Chapter 16
XML

목차

- 파이썬과 XML
- XML 문서 파싱하기
- 노드
- 엘리먼트
- 엘리먼트 쉽게 다루기 (etree를 사용한 검색)
- XML 문서를 HTML로 변환하기
- 도서 관리 프로그램

파이썬과 XML

- XML (**eXensible Markup Language**)
  - W3C에서 1998년에 표준으로 채택한 다목적 마크업 언어
  - 수 많은 데이터 표현 기술로 사용 : 인터넷, RSS, Open API, …
  - 파이썬 내장 모듈 (DOM, ElementTree, …)을 사용하여 도서관리 프로그램 생성
- 파이썬3에서 지원하는 XML 관련 패키지
  - DOM (**Document Object Model**) API, SAX (**Simple API for XML**), 등

패키지 이름	이름공간
Fast XML parsing using Expat	xml.parsers.expat
DOM API	xml.dom
SAX	xml.sax
The ElementTree XML API	xml.etree.ElementTree

DOM의 이해

- 중첩된 형태의 Element들로 구성됨
  - Element를 시작하는 OpenTag
  - Element를 끝내는 CloseTag
  - Element의 속성을 설명하는 attribute
  - text

- 예

```
<tr class="up">
        <th scope="row">
                <a href="……" >천연가스(02.08)</a>
        </th>
        <td>4.25</td>
</tr>
```

파이썬과 XML

xml.parsers.expat 모듈

- **Fast XML parsing using Expat**
  - 빠른 XML 문서 파싱
  - xml.parsers.expat.ParserCreate([인코딩[, 이름공간구별자]])
    : xmlparser 객체를 생성하고 파싱을 수행
  - 빠른 파싱을 위해 문서의 유효성 검사를 하지 않음

xmlparser 메서드	**설명**
xmlparser.Parse(data[, isfinal])	data를 파싱. 더 이상 파싱하지 않으려면 isfinal을 True로 설정해서 호출
xmlparser.ParseFile(file)	파일의 데이터를 파싱
xml.parsers.expat.ErrorString(errno)	errno에 해당하는 에러문을 반환

  - 이벤트 발생 시에 이벤트 핸들러에 함수 등록

xmlparser 핸들러 이름	**설명**
xmlparser.StartElementHandler(name, attributes)	모든 엘리먼트의 시작 부분에서 호출됨. attributes: 사전 형식의 속성값들.
xmlparser.CharacterDataHandler(data)	문자데이터를 발견하면 호출됨.

xml.parsers.expat 모듈

xml_expat.py

- **Expat (xmlparser객체) 이용한 XML 파싱**

```
import xml.parsers.expat

xmlsrc = '''<?xml version="1.0"?><book ISBN="1111">
<title>Loving Python</title></book>'''

def start_element(name, attrs):
    print('Start element:', name, attrs)
def char_data(data):
    print('Character data:', repr(data))

pa = xml.parsers.expat.ParserCreate()          #xmlparser 객체 생성
pa.StartElementHandler = start_element         #이벤트 핸들러 연결
pa.CharacterDataHandler = char_data            #이벤트 핸들러 연결
pa.Parse(xmlsrc)
```

[실행결과]

```
Start element: book {'ISBN': '1111'}
Character data: '\n'
Start element: title {}
Character data: 'Loving Python'
```

xml.dom.minidom 모듈

- **DOM API**
  - 객체 기반 문서 모델(DOM: Document Object Model): XML 각 성분을 객체로 표현하고 모든 객체를 트리 형태로 메모리에 저장하고 관리
  - 최근 XML 관리를 위해 가장 많이 쓰는 방법
  - 연관된 데이터를 연속적으로 참고 가능 (SAX는 한번에 한 데이터만 참조 가능)
  - 기본 모듈 2개를 지원

  : minidom (대부분 사용), pulldom(문서가 클 경우 사용)

xml.dom.minidom 모듈

- **minidom 모듈 안의 클래스 객체들.**

DOM 안의 객체 이름	설명
DOMImplementation	DOM을 만드는 기본적인 인터페이스 포함.
Node	도큐먼트상에 존재하는 대부분 객체의 부모 객체
NodeList	노드 리스트 객체
DocumentType	도큐먼트를 처리하는 데 필요한 선언(시스템ID, 엔티티, 노테이션)을 위한 객체
Document	도큐먼트 전체를 나타내는 객체
Element	노드 엘리먼트 인터페이스 객체
Attr	엘리먼트 안의 속성값 노드 객체
Comment	소스 XML 문서에서 주석 부분을 처리하기 위한 객체
Text	도큐먼트에서 문자 정보를 포함하고 있는 객체
ProcessingInstruction	도큐먼트 처리를 위한 도구를 나타내는 객체

