ІНФОРМАТИКА ТА ПРОГРАМУВАННЯ

Тема 28. XML та JSON

XML та JSON

- При спілкуванні програм у мережі, окрім стандартів взаємодії самих програм – мережних протоколів, - велику роль грають також стандарти обміну даними.
- На сьогодні застосовують два домінуючих стандарти обміну даними у мережі: XML та JSON.
- Звичайно, ці стандарти можуть не тільки визначати формат обміну у мережі, але й організацію файлів на локальному комп’ютері.
- Розглянемо спочатку більш простий стандарт – JSON.

JSON

- JSON (JavaScript Object Notation або об’єктна нотація JavaScript) – це нескладний формат обміну даними.
- Дані у форматі JSON легко читати людині а також аналізувати за допомогою програм.
- JSON базується на двох структурах: об’єкт та список (масив).
- Об’єкт складається з пар <ключ> : <значення>.
- Ці пари розділяються комами, а сам об’єкт береться у фігурні дужки { }.
- У Python прямим аналогом об’єкту JSON є словник.
- Список містить послідовність значень, що розділяються комами. Список береться у квадратні дужки [ ].
- У Python прямим аналогом списку JSON є список.
- Значеннями у об’єктах або списках можуть бути рядки (беруться у подвійні лапки “ “), числа, бульові значення (позначаються true, false) а також null (аналог None у Python).

JSON.2

- Приклад документу JSON – нижче:

```
[
    {
        "question" : "Чи є цикл повторенням деякої інструкції?",
        "type" : "yesno",
        "points" : 1
    },
    {
        "question" : "Що таке хоарівська трійка?",
        "type" : "onlyone",
        "points" : 2
    }
]
```

Засоби обробки даних JSON у Python

- Python містить засоби для перетворення даних JSON у формат внутрішніх об’єктів Python та навпаки, об’єктів Python у дані JSON.
- Ці засоби зібрано у модулі json.
- Для завантаження даних JSON з текстового файлу використовують функцію

```
a = json.load(f)
```

  - де f – відкритий для читання текстовий файл, що містить дані JSON.
- Для завантаження даних JSON з рядка s використовують функцію

```
a = json.loads(s)
```

- Після виклику однієї з цих функцій a набуде значення словника або списку в залежності від структури даних JSON.
- Усі об’єкти JSON перетворюються у словники Python, списки JSON - у списки Python, дані простих типів – у відповідні дані простих типів Python.

Засоби обробки даних JSON у Python.2

- Для запису даних JSON у текстовий файл використовують функцію

```
json.dump(a, f, ensure_ascii=True, indent=None,
sort_keys=False)
```

  - де a – об’єкт Python (словник або список), f – відкритий для запису текстовий файл, що містить дані JSON.
  - Параметр ensure_ascii означає перетворення символів Unicode у їх коди.
  - Параметр sort_keys дозволяє встановити впорядкування ключів у об’єктах за алфавітом.
  - Параметр indent дозволяє задати відступ у задану кількість пропусків при виведенні даних JSON у текстовий файл.
- Для створення рядка з даними JSON використовують функцію

```
s = json.dumps(a, ensure_ascii=True, indent=None,
sort_keys=False)
```

Приклад: телефонний довідник (JSON)

- Створити телефонний довідник, який містить телефони знайомих.
- Реалізувати функції створення нового довідника, додавання одного запису, пошуку телефону за прізвищем а також зміни існуючого номеру телефону.
- Для розв’язання задачі опишемо клас JSONRefBook.
- У файлі JSON записи довідника зберігаються у форматі:

[
{"friend": <name>,
"phone", <phone>},
...
]

- Цей список для зручності обробки треба перетворити у словник.
- Ключ у словнику - значення "friend", а значення - значення "phone".
- При записі треба здійснити обернене перетворення.

Клас JSONRefBook

- Клас JSONRefBook має поля:
  - self.filename - ім'я файлу довідника
  - self.key_field - ключ, що використовується при утворенні словника
  - self.fields_list - список імен полів з файлу JSON
- Клас також має методи: конструктор __init__, createrb – створити довідник, apprb – додати запис, searchrb – знайти телефон, replacerb – замінити телефон.
- Внутрішні методи _list_to_dict та, _dict_to_list виконують перетворення списку у словник та навпаки.
- Основна частина програми забезпечує вибір та виконання однієї з доступних функцій довідника.
- Версія 2 програми відрізняється тим, що не-ascii символи не перетворюються у коди, а записуються, як є.

