Laboratorium 12 — Ćwiczenia w odczycie i zapisie plików tekstowych, binarnych i XML.

Zapis i odczyt danych do/z pliku binarnego

Aby móc zapisać dane do pliku binarnego, konieczne jest przekonwertowanie ich na ciąg bitów (typ ten jest odpowiednikiem string-a znanego z C++ i może on przechowywać tekst zgodny z kodowaniem ASCII, od swojego odpowiednika wspierającego Unicode odróżnia go litera b z przodu, np. b'tekst_ascii', w przypadku języka Python 2.0 typ ten był domyślnym typem tekstowym). W tym celu, należy posłużyć się poleceniem `pack` z modułu `struct`. Aby je zastosować należy wskazać odpowiedni modyfikator przypisany typom danych z języka `C`, np. `i` odpowiada 4-bajtowemu integerowi, `d` to 8-bajtowy float.

```
a = 7
struct.pack('i', a)
```

co daje:

```
b'\x07\x00\x00\x00'
```

W celu odpakowania/odczytu należy skorzystać z polecenia `unpack`, które zwraca `tuple`, nawet jeśli odczytujemy jedną wartość.

```
struct.unpack('i', b'\x07\x00\x00\x00')
```

co daje:

```
(7,)
```

Ponieważ język Python pozwala na zapis liczb całkowitych o dowolnej wielkości, powyższa technika może nie zadziałać. W takim przypadku, konieczne jest skorzystanie z metody klasy `int` wprowadzonej w wersji `3.2`.

```
a = 10123**16
print(a)
b = a.to_bytes(30, 'big')
print(b)
c = int.from_bytes(b, byteorder = 'big')
print(c)
```

wynik:

```
12160398008385734478135195046302226613998462680273452528699540161
b'\x00\x00\x00\x1d\x8fo+\x9a\xd0\xcf\x99BXx,b\xf0\x96\x8a\x18\x8e\\\xb1\rC\
    x1eXq\x82\xc1'
12160398008385734478135195046302226613998462680273452528699540161
```

Gdy mamy do czynienia z napisem, zamiast `pack` i `unpack`, w celu konwersji, można wykorzystać konstruktor klasy `bytes` oraz metodę klasy `bytes decode`, co umożliwi podanie oczekiwanego kodowania.

```
s = 'informatyka'
p = bytes(s, encoding='raw_unicode_escape')
print(p)
s2 = p.decode('raw_unicode_escape')
print(s2)
```

wynik:

```
b'informatyka'
informatyka
```

Zadanie 1 Zapisz do pliku binarnego o nazwie `zad1.bin` dane z Zadania 3, z poprzednich zajęć. Pamiętaj, aby w przypadku stringów zapisać także ich rozmiary (podpowiedź: użyj funkcji `pack` z modułu `struct` oraz funkcji `bytes` z parametrem `raw_unicode_escape`).

Zadanie 2 Wczytaj dane z pliku binarnego o nazwie `zad1.bin` — plik z Zadania 1 (podpowiedź: przydatna może się okazać funkcja `unpack` z modułu `struct`).

Format XML

XML to uniwersalny język znaczników przeznaczony do reprezentowania różnych danych w strukturalizowany sposób i ułatwia wymianę dokumentów pomiędzy różnymi systemami.

W Pythonie do obsługi plików XML można się posłużyć modułem `xml.dom.minidom` oraz `xml.etree.ElementTree`. Na laboratoriach skupimy się w szczególności na drugim z nich.

Odczyt plików XML

Aby odczytać dane z pliku xml należy na początku użyć funkcji `parse` w celu utworzeniu struktury drzewa i uzyskania korzenia (eng. root). Po tych czynnościach można w łatwy sposób przemierzać drzewo, które tworzy graf spójny:

```
drzewo = xml.etree.ElementTree.parse(src)
korzen = drzewo.getroot()
```

Aby wypisać wszystkie atrybuty należy skorzystać z pola `element.attrib` — element implementuje `__iter__`, pozwalając na łatwe przejście po zagnieżdżonych elementach. Natomiast do wypisania wartości elementów `element.text`. Do potomków elementu można odwoływać się także jak do zwykłej tablicy.

Zadanie 3 Wczytaj plik `pierwszy.xml`, a następnie wypisz z niego wszystkie atrybuty dla elementów na trzecim poziomie.

Zadanie 4 Wczytaj plik `pierwszy.xml`, a następnie wypisz z niego wszystkie wartości elementów na trzecim poziomie.

Zadanie 5 Odwołując się bezpośrednio do korzenia, wyświetl atrybut i jego wartość dla drugiego dziecka jego bezpośredniego pierwszego potomka.

Zapis plików XML

Do tworzenia nowego pliku xml przydatne są metody:

- `xml.etree.ElementTree.Element(arg)` — tworzenie elementu głównego,
- `xml.etree.ElementTree.SubElement(arg1, arg2)` — tworzenie elementu potomnego,
- `.set(arg1, arg2)` — dodanie atrybutu dla danego węzła,
- `.text = arg` — dodanie wartości atrybutu dla danego węzła,
- `xml.etree.ElementTree.tostring(data).decode('utf-8')` — tworzy testową reprezentację dokumentu xml.

Po powyższych czynnościach, należy zapisać utworzoną reprezentację do pliku.

Zadanie 6 Utwórz i zapisz do pliku o nazwie `uczelnia.xml` poniższą strukturę.

```
<root>
    <uczelnia Nazwa="SIMS">
        <student Nrindeksu="12333">Anna Kowalska</student>
        <student Nrindeksu="23795">Przemyslaw Nowak</student>
        <student Nrindeksu="65789">Piotr Pilawka</student>
    </uczelnia>
</root>
```

Przeszukiwanie plików XML

Biblioteka ta zawiera także funkcje do znajdowania węzłów o zadanych parametrach:

- `find`,
- `findall`,
- `findtext`.

Zadanie 7 Na podstawie bazy sklepu w pliku `sklep.xml`, gdzie `item` oznacza towar fizyczny, a `vitem` wersje cyfrowe, wypisz i zlicz wszystkie wystąpienia egzemplarzy cyfrowych.

Zadanie 8 W pliku z Zadania 7 znalazł się błąd w nazwie wydawnictwa książki pt. „Python3". Popraw błąd (oriley → oreilly) i zapisz poprawioną wersję pliku jako `sklep2.xml`.