xml.sax 패키지

- **SAX(Simple API for XML)**
  - XML 문서를 순차적으로 읽어들이면서 구성요소(엘리먼트, 속성, 문자열)를 발견할 때마다 이벤트를 발생시켜 XML 문서를 처리하는 방법 (이벤트 기반 문서 처리)
  - DOM과 달리 메모리에 전부 로딩하고 파싱하는것이 아니라서 메모리 사용량이 적고 read only.
  - XML 문서는 forward 로만 진행 (DOM과 같은 객체기반 방식보다 빠르지만 한번 처리한 문서를 다시 사용할 수 없음.)
  - 매우 큰 XML 문서 처리에 많이 사용됨.

xml.sax 메서드	**설명**
xml.sax.make_parser([parser_list])	SAX XMLReader 를 생성하고 반환
xml.sax.parse(filename_or_stream, handler[, error_handler])	파일이나 스트림으로 부터 입력받은 XML 문서 파싱
xml.sax.parseString(string,handler [, error_handler])	문자열로 입력받은 XML 문서 파싱

xml.etree 패키지

xml_etree.py

- **The ElementTree XML API**(https://docs.python.org/ko/3/library/xml.etree.elementtree.html) :
- XML 문서의 엘리먼트들을 리스트나 사전으로 다룰 수 있는 인터페이스 제공

```
from xml.etree import ElementTree
xmlsrc='''<?xml version="1.0" ?>
<booklist>
   <book ISBN="0399250395">
      <title>The Very Hungry Caterpillar Pop-Up Book</title>
   </book>
   <book ISBN="0964729237">
      <title lang="english">The Shack</title>
   </book>
</booklist>
'''
try:
   tree = ElementTree.fromstring(xmlsrc)
except Exception:
   print ("Element Tree parsing Error : maybe the xml document is not corrected.")
   exit()
bookElements = tree.iter("book")
for item in bookElements:
   strTitle = item.find("title")
   print(strTitle.text)
```

[실행결과]

The Very Hungry Caterpillar Pop-Up Book
The Shack

xml.dom.minidom 모듈로 XML 문서 파싱하기

- DOM을 이용한 간단한 방법 : **minidom모듈 이용**

메소드 이름	설명
xml.dom.minidom.parse(file, parser)	file로부터 XML문서를 읽어서 Document 객체를 반환합니다. parser 인수가 주어지면 minidom의 기본 파서가 아닌 사용자가 원하는 파서를 사용할 수 있습니다(파서는 SAX2 parser 객체만 사용 가능합니다).
xml.dom.minidom.parseString(data, parser)	parse()와 비슷하지만 입력으로 파일 대신 문자열을 받습니다.

메서드 이름	설명
Document.createElement(tagName)	새로운 엘리먼트 객체 생성
Document.createTextNode(data)	새로운 문자노드 생성
Document.createAttribute(name)	속성(attribute) 생성
Document.getElementsByTagName(tagName)	tagName과 같은 모든 엘리먼트 가져옴

xml.dom.minidom 모듈

xml_minidom.py

- **minidom** 이용한 XML 파싱

```
from xml.dom.minidom import *
xmlsrc = """ <item>
<name>test</name>
</item>
"""
doc = parseString(xmlsrc)  #문자열 입력 파싱 함수, DOC객체 반환
print('doc=',doc)              #XML문서의 모든 엘리먼트, 속성, 주석 등 포함
print('doc.toxml()=', doc.toxml())
names = doc.getElementsByTagName("name") #"name"이라는 엘리먼트 가져오기
print('names=', names)
print('names.length=', names.length)
```

[실행결과]

```
doc= <xml.dom.minidom.Document object at 0x000001A84FE0C100>
doc.toxml()= <?xml version="1.0" ?><item>
<name>test</name>
</item>
names= [<DOM Element: name at 0x1a84fe145e0>]
names.length= 1
```