Приклад: зберігання тестів у файлах JSON. Проходження тестів

- Скласти програму, яка підтримує проходження тестів у веб-браузері.
- Програма повинна забезпечити введення імені та паролю користувача, вибір теми тесту, передачу питань, отримання та аналіз відповідей, а також показ результату тесту (кількості балів).
- Тест може містити питання трьох типів:
  - Так або ні
  - З вибором одного варіанту відповіді
  - З вибором декількох варіантів відповіді
- За кожне питання, на яке дано правильну відповідь, нараховується визначена кількість балів.
- Цю задачу ми розглядали у темі «Побудова веб-серверів».
- Тоді тести зберігались у файлі MS Excel, а результати ми зберігали у текстовому файлі.
- Зараз для збереження тестів, даних користувачів, результатів використаємо JSON.

Проходження тестів. Структура файлів JSON

- Тести зберігаються у файлі JSON та мають таку структуру:

```
[
    {
    "quiz" : ,<назва тесту>,
    "questions" : [ {
                    "question" : <питання>,
                    "type" : <тип питання>,
                    "points" : <кількість балів>,
                    "answers" : [
                                    {
                                    "answer" : <відповідь>,
                                    "value" : <чи правильна відповідь>
                                    },
                                    ...
                                ]
                    },
                ]
                ...
    },
    ...
]
```

Проходження тестів. Структура файлів JSON.2

- Імена та паролі користувачів зберігаються у файлі:

```
[
    {
    "user" : <ім'я>,
    "password" : <пароль>
    },
    ...
]
```

- Для кожного проходження тесту будемо зберігати результати у окремому файлі JSON:

```
{
    "maxpoints": <максимальна кількість балів>,
    "points": <набрано балів>,
    "quiz": <тема тесту>,
    "user": <користувач>,
    "when": <дата та час проходження>
}
```

Проходження тестів. Реалізація.

- Виберемо реалізацію з використанням WSGI.
- Щоб реалізувати тести зі збереженням даних у JSON, достатньо описати клас читання/запису даних тестів, який буде нащадком абстрактного класу TestIO, а також описати головний модуль.
- Назвемо клас, який реалізує інтерфейс читання/запису набору тестів з файлів JSON, TestJSONIO.
- Окрім реалізації абстрактних методів та властивостей класу TestIO, цей клас також має внутрішні методи _readws (прочитати аркуш робочої книги) та _readquiz (прочитати один тест).
- Головний модуль створює об’єкт класу QuizApplication та передає йому потрібні параметри.
- Також у цьому модулі ініціюється робота веб-сервера WSGI.

XML

- XML (eXtensible Markup Language –розширювана мова розмітки) є мовою розмітки, яка визначає набір правил для кодування документів у форматі, який є одночасно зручним для сприйняття людиною і комп'ютером.
- XML схожий на HTML тим, що структура документу визначається тегами: початковими та кінцевими.
- Тег береться з обох боків у кутові дужки: < >.
- Кожний тег має власне ім'я.
- Кінцевий тег має таке ж ім'я, як і відповідний початковий, але відрізняється від початкового тим, що починається з косої риски '/'.
- Як і у HTML, теги XML можуть мати атрибути.
- На відміну від HTML, теги XML не є фіксованими, отже мова є розширюваною.
- Документ XML має ієрархічну структуру та є деревом, оскільки може містити тільки один кореневий тег.
- Кожен тег визначає окремий вузол дерева.

