablo. Ako se `root_el` navede, to treba da bude objekat klase `Element` koji će postati koreni element stabla;
- `parse(file)` – formira stablo u memoriji na osnovu XML dokumenta u fajlu. Jedini argument (`file`) može biti fajl objekat (ranije otvoren funkcijom `open`) ili string sa imenom fajla;
- `write(file, encoding)` – zapisuje sadržaj stabla u fajl. Na argument fajl se odnosi isto što i na `file` argument metode `parse`, a `encoding` je string koji određuje tip

2 Za kompletan spisak metoda treba konsultovati dokumentaciju (https://docs.python.org)

kodiranja teksta u XML dokumentu. Skoro bez izuzetka njegov sadržaj je `'UTF-8'`;

- `getroot()` - vraća objekat klase `Element` koji čini koren stabla. Od ovog elementa naniže moguće je pretraživati stablo i koristiti ili modifikovati sadržaj elemenata (videti sledeći odeljak).

Metode i elementi članovi klase `Element`

Najvažnije metode:

- `Element([tag])` – konstruktor;
- `set(attrib, value)` – postavljanje jednog atributa elementa;
- `get(attrib)` – čitanje jednog atributa elementa;
- `append(child)` – dodavanje jednog deteta;
- `find(xpath)` – pretraga po podređenim elementima (deci) i vraćanje jednog (prvog po redu) elementa koji odgovara specifikaciji. Za argument `xpath` videti sledeći odeljak;
- `findall(xpath)` – pretraga kao i prethodna funkcija, ali vraća se lista sa svim elementima koji odgovaraju specifikaciji. **Pažnja:** ni ova ni prethodna funkcija ne pretražuje sadržaj elemenata zadatih tagovima na određenoj putanju. Vraćeni element ili listu elemenata treba dodatno proveriti sadrži li potrebne podatke ili atribute.

Najvažniji elementi članovi:

- `text` – string koji odgovara sadržaju elementa;
- `tag` – string koji odgovara tag-u elementa (ime elementa);
- `attrib` – rečnik gde su ključevi stringovi koji odgovaraju imenu atributa, a pridružene vrednosti su stringovi sa sadržajem (vrednostima) atributa.

Pretraga metodama `find` i `findall`

Pošto je XML dokument strukture stabla kao i fajl sistem na spoljašnjoj memoriji računarskog sistema, svaki element XML dokumenta se može shvatiti kao određeni fajl u hijrarhiji sa imenom koji odgovara tag-u. Na taj način se može formirati apsolutna ili relativna putanja (path) svakog elementa koja se navodi kao argument `xpath` odgovarajućih funkcija (metoda) za pretragu.

U `xpath` specifikaciji postoji nekoliko specijalnih karaktera. Slede najvažniji sa primerima:

- `.` – aktuelni element. Obično se koristi na početku putanje kao indikacija da je u pitanju relativna putanja, npr. `'./lista/objekat'`;
- `*` – svi podređeni elementi (deca), npr. `'*/objekat'` odnosi se na sve elemente koji su podređeni svim podređenim elementima (unuci) trenutno aktuelnog elementa, a tag im je `objekat`;
- `//` – svi podređeni elementi na svim nivoima ispod aktuelnog elementa, npr. `'.//objekat'` se odnosi na sve elemente podređene na bilo kom nivou trenutno aktualnom elementu, a imaju tag `objekat`.

Namespace-ovi (kvalifikovana imena) tag-ova i attributa[3]

Postoje mnoge specifikacije dokumenata koje se oslanjaju na jezik XML. Mnogi projekti koriste te specifikacije i proširuju ih, pa postoji opasnost da razna proširenja unesu u specifikaciju isti tag sa različitim značenjima. Da bi se taj problem prevazišao u jezik XML uvedene su oblasti važenja imena (tagova i atributa) – Namespace-ovi. U tom smislu bilo koji tag ili atribut nije više samo jedno ime, nego par – namespace (ns) i ime, npr. `<kelftn:predmet>`. Ovako zapisano me `predmet` ovim naziva se potpuno kvalifikovano (*fully qualified*), tj. svrstano u određeni namespace, u ovom slučaju `kelftn`. Ovo za rezultat ima da će ime `predmet` koje pripada drugom namespace-u (npr. `<eureka:predmet>`) važiti za potpuno drugo ime.

Opisana situacija je nešto komplikovanija i zahteva dodatno objašnjenje. Imena namespace-ova navedena u gornjim primerima su zapravo samo oznake, odnosno skraćena imena i važe samo u onom delu XML dokumenta u kojem su definisana. U praksi se, ipak, skoro uvek definišu u korenom elementu, pa važe za ceo dokument. Definicija se obavlja putem atributa elementa unutar kojeg važe i u tu svrhu služi atribut sa specijalnim zančenjem – `xmlns`. Zadaje se u obliku `xmlns:ns = URI`, gde je URI (*Uniform Resource Indetifier*) vezan za organizaciju koja je dati namespace uvela. (Ako se dati URI zada kao adresa u Web pretraživaču, dobiće se informacije vezane za dati namespace.) U gronjem primeru će se dalje `ns` koristiti za kvalifikaciju imena unutar elementa u kojem je definisan u obliku `<ns:ime>`. Ako se nikakva oznaka ne stavi uz neko ime, to znači da je to ime u tzv. default namespace-u, ali se default namespace mora zadati kao xmlns atribut bez posebne oznake. Sledi primer:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<rootelem xmlns=”http://www.elektronika.ftn.uns.ac.rs/kelftn”
xmlns:kelftn=”http://www.elektronika.ftn.uns.ac.rs/kelftn”
xmlns:eureka=”http://www.fictive-eureka.com/eureka”>
     <list eureka:atx=”A”>
          <kelftn:elx kelftn:no=”01”>Informacija 1<\kelftn:elx>
          <elx no=”02”\>
     </list>
     <eureka:elist>
     </eureka:elist>
</root>
```

Definisana su dva namespace-a: `kelftn` i `eureka`. Default je isti kao i `kelftn`, pa ako se ništa ne napiše uz neko ime, taj se podrazumeva. Dakle tagovi `list`, `elx` i atribut `no` su u namespace-u `kelftn`, dok tag `elist` i atribut `atx` pripadaju namespace-u `eureka`.

Rukovanje namespace-ovima pomoću modula ElementTree

Svi objekti iz modula ElementTree ispred imena koja pripadaju određenom namespacu dodaju URI koji definiše taj namespace unutar velikih zagrada. Na taj način string sa tagom `list` iz gornjeg primera biće

```
'{http://www.elektronika.ftn.uns.ac.rs/kelftn}list'
```

Ovo će se desiti bez obzira pripada li određeno ime default namespace-u ili nekom drugom. O ovome treba voditi računa prilikom pretraga ili izmena podataka.

3 U ovoj vežbi, namespace-ovi se ne koriste, ali u mnogim primanama u praksi su prisutni. Već od sledeće vežbe razumevanje ovog sistema biće neophodno.