도서 관리 프로그램

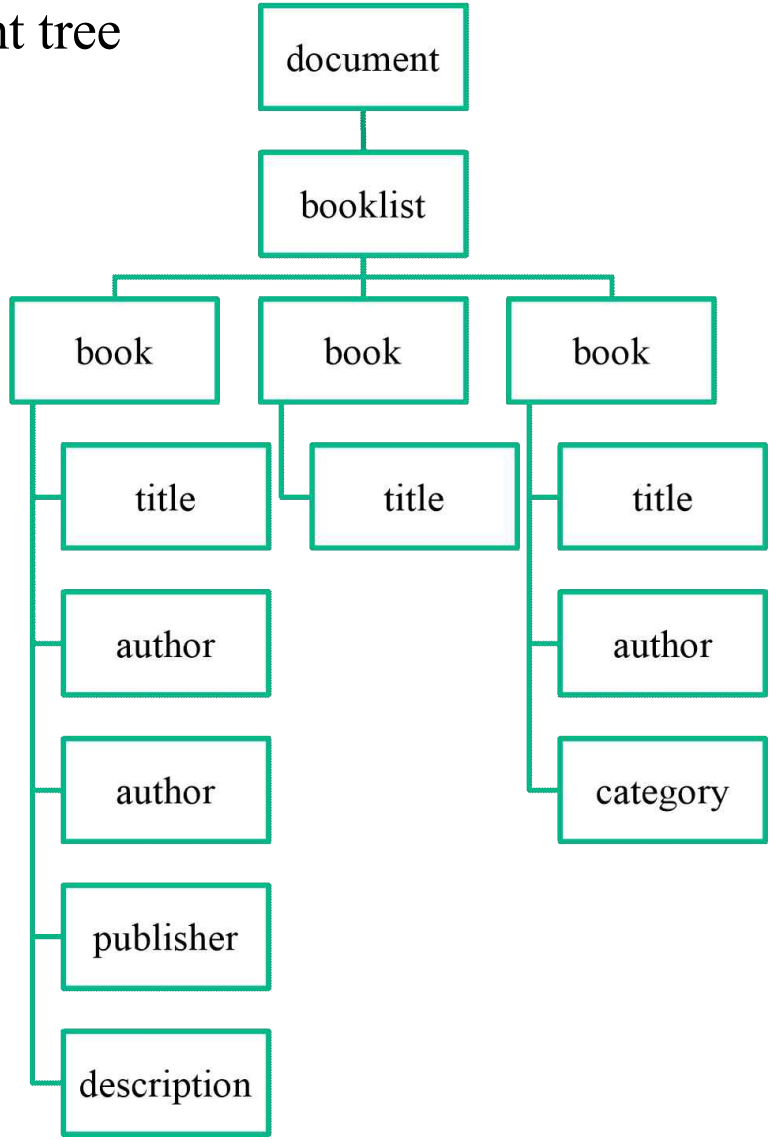
도서관리 프로그램

- 도서 정보 XML 문서 파싱, 도서 목록 출력, 새로운 도서 추가, 도서 제목 검색
- 예제 book.xml (도서 정보 XML 문서)

```
<?xml version="1.0" ?>
<booklist cnt="3">
  <book ISBN="0399250395">
    <title>The Very Hungry Caterpillar Pop-Up Book</title>
    <author name="Eric Carle"/>
    <author name="Keith Finch"/>
    <publisher> Philomel Books</publisher>
    <description> Celebrating the 40th anniverary of one of the most popular children's
books ever created</description>
  </book>
  <book ISBN="0964729237">
    <title lang="english">The Shack</title>
  </book>
  <book ISBN="0553281097">
    <title>You Can Negotiate Anything</title>
    <author name="Herb Cohen"/>
    <category cid="12">Negotiate and narrative skill</category>
  </book>
</booklist>
```

도서관리 프로그램

- book.xml 의 element tree



도서관리 프로그램 (수행결과 1/3)

```
Welcome! Book Manager Program (xml versic
========Menu==========
Load xml:  l
Print dom to xml: p
Quit program:   q
print Book list: b
Add new book: a
sEarch Book Title: e
Make html: m
====================
select menu :l
please input file name to load :book.xml
XML Document loading complete
```

```
Welcome! Book Manager Program (xml version
========Menu==========
Load xml:  l
Print dom to xml: p
Quit program:   q
print Book list: b
Add new book: a
sEarch Book Title: e
Make html: m
====================
select menu :p
<?xml version="1.0" ?><booklist cnt="3">
   <book ISBN="0399250395">
      <title>The Very Hungry Caterpillar Pop-Up
    Book</title>
      <author name="Eric Carle"/>
      <author name="Keith Finch"/>
      ..........
```

차후의 모든 기능들이 읽어 들인 내용을 체크하므로 항상 load 먼저 하기!