XML.2

- Приклад документу XML – нижче:

```
<countries total="2">
    <country id="AFG">
        <iso2Code>AF</iso2Code>
        <name>Afghanistan</name>
        <region id="SAS">South Asia</region>
        <capitalCity>Kabul</capitalCity>
        <longitude>69.1761</longitude>
        <latitude>34.5228</latitude>
    </country>
    <country id="ARG">
        <iso2Code>AR</iso2Code>
        <name>Argentina</name>
        <region id="LCN">Latin America & Caribbean</region>
        <capitalCity>Buenos Aires</capitalCity>
        <longitude>-58.4173</longitude>
        <latitude>-34.6118</latitude>
    </country>
</countries>
```

Області імен XML

- Теги XML можуть належати до різних областей імен (namespaces).
- Область імен має власне ім’я та адресу у мережі (URL).
- Якщо тег t входить до області імен ns, це позначають так: <ns: t>.
- Області імен дозволяють розрізняти теги з однаковим ім’ям але різним смисловим навантаженням.
- Наприклад, регіон може означати регіон країн або регіон у деякій країні.
- Ім’я та адреса області імен позначається атрибутом з префіксом xmlns.
- Приклад документу, що визначає область імен, - нижче:

Області імен XML.2

```
<wb:countries xmlns:wb="http://www.worldbank.org" total="2">
    <wb:country id="AFG">
        <wb:iso2Code>AF</wb:iso2Code>
        <wb:name>Afghanistan</wb:name>
        <wb:region id="SAS">South Asia</wb:region>
        <wb:capitalCity>Kabul</wb:capitalCity>
        <wb:longitude>69.1761</wb:longitude>
        <wb:latitude>34.5228</wb:latitude>
    </wb:country>
    <wb:country id="ARG">
        <wb:iso2Code>AR</wb:iso2Code>
        <wb:name>Argentina</wb:name>
        <wb:region id="LCN">Latin America &
Caribbean</wb:region>
        <wb:capitalCity>Buenos Aires</wb:capitalCity>
        <wb:longitude>-58.4173</wb:longitude>
        <wb:latitude>-34.6118</wb:latitude>
    </wb:country>
</wb:countries>
```

Стандарти, пов’язані з XML

- З XML пов’язаний ряд стандартів, які дозволяють розширити використання XML не тільки для обміну даними, але й для інших застосувань.
- Це, зокрема, XSLT, XPath, Schema, DTD.
- XSLT задає трансформацію документа XML у інший формат, наприклад, HTML.
- XPath дозволяє здійснювати пошук у документі XML.
- Schema та DTD використовуються для опису структури документів XML певного класу.
- Такий опис дозволяє перевірити правильність конкретного документу.

DOM та SAX

- Для аналізу та модифікації документів XML використовують два різних підходи, які засновано на стандартах DOM та SAX.
- DOM (Document Object Model – об'єктна модель документу) – це підхід, який базується на представленні документу XML у вигляді дерева у пам'яті комп'ютера.
- Такий підхід дозволяє легко аналізувати та маніпулювати документом XML, але має недоліки при обробці дуже великих документів.
- SAX (the Simple API for XML – простий програмний інтерфейс для XML) для обробки документу використовує модель програмування, що керується подіями.
- Коли зустрічається вузол або текст у документі, SAX ініціює подію та викликає відповідну функцію її обробки.
- Цей підхід дозволяє без проблем обробляти великі документи, але не надає можливості перевірки правильності документу.

Засоби обробки XML у Python

- Python містить декілька модулів для обробки XML на базі DOM, SAX та власної розробки.
- Зокрема:
  - xml.etree.ElementTree: програмний інтерфейс ElementTree, простий та легкий процесор XML
  - xml.dom: реалізація інтерфейсу DOM
  - xml.dom.minidom: мінімальна реалізація DOM
  - xml.sax: базові класи та функції SAX
- Далі у даній темі ми будемо розглядати модуль ElementTree.

Модуль ElementTree

- Модуль xml.etree.ElementTree містить два основних класи для аналізу та модифікації XML: ElementTree та Element.
- ElementTree представляє усе дерево документу XML, а Element – один вузол документу.
- Основні функції модуля зібрані у таблиці:

Функція	Опис
parse(source)	Розібрати документ, що міститься у source. source – це ім'я файлу, або вже відкритий файл, що містить XML. Повертає дерево – об'єкт класу ElementTree.
iterparse(source, events=None)	Створити ітератор, що розбирає документ з source у процесі його читання та повідомляє про події (events), що виникають. source має те ж значення, що й у функції parse. events – це кортеж, який може складатись з рядків "start" (початок елемента), "end" (кінець елемента), "start-ns" (початок області дії імен), "end-ns" (кінець області дії імен). Якщо параметр events не вказано, обробляються тільки події завершення елемента ("end").
fromstring(text)	Розібрати документ, що міститься у рядку text. Повертає кореневий вузол дерева – об'єкт класу Element.
tostring(element, encoding="us-ascii")	Перетворити елемент XML element разом з усіма його піделементами у рядок. encoding – кодування тексту у рядку.

Клас Element

- Основні методи та властивості зібрані у таблиці:

Властивість/метод	Опис
el = Element(tag)	Створити об’єкт класу Element el з тегом tag
el.tag	Повернути тег елемента
el.text	Повернути текст елемента
el.tail	Повернути текст елемента, який йде після внутрішніх елементів (якщо є)
el.attrib	Повернути словник атрибутів елемента
el.clear()	Очистити елемент, видаливши усі піделементи, текст та атрибути
el.get(key, default=None)	Отримати значення атрибута key. Якщо такого атрибуту немає, повертає значення параметра default.
el.set(key, value)	Встановити значення value атрибута key.
append(subelement)	Додати піделемент subelement.
el.find(match, namespaces=None)	Знайти перший піделемент, який відповідає match. match – це тег або вираз XPath. namespaces – словник, що містить умовні імена та адреси областей дії імен

Клас Element.2

Властивість/метод	Опис
el.findall(match, namespaces=None)	Знайти список усіх піделементів, які відповідають match. match – це тег або вираз XPath. namespaces – словник, що містить умовні імена та адреси областей дії імен
el.findtext(match, default=None, namespaces=None)	Повернути текст першого піделемента, який відповідає match. match – це тег або вираз XPath. namespaces – словник, що містить умовні імена та адреси областей дії імен. Якщо піделемент не знайдено, повертає значення параметра default.
el.insert(index, subelement)	Вставити піделемент subelement у позицію index.
el.iter(tag=None)	Створити ітератор, який повертає усі піделементи. Якщо вказано параметр tag, то будуть повертатись тільки елементи з тегом tag.
el.iterfind(match, namespaces=None)	Створити ітератор, що знаходить список усіх піделементів, які відповідають match. match – це тег або вираз XPath. namespaces – словник, що містить умовні імена та адреси областей дії імен
el.itertext()	Створити ітератор, що по черзі повертає текст даного елемента та усіх його піделементів.
el.remove(subelement)	Видалити елемент subelement.

Клас ElementTree

- Основні методи зібрані у таблиці:

Метод	Опис
eltree = ElementTree(element=None)	Створити об’єкт класу ElementTree з кореневого елемента element.
eltree.find(match, namespaces=None)	Те ж саме, що й Element.find(…), починаючи з кореневого елемента дерева
eltree.findall(match, namespaces=None)	Те ж саме, що й Element.findall(…), починаючи з кореневого елемента дерева
eltree.findtext(match, default=None, eltree.namespaces=None)	Те ж саме, що й Element.findtext(…), починаючи з кореневого елемента дерева
eltree.getroot()	Повернути кореневий елемент дерева.
eltree.iter(tag=None)	Те ж саме, що й Element.iter(…), починаючи з кореневого елемента дерева
eltree.iterfind(match, namespaces=None)	Те ж саме, що й Element.iterfind(…), починаючи з кореневого елемента дерева.
eltree.write(file, encoding="us-ascii")	Зберегти дерево XML у файлі. file – ім’я файлу або файл, відкритий для запису. encoding – кодування тексту у файлі.

Приклад: телефонний довідник (XML)

- Створити телефонний довідник, який містить телефони знайомих.
- Реалізувати функції створення нового довідника, додавання одного запису, пошуку телефону за прізвищем а також зміни існуючого номеру телефону.
- Для розв’язання задачі опишемо клас XMLRefBook.
- У файлі XML записи довідника зберігаються у форматі:

```
<refbook>
    <friend name="ім'я">телефон</friend>

    ...
</refbook>
```

Клас XMLRefBook

- Клас XMLRefBook має поля:
  - self.filename - ім'я файлу довідника
- Клас також має методи: конструктор __init__, createrb – створити довідник, apprb – додати запис, searchrb – знайти телефон, replacerb – замінити телефон.
- Основна частина програми забезпечує вибір та виконання однієї з доступних функцій довідника.