도서관리 프로그램 (수행결과 2/3)

```
Welcome! Book Manager Program (xml version)
========Menu==========
Load xml:  l
Print dom to xml: p
Quit program:   q
print Book list: b
Add new book: a
sEarch Book Title: e
Make html: m
====================
select menu :b
title= The Very Hungry Caterpillar Pop-Up Book
title= The Shack
title= You Can Negotiate Anything
```

```
Welcome! Book Manager Program (xml vers
========Menu==========
Load xml:  l
Print dom to xml: p
Quit program:   q
print Book list: b
Add new book: a
sEarch Book Title: e
Make html: m
====================
select menu :e
input keyword to search :Shack
('0964729237', 'The Shack')
```

도서관리 프로그램 (수행결과 3/3)

```
Welcome! Book Manager Program (xml version)
========Menu==========
Load xml:  l
Print dom to xml: p
Quit program:   q
print Book list: b
Add new book: a
sEarch Book Title: e
Make html: m
====================
select menu :m
input keyword code to the html  :Shack
----------------------
<?xml
    version="1.0" ?><html><header/><body><b>I
    SBN:0964729237</b><p>Title:The
    Shack</p><br/></body></html>
----------------------
```

```
Welcome! Book Manager Program (xml vers
========Menu==========
Load xml:  l
Print dom to xml: p
Quit program:   q
print Book list: b
Add new book: a
sEarch Book Title: e
Make html: m
====================
select menu :q
Thank you! Good Bye
```

도서관리 프로그램 구현 계획

- 예제 book.py (도서관리 프로그램) : 필요한 함수들 정리 (흐린 색: utility함수)
  - printMenu() # 메뉴 출력
  - launcherFunction(menu) # 입력한 메뉴에 따른 함수를 호출해 줌
  - LoadXMLFromFile() # (L)oad xml
  - BooksFree() # 종료 직전에 dom을 해제
  - QuitBookMgr() # (Q)uit program
  - PrintDOMtoXML() # (P)rint dom to xml
  - PrintBookTitle(tags) # print (B)ook list
  - AddBook(bookdata) # (A)dd new book
  - SearchBookTitle(keyword) # s(E)arch book title
  - MakeHtmlDoc(BookList) # (M)ake Html
  - printBookList(blist) # SearchBookTitle()의 결과 리스트를 출력
  - checkDocument() # BooksDoc != None인지 체크.

```
##### run #####
while(loopFlag > 0):
    printMenu()
    menuKey = str(input ('select menu : '))
    launcherFunction(menuKey)
else:
    print ("Thank you! Good Bye")
```

도서관리 프로그램

book.py

- XML문서
  - **booklist** 루트 엘리먼트
  - 그 아래에 책 정보를 가진 **book** 엘리먼트 들 (속성 : Title, ISBN, author, description,…)

```
# -*- coding: utf-8 -*-
from xml.dom.minidom import parse  #파일파싱에 사용할 함수
from xml.etree import ElementTree #element 관리에 사용할 클래스

##### global
loopFlag = 1                    #무한 루프 제어변수
BooksDoc = None                 #XML문서 파싱한 후 반환된 DOM 객체 변수

…
```

도서관리 프로그램 (printMenu 함수)

book.py

```
def printMenu():
    print("\nWelcome! Book Manager Program (xml version)")
    print("========Menu==========")
    print("Load xml:  l")
    print("Print dom to xml: p")
    print("Quit program:   q")
    print("print Book list: b")
    print("Add new book: a")
    print("sEarch Book Title: e")
    print("Make html: m")
    print("====================")
```

[실행결과]

```
Welcome! Book Manager Program (xml version)
========Menu==========
Load xml:  l
Print dom to xml: p
Quit program:   q
print Book list: b
Add new book: a
sEarch Book Title: e
Make html: m
====================
select menu :
```

도서관리 프로그램

book.py

```
def  QuitBookMgr(): # Quit Program
    global loopFlag
    loopFlag = 0
    BooksFree()

def  printBookList(blist): # 리스트 프린트 utility
    for res in blist:
        print (res)

def  checkDocument(): # 읽어들인 문서가 있는지 체크하는 utility
    global BooksDoc
    if BooksDoc == None:
        print("Error : Document is empty")
        return False
    return True
```

[실행결과]

```
Welcome! Book Manager Program (xml versi
=========Menu==========
Load xml:  l
Print dom to xml: p
Quit program:   q
print Book list: b
Add new book: a
sEarch Book Title: e
```

도서관리 프로그램

book.py

- **LoadXMLFromFile()** : minidom에서드 XML 문서 파싱하고 전역변 BooksDoc에 저장

```
def LoadXMLFromFile():
    fileName = str(input ("please input file name to load :")) # 읽을 파일 경로 입력

    try:
        with open(fileName, encoding='utf-8') as xmlFD: # XML 문서 open
            try:
                dom = parse(xmlFD) # XML 문서 파싱
            except Exception:
                print ("loading fail!!!")
            else:
                print ("XML Document loading complete")
                return dom # dom 반환
    except IOError:
        print ("invalid file name or path")
        return None
    return None
```

도서관리 프로그램

book.py

- **BooksFree()** : 사용후에는 unlink()메서드 호출해 DOM 객체 내부의 참조를 제거

```
def BooksFree():
    if checkDocument():
        BooksDoc.unlink()   # minidom 객체 해제
```