Використання JSON та XML для обміну даними у мережі

- Справжню силу JSON та XML демонструють у задачах обміну даними у мережі.
- На сьогодні створено багато серверів, які надають доступ до відкритих даних.
- Відкриті дані – це дані, які можуть бути отримані за допомогою програмного забезпечення.
- Ці дані повертаються у форматах JSON або XML.
- Кожний сайт надає програмний інтерфейс (API) доступу до даних.
- Цей програмний інтерфейс базується, як правило, на HTTP-запитах GET або POST.
- Параметри запитів дозволяють відібрати потрібні дані для подальшого аналізу.
- Більшість сайтів потребує реєстрації програм, що аналізують дані.
- Після реєстрації сайт видає ключ доступу, так званий ApiKey.
- Ключ потрібно вказувати у всіх запитах до сервера.

Веб-сервіси

- XML може використовуватись не тільки для отримання відповіді сервера, але й для формування запиту.
- Прикладом слугують так звані веб-сервіси.
- Веб-сервіси це спеціальний стандартизований програмний інтерфейс (API).
- На сьогодні найбільш розповсюдженими є моделі XML-RPC та SOAP.
- Обидві моделі передбачають передачу XML у запитах до сервера та отримання відповіді у вигляді XML.
- XML-RPC (XML Remote Procedure Call – віддалений виклик процедур у XML) означає виклик підпрограм, які знаходяться на віддаленому комп’ютері.
- У запиті передають ім’я підпрограми та її параметри, а відповідь, у разі успіху, містить результати виконання.

Веб-сервіси.2

- SOAP (Simple Object Access Protocol) також передбачає віддалене виконання підпрограм та повернення результатів, але цей стандарт є набагато складнішим за XML-RPC.
- Веб-сервіси дозволяють будувати великі розподілені системи.
- При цьому, частини систем можуть працювати у різній архітектурі комп’ютерів та використовувати різні мови програмування.
- Також є можливість вбудовувати вже створені веб-сервіси у власні системи.

Приклад: аналіз змін населення по країнах за період

- Нехай необхідно оцінити зміни населення у країнах світу за деякий період та виділити 10 країн з найбільшим приростом населення та 10 країн з найменшим приростом (найбільшим зменшенням) населення.
- Для розв'язання задачі використаємо відкриті дані, що надає Світовий Банк.
- Адреса сайту з описом відкритих даних:
- https://datahelpdesk.worldbank.org/knowledgebase/topics/125589-developer-information
- Світовий Банк, на відміну від багатьох інших серверів відкритих даних, не вимагає реєстрації та не видає ApiKey.
- Отже, для доступу до даних достатньо засвоїти правила формування запитів.
- Дані Світового Банку містять, у тому числі, щорічну інформацію про населення країн світу. Їх ми і будемо аналізувати.

Приклад: аналіз змін населення по країнах за період.2

- Для того, щоб отримати інформацію про населення, треба спочатку отримати інформацію про країни.
- Кожна країна, окрім назви, має двохсимвольний та трьохсимвольний код країни.
- Саме двохсимвольний код застосовують для повернення інформації про населення.
- Також треба зважити на те, що у одному списку з країнами містяться дані про регіони, наприклад, Африка.
- Ці дані треба відділити в рамках нашого аналізу.
- Сервер Світового Банку може повертати дані у форматі XML (за угодою) або JSON.

Приклад: аналіз змін населення по країнах за період (JSON)

- Дані про країни та про населення у форматі JSON поділяються на сторінки.
- Кожна сторінка є списком JSON та містить об’єкт-заголовок та список об’єктів-даних про країни.
- Фрагмент сторінки даних нижче:


```
[{"page":1,"pages":7,"per_page":"50","total":304},
    [
    ...
    {"id":"AFG",
    "iso2Code":"AF",
    "name":"Afghanistan",
    "region":
        {"id":"SAS",
        "value":"South Asia"},
    "adminregion":
        {"id":"SAS",
        "value":"South Asia"},
    "incomeLevel":
        {"id":"LIC",
        "value":"Low income"},
    "lendingType":
        {"id":"IDX",
        "value":"IDA"},
    "capitalCity":"Kabul",
    "longitude":"69.1761",
    "latitude":"34.5228"},
```

Приклад: аналіз змін населення по країнах за період (JSON).2


```
    {"id":"AFR",
    "iso2Code":"A9",
    "name":"Africa",
    "region":
        {"id":"NA",
        "value":"Aggregates"},
    "adminregion":
        {"id":"",
        "value":""},
    "incomeLevel":
        {"id":"NA",
        "value":"Aggregates"},
    "lendingType":
        {"id":"",
        "value":"Aggregates"},
    "capitalCity":"",
    "longitude":"",
    "latitude":""}
    ...
    ]
]
```

Приклад: аналіз змін населення по країнах за період (JSON).3

- У заголовку нас буде цікавити поле "pages", яке містить кількість сторінок документу.
- Для кожної країни ми будемо вибирати її двохсимвольний код "iso2Code" та назву "name".
- Також будемо враховувати, що для регіонів текст у полі "region"/"value" – це рядок "Aggregates".
- Щоб отримати сторінку 3 даних про країни, треба сформувати запит:
- http://api.worldbank.org/countries?format=json&page=3
- Дані про населення містять аналогічний заголовок та список даних про населення країн.
- Фрагмент даних для однієї країни – нижче:

Приклад: аналіз змін населення по країнах за період (JSON).4


```
...
{"indicator":
    {"id":"SP.POP.TOTL",
    "value":"Population, total"
    },
"country":
    {"id":"AF",
    "value":"Afghanistan"
    },
"value":"20531160",
"decimal":"0",
"date":"2001"
}
...
```


- Нас буде цікавити поле "country"/"id" для ідентифікації країни а також поле "value", що містить дані про населення за вказаний рік.
- Щоб отримати сторінку 3 даних про населення за 2001 рік, треба сформувати запит:
- http://api.worldbank.org/countries/all/indicators/SP.POP.TOTL?date=2001&page=3&format=json

Аналіз змін населення по країнах за період (JSON). Реалізація

- Оскільки нам треба аналізувати дані про країни та населення та ці дані складаються з багатьох сторінок, опишемо функцію multipage_reader, яка буде читати документ JSON по сторінках та викликати задану у параметрах функцію обробки кожної сторінки.
- multipage_reader окремо читає першу сторінку та оцінює загальну кількість сторінок, аналізуючи заголовок.
- Для обробки документу JSON опишемо клас PopulationJSON.
- Цей клас містить поля:
  - self.countries - словник країн. Ключ - двохсимвольний код країни. Дані - список кортежів з назви країни та населення у початковому та кінцевому роках
  - self.pop_change - список кортежів (<зміна населення>, <код країни>, <назва>). Зміна населення – відношення населення у кінцевий рік до населення у початковому році.
  - self_start_year - початковий рік
  - self.fin_year - кінцевий рік
  - self.list_index - індекс у списку для населення країни (початковий рік - 1, кінцевий рік - 2)

Аналіз змін населення по країнах за період (JSON). Реалізація.2

- Методи у класі PopulationJSON – це конструктор, evaluate_changes – оцінити зміни населення, process_countries_page – обробити сторінку інформації про країни, process_population_page – обробити сторінку інформації про населення у деякому році.
- Головна частина модуля задає параметри, створює об’єкт класу PopulationJSON отримує та показує результати.

Приклад: аналіз змін населення по країнах за період (XML)

- Дані про країни та про населення у форматі XML також поділяються на сторінки.
- Кожна сторінка містить кореневий елемент-заголовок та послідовність елементів-даних про країни.
- Фрагмент сторінки даних нижче:

```
<wb:countries xmlns:wb="http://www.worldbank.org" page="1"
pages="7" per_page="50" total="304">
    ...
    <wb:country id="AFG">
        <wb:iso2Code>AF</wb:iso2Code>
        <wb:name>Afghanistan</wb:name>
        <wb:region id="SAS">South Asia</wb:region>
        <wb:adminregion id="SAS">South Asia</wb:adminregion>
        <wb:incomeLevel id="LIC">Low income</wb:incomeLevel>
        <wb:lendingType id="IDX">IDA</wb:lendingType>
        <wb:capitalCity>Kabul</wb:capitalCity>
        <wb:longitude>69.1761</wb:longitude>
        <wb:latitude>34.5228</wb:latitude>
    </wb:country>
```