도서관리 프로그램

book.py

- PrintDOMtoXML() : DOM 객체를 XML 문서로 (보기 편하게) 변환

```
def PrintDOMtoXML():
    if checkDocument():
        print(BooksDoc.toprettyxml(newl=‘’))
```

```
Welcome! Book Manager Program (xml version)
========Menu==========
Load xml:  l
Print dom to xml: p
Quit program:   q
print Book list: b
Add new book: a
sEarch Book Title: e
Make html: m
==================
select menu :p
<?xml version="1.0" ?><booklist cnt="3">
        <book ISBN="0399250395">
                <title>The Very Hungry Caterpillar Pop-Up Book</title>
                <author name="Eric Carle"/>
…
```

도서관리 프로그램

- DOM의 모든 컴포넌트들은 Node의 서브클래스
  - Node는 엘리먼트(Element), 속성(Attribute), 텍스트(Text), …

속성 이름	설명
Node.nodeType	노드타입을 나타내는 정수값입니다 (노드타입에 대한 자세한 설명은 다음 뱀잡기를 참고해주세요).
Node.parentNode	현재 노드의 부모 노드, 만약 parentNode 값이 None이면 현재 엘리먼트는 루트 엘리먼트입니다.
Node.hasAttributes()	노드가 있으면 True를 리턴합니다.
Node.hasChildNodes()	자식 노드가 있으면 True를 리턴합니다.
Node. appendChild(newChild)	새로운 노드를 현재 노드의 자식 노드로 추가 합니다.
Node. insertBefore(newChild, refChild)	새로운 노드(newChild)를 지정된 노드(refChild) 앞쪽에 삽입 합니다.
Node. removeChild(oldChild)	지정된 자식노드(oldChild)를 삭제 합니다.

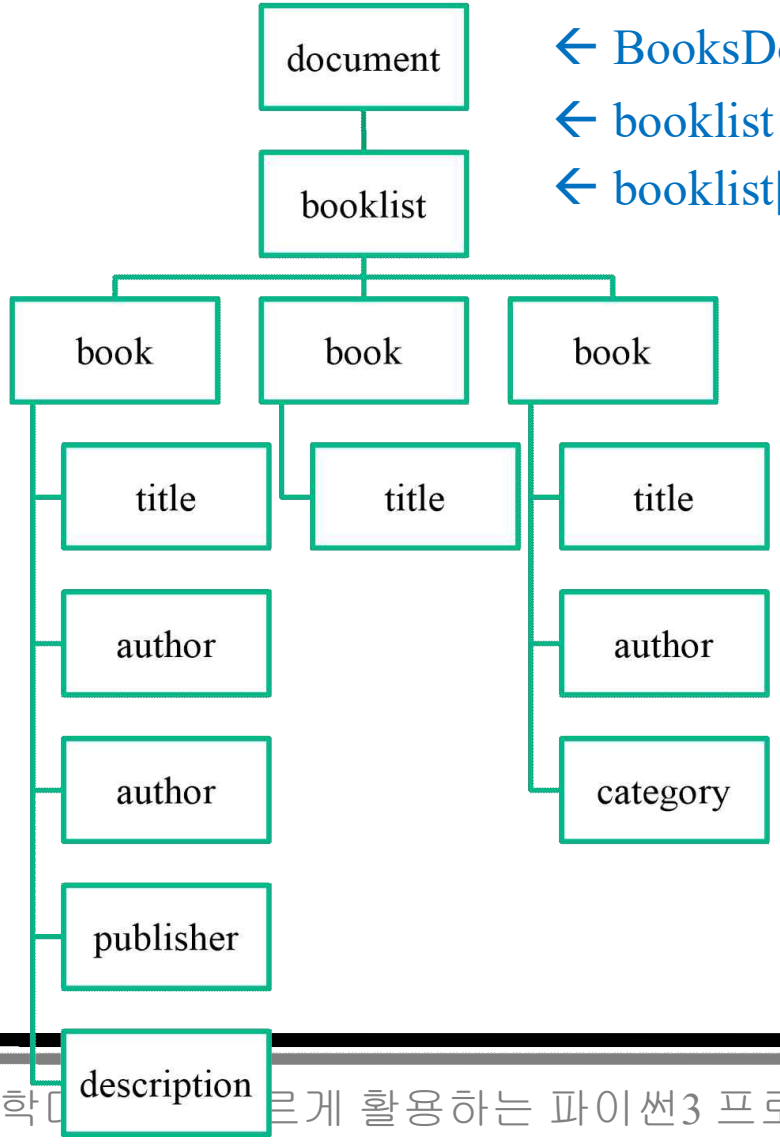
도서관리 프로그램

- 노드 타입
  - ELEMENT_NODE = 1, TEXT_NODE = 3, DOCUMENT_TYPE_NODE = 9, …

```
>>> from xml.dom.minidom import parse
>>> dom = parse("book.xml")          #DOC 객체
>>> dom.nodeType                     #nodeType -> 9 (DOCUMENT_TYPE_NODE)
9
>>> booklists = dom.childNodes       #books의 자식노드 = 루트 엘리먼트
>>> type(booklists)                  #childNodes의 타입은 NodeList
<class ' xml.dom.minicompat.NodeList ' >
>>> len(booklists)
1
>>> booklists[0]                     #NodeList 이므로 배열을 사용해 각 노드에 접근
<DOM Element: booklist at 0x202137e1178>
>>> booklists[0].nodeType            #nodeType -> 1 (ELEMENT_NODE)
1
>>> book = booklists[0].childNodes   #bookslist의 자식노드
```

도서관리 프로그램

- book.xml 의 element tree 탐색 함수들
  - xml.dom 문서 객체 모델 API (https://docs.python.org/ko/3/library/xml.dom.html)



← BooksDoc = parse(open(fileName, encoding='utf-8'))

← booklist = BooksDoc.childNodes

← booklist[0]

←books = booklist[0].childNodes

←books[0].nodeName == "book"

←subitems = book.childNodes

←subitems[0].nodeName == “title"

도서관리 프로그램

```
def  launcherFunction(menu):
    global BooksDoc
    if menu ==  'l':
        BooksDoc = LoadXMLFromFile()
    elif menu == 'q':
        QuitBookMgr()
    elif menu == 'p':
        PrintDOMtoXML()
    elif menu == 'b':
        PrintBookTitle(["title",])
    elif menu == 'a':
        ISBN = str(input ('insert ISBN :'))
        title = str(input ('insert Title :'))
        AddBook({'ISBN':ISBN, 'title':title})
    elif menu == 'e':
        keyword = str(input ('input keyword to search :'))
        printBookList(SearchBookTitle(keyword))
    elif menu == 'm':
        keyword = str(input ('input keyword code to the html  :'))
        html = MakeHtmlDoc(SearchBookTitle(keyword))
        print("----------------------")
        print(html)
        print("----------------------")
    else:
        print ("error : unknow menu key")
```

도서관리 프로그램

book.py

- **PrintBookTitle(tags)** : 책 목록 출력 (tags 리스트를 인자로 받아 매칭되는 엘리먼트 출력)

```
========Menu==========
print Book list: b
====================
select menu : b
title= The Very Hungry Caterpillar Pop-Up Book
title= The Shack
title= You Can Negotiate Anything
```

```
def  PrintBookTitle(tags):
    global BooksDoc
    # DOM이 None인지 검사
    if not checkDocument():
        return None

    booklists = BooksDoc.childNodes
    books = booklists[0].childNodes
    for book in books:
        if book.nodeName == "book":        # 엘리먼트를 중 book인 것을 추출
            subitems = book.childNodes     # book에 들어 있는 노드들을 가져옴
            for item in subitems:
                if item.nodeName in tags:
                    print( " title= " ,item.firstChild.nodeValue)  # 책 목록을 출력
```

도서관리 프로그램

- **Element** : XML 문서는 element로 이루어져 있음
  - 새로운 도서 등록 : 하나의 도서 데이터는 book element와 하위 element로 구성됨

메소드 이름	설명
Element. getElementsByTagName(tagName)	엘리먼트 이름 중 tagName과 매칭되는 엘리먼트들을 리턴합니다.
Element.getElementsByTagNameNS (namespaceURI, localName)	XML문서에 이름공간이 지정되어 있는 경우, namespaceURI안에 localName과 매칭되는 엘리먼트들을 리턴합니다.
Element.hasAttribute(name)	엘리먼트 속성 중 name에 해당하는 속성이 있으면 참을 리턴합니다.
Element. hasAttributeNS(namespaceURI, localName)	XML문서에 이름공간이 지정되어 있는 경우, namespaceURI안에 localName과 매칭되는 엘리먼트가 있으면 참을 리턴합니다.
Element.getAttribute(name)	name에 해당하는 속성 값을 출력합니다.