Приклад: аналіз змін населення по країнах за період (XML).2

```
    <wb:country id="AFR">
        <wb:iso2Code>A9</wb:iso2Code>
        <wb:name>Africa</wb:name>
        <wb:region id="NA">Aggregates</wb:region>
        <wb:adminregion id=""/>
        <wb:incomeLevel id="NA">Aggregates</wb:incomeLevel>
        <wb:lendingType id="">Aggregates</wb:lendingType>
        <wb:capitalCity/>
        <wb:longitude/>
        <wb:latitude/>
    </wb:country>
    ...
</wb:countries>
```

- У заголовку нас буде цікавити атрибут "pages", який містить кількість сторінок документу.
- Для кожної країни ми будемо вибирати її двохсимвольний код "iso2Code" та назву "name".
- Також будемо враховувати, що для регіонів текст у полі "region"/"value" – це рядок "Aggregates".

Приклад: аналіз змін населення по країнах за період (XML).2

- Щоб отримати сторінку 3 даних про країни, треба сформувати запит:
- http://api.worldbank.org/countries?page=3
- Дані про населення містять аналогічний заголовок та послідовність даних про населення країн.
- Фрагмент даних для однієї країни – нижче:

```
...
<wb:data>
    <wb:indicator id="SP.POP.TOTL">Population,
total</wb:indicator>
    <wb:country id="AF">Afghanistan</wb:country>
    <wb:date>2001</wb:date>
    <wb:value>20531160</wb:value>
    <wb:decimal>0</wb:decimal>
</wb:data>
...
```

- Щоб отримати сторінку 3 даних про населення за 2001 рік, треба сформувати запит:
- http://api.worldbank.org/countries/all/indicators/SP.POP.TOTL?date=2001&page=3

Аналіз змін населення по країнах за період (XML). Реалізація

- Реалізація у XML фактично повторює реалізацію у JSON з урахуванням специфіки обробки XML.
- Ми знову використаємо функцію multipage_reader, яка буде читати документ XML по сторінках та викликати задану у параметрах функцію обробки кожної сторінки.
- Але при читанні однієї сторінки потрібно врахувати те, що дані XML можуть бути стиснуті (заархівовані) у форматі gzip.
- Щоб перевірити, чи заархівовано дані, проаналізуємо заголовки HTTP-відповіді.
- Наявність заголовку "Content-Encoding" вказує на те, що дані заархівовано.
- Щоб розархівувати дані, використаємо функцію decompress з модуля zlib.

Аналіз змін населення по країнах за період (XML). Реалізація.2

- Для обробки документу XML опишемо клас PopulationXML, який практично повторює клас PopulationJSON.
- Відрізняється тільки реалізація методів process_countries_page – обробити сторінку інформації про країни, process_population_page – обробити сторінку інформації про населення у деякому році.
- У цих методах для пошуку відповідних тегів до імені тегу треба додати адресу області дії імен.
- Замість "wb" – вставити "{http://www.worldbank.org}".
- Головна частина модуля задає параметри, створює об’єкт класу PopulationXML отримує та показує результати.

Резюме

- Ми розглянули:
  1. JSON. Засоби обробки даних JSON у Python
  2. XML. Області імен XML.
  3. Стандарти, пов’язані з XML. DOM та SAX
  4. Засоби обробки XML у Python
  5. Модуль ElementTree
  6. Клас Element
  7. Клас ElementTree
  8. Веб-сервіси

Де прочитати

1. Peter Norton, Alex Samuel, David Aitel та інші - Beginning Python
2. Mark Lutz - Programming Python. 4th Edition - 2011
3. Mark Pilgrim - Dive into Python, Version 5.4 - 2004
4. John Goerzen -Foundations of Python Network Programming. – 2004
5. Christopher A. Jones, Fred L. Drake, Jr. XML Processing with Python. - 2002
6. Марк Саммерфилд - Python на практике. ДМК - 2014
7. David Beazley - Python Cookbook, 3rd edition – 2013
8. http://www.w3schools.com/xml/
9. http://www.json.org/
10. http://citforum.ru/internet/xml/xml_rpc/