도서관리 프로그램

book.py

- **AddBook(bookdata)**: 인자로 사전**{'ISBN':ISBN, 'title':title}**을 입력받음
  - 새로운 도서 등록 : 하나의 도서 데이터는 book 엘리먼트와 하위 엘리먼트로 구성됨

```
elif menu == 'a':
        ISBN = str(input ('insert ISBN :'))
        title = str(input ('insert Title :'))
        AddBook({'ISBN':ISBN, 'title':title})
```

```
def  AddBook(bookdata):
    global BooksDoc
    if not checkDocument() :
        return None

    newBook = BooksDoc.createElement('book')    # Book 엘리먼트 생성
    newBook.setAttribute('ISBN',bookdata['ISBN']) #ISBN 속성 설정
    titleEle = BooksDoc.createElement('title')        # Title 엘리먼트 생성
    titleNode = BooksDoc.createTextNode(bookdata['title']) #TextNode 생성
    ……
```

도서관리 프로그램

book.py

- **AddBook(bookdata)**: 인자로 사전**{'ISBN':ISBN, 'title':title}**을 입력받음
  - 새로운 도서 등록 : 하나의 도서 데이터는 book 엘리먼트와 하위 엘리먼트로 구성됨

```
def  AddBook(bookdata):
    ……
    try:
        titleEle.appendChild(titleNode) # 텍스
        newBook.appendChild(titleEle)
        booklist = BooksDoc.firstChild
    except Exception:
        print ("append child fail - please,check the parent element & node!!!")
        return None
    else:
        if booklist != None:
            booklist.appendChild(newBook)
            cnt = int(booklist.getAttribute('cnt')) + 1
            booklist.setAttribute('cnt',str(cnt))
```

```
elif menu == 'a':
        ISBN = str(input ('insert ISBN :'))
        title = str(input ('insert Title :'))
        AddBook({'ISBN':ISBN, 'title':title})
```

```
<?xml version="1.0" ?>
<booklist cnt="3"> → “4”로 바꾸기
……
 <book ISBN=“1234567">
      <title>Loving Python</title>
</book>
</booklist>
```

도서관리 프로그램

- **Add new book 기능 테스트.**

```
Welcome! Book Manager Program (xml version)
========Menu==========
Load xml:  l
Print dom to xml: p
Quit program:   q
print Book list: b
Add new book: a
sEarch Book Title: e
Make html: m
==================
select menu :a
insert ISBN :1234567
insert Title :Loving Python
. . .
==================
select menu :b
title= The Very Hungry Caterpillar Pop-Up Book
title= The Shack
title= You Can Negotiate Anything
title= Loving Python
```

'b' 와 'p' 로써 확인하기!
- 새 책 정보 보이는지
- <booklist>의 cnt 속성값이 바뀌었는지.

도서관리 프로그램

- 엘리먼트 쉽게 다루기 : **xml.etree.ElementTree** 클래스
  - 엘리먼트 생성, 값 변경 검색, 엘리먼트를 **XML**로 변환

ElementTree 주요 모듈 함수	**설명**
xml.etree.ElementTree.parse(file[,parser])	파일로부터 XML문서 파싱
xml.etree.ElementTree.fromstring(text)	문자열 text를 파싱 ElementTree 객체 반환
xml.etree.ElementTree.Element(tag[,attrib][,**extra])	tag 이름을 가진 엘리먼트를 생성
xml.etree.ElementTree.SubElement(parent,tag[,attrib[,**extra]]	Element()와 비슷하지만 parent의 자식 엘리먼트로 만들어짐
xml.etree.ElementTree.tostring(element[,encoding])	element객체를 XML문자열로 변환

도서관리 프로그램

- 엘리먼트 쉽게 다루기 : **xml.etree.ElementTree** 클래스
  - 엘리먼트 생성, 값 변경 검색, 엘리먼트를 XML로 변환

ElementTree 객체의 주요 메서드	설명
find(path)	path에 매칭되는 엘리먼트 반환
iter([tag])	현재 엘리먼트의 하위 엘리먼트를 모두 가져 옴. tag가 지정되어 있으면 tag에 해당하는 엘리먼트만 반환
getroot()	현재 XML 문서 중 가장 상위 엘리먼트 객체 반환
write(file[,encoding])	현재 ElementTree객체를 file에 저장

도서관리 프로그램

book.py

- 도서 검색 : **SearchBookTitle(keyword)** 함수 (ElementTree 이용)
  - 인자로 받은 **keyword**로 **Title** 검색한 후 **(ISBN, Title)** 리스트 출력
  - ElementTree.fromstring, Element의 iter(), find()함수와 text속성 이용.

```
def SearchBookTitle(keyword):
    global BooksDoc
    retlist = []
    if not checkDocument():
        return None
    try:
        tree = ElementTree.fromstring(BooksDoc.toxml()) #xml.etree.ElementTree.Element 객체
    except Exception:
        print ("Element Tree parsing Error : maybe the xml document is not corrected.")
        return None

    #Book 엘리먼트 리스트 가져 오기.
    bookElements = tree.iter("book")      #_element_iterator 객체
    for book in bookElements:             #xml.etree.ElementTree.Element 객체
        title = book.find("title")        #'title' xml.etree.ElementTree.Element 객체
        if (title.text.find(keyword) >=0 ):  #keyword 검색
            retlist.append((book.attrib["ISBN"], title.text))  #리스트에 (ISBN, title) 튜플추가
    return retlist
```

break point 잡고 F10으로 진행하면서 빨간색 객체(tree, book, title)들의 tag, text 속성값 살펴보기.

도서관리 프로그램

- 도서 검색 : **SearchBookTitle(keyword)** 함수 수행
  - 인자로 입력받은 **keyword**로 **Title** 검색한 후 **(ISBN, Title)** 리스트 출력

```
Welcome! Book Manager Program (xml version)
========Menu==========
Load xml:  l
Print dom to xml: p
Quit program:   q
print Book list: b
Add new book: a
sEarch Book Title: e
Make html: m
==================
select menu :e
input keyword to search :The
('0399250395', 'The Very Hungry Caterpillar Pop-Up Book')
('0964729237', 'The Shack')
```

도서관리 프로그램 - XML 문서를 HTML 로 변

book.py

- **MakeHtmlDoc(BookList) 함수 :** DOM 객체를 생성하고 부모 엘리먼트와 자식 엘리먼트를 생성해 DOM 객체에 추가

다음 페이지의 결과를 보고 HTML 문서 구조를 이해한 후에 작성하자.

```
def  MakeHtmlDoc(BookList):
    from xml.dom.minidom import getDOMImplementation
    # DOM(xml.dom.minidom.DOMImplementation) 객체를 생성
    impl = getDOMImplementation()
    # xml.dom.minidom.Document 객체 생성
    newdoc = impl.createDocument(None, "html", None) #최상위 엘리먼트 = 'html'
    top_element = newdoc.documentElement #최상위 엘리먼트 가져오기.
    header = newdoc.createElement('header') # 'header' 엘리먼트 만들어서
    top_element.appendChild(header)  # 'header' 엘리먼트 추가

    # Body 엘리먼트 생성
    body = newdoc.createElement('body') ') # 'body' 엘리먼트 만들어서


    ……

    top_element.appendChild(body) # 'body' 엘리먼트 추가
    return newdoc.toprettyxml()
```

도서관리 프로그램 - XML 문서를 HTML 로 변

book.py

- **MakeHtmlDoc(BookList) 함수 :** DOM 객체를 생성하고 부모 엘리먼트와 자식 엘리먼트를 생성해 DOM 객체에 추가

```
<body>
   <b>ISBN:0399250395</b>
   <p>Title:The Very Hungry …</p>
   <br/>
</body>
```

```
def  MakeHtmlDoc(BookList):
   ……
   for bookitem in BookList:
      b = newdoc.createElement('b') # Bold 엘리먼트 생성
      ibsnText = newdoc.createTextNode("ISBN:" + bookitem[0]) # 텍스트 노드 생성
      b.appendChild(ibsnText)  # Bold 엘리먼트에 추가
      body.appendChild(b) # Bold 엘리먼트 를 body에 추가.

      p = newdoc.createElement('p') # title 엘리먼트 생성
      titleText= newdoc.createTextNode("Title:" + bookitem[1]) # 텍스트 노드 생성
      p.appendChild(titleText) # paragraph 엘리먼트에 추가
      body.appendChild(p) # paragraph 엘리먼트 를 body에 추가.

      br = newdoc.createElement('br')   # line break (br tag) 생성
      body.appendChild(br) # line break (br tag)를 body에 추가.
   ……
```

도서관리 프로그램 - XML 문서를 HTML 로 변환

- MakeHtmlDoc(BookList) 함수 수행

```
Welcome! Book Manager Program (xml version)
========Menu==========
Load xml:  l
Print dom to xml: p
Quit program:   q
print Book list: b
Add new book: a
sEarch Book Title: e
Make html: m
==================
select menu :m
input keyword code to the html  :The
----------------------
<?xml version="1.0" ?>
<html>
        <header/>
        <body>
                <b>ISBN:0399250395</b>
                <p>Title:The Very Hungry Caterpillar Pop-Up Book</p>
                <br/>
                <b>ISBN:0964729237</b>
                <p>Title:The Shack</p>
                <br/>
        </body>
</html>
```

Term Project 팀 준비

- 팀 구성
  - 2인 1팀 권장
  - 1인 1팀 가능

- 프로젝트 개요
  - Open API 이용한 GUI(Tkinter) 프로그램
  - GUI 게임들 (Tic-Tac-Toe, 사목게임, Hangman, Yahtzee) 동시 진행
  - Youtube 이용한 발표
  - Github 이용한 프로젝트 진행 (commit 시기와 횟수를 평가에 반영)