Lab. XML file 다루기

1. xml.ElementTree package
   1)parse()로 ElementTree 클래스를 만든후, tree를 전부 조회하기

```
      import xml.etree.ElementTree as ET

      tree = ET.parse('books.xml')

      print(type(tree))
      s = tree.getiterator()
      for i in s :
          print(i)

      print(type(tree))
      root = tree.getroot()
      print(root)
      print(type(root))
```

   2)Element 내부 속성과 method 확인하기

```
      import xml.etree.ElementTree as ET

      tree = ET.parse('books.xml')
      root = tree.getroot()
      print(root)
      print(root.tag)
      print(root.attrib)
      print(root.tail)
      print(root.text)

      books = root.getchildren()
      print(books[0].get("id"))  #첫번째 book의 id속성값 구하기
      print(books[1].get("id"))
      print(books[2].get("id"))
      print(books[0].keys())
      print(books[0].items())

      for i in books[0].getchildren():
         print(i)   #하위 element 읽어오기
```

2. XML 문서 순환 조회
   1)반복형이나 반복자로 점검하면 False가 나온다.
   2)하지만 Class내에 __getitem__()가 있기 때문에 반복가능하다.

```
      import xml.etree.ElementTree as ET
      import collections.abc as cols

      tree = ET.ElementTree(file = 'books.xml')
      root = tree.getroot()

      print(issubclass(type(root), cols.Iterable))    #False
      print(issubclass(type(root), cols.Iterator))    # False
      for i in root:
         print(i)

   3)XML문서를 반복하면서 내부의 속성인 tag와 attrib 출력하기
      import xml.etree.ElementTree as ET

      tree = ET.ElementTree(file = 'books.xml')
      root = tree.getroot()

      for child_tag in root:
          print(child_tag.tag, child_tag.attrib, child_tag.text)


3. xpath를 이용한 순환처리
   1)특정 tag 찾기
      import xml.etree.ElementTree as ET

      tree = ET.ElementTree(file = 'doc.xml')
      root = tree.getroot()

      print(root.find('branch').tag)
      print(root.findtext('branch'), end='')
      b = root.findall('branch')
      for child_tag in b:
          a = child_tag.text
          print(a, end = '')

   2)특정 tag의 text를 조회하기
      import xml.etree.ElementTree as ET

      tree = ET.ElementTree(file = 'doc.xml')
      root = tree.getroot()

      print(root.findtext('branch/sub-branch'))
      print(root.find('branch/sub-branch').text)

   3)반복자를 이용해서 tag 찾기
      import xml.etree.ElementTree as ET

      tree = ET.ElementTree(file = 'doc.xml')
      root = tree.getroot()

      for i in root.iterfind('branch'):
          print(i.tag, i.attrib, i.text)

   4)특정 하위 요소에서 찾기
      import xml.etree.ElementTree as ET

      tree = ET.ElementTree(file = 'doc.xml')
      root = tree.getroot()
      for i in root.iterfind('*//sub-branch'):
          print(i.tag, i.attrib, i.text)

   5)특성 속성을 갖고 있는 요소 찾기
      import xml.etree.ElementTree as ET

      tree = ET.ElementTree(file = 'doc.xml')
      root = tree.getroot()

      for i in root.iterfind("branch[@name='invalid']"):
          print(i.tag, i.attrib, i.text)

   6)문서에 있는 모든 text 조회해서 출력하기
      import xml.etree.ElementTree as ET

      tree = ET.ElementTree(file = 'doc.xml')
      root = tree.getroot()

      for i in root.itertext():
          print(i, end='')


4. XML 문서 생성하기
   1)Element 생성해서 요소 만들기(append() 이용)
      import xml.etree.ElementTree as ET

      root = ET.Element('root')
      print(root)
      print(root.tag)

      child = ET.Element('child')
      print(child)
      print(child.tag)

      root.append(child)

      ET.dump(root)

   2)Element 생성해서 요소 만들기(SubElement class 이용)
      import xml.etree.ElementTree as ET

      root = ET.Element('root')
      print(root)
      print(root.tag)

      ET.SubElement(root, 'child')

      ET.dump(root)

   3)Element의 insert() 사용해서 요소의 특정 위치 지정하기
```

```
  import xml.etree.ElementTree as ET

  root = ET.Element('root')
  print(root)
  print(root.tag)

  ET.SubElement(root, 'child1')
  child2 = ET.Element('child2')
  root.insert(2, child2)    #2번째 위치에 삽입
  ET.dump(root)
```

4)remove()를 이용하여 특정 요소 삭제하기

```
  import xml.etree.ElementTree as ET

  root = ET.Element('root')
  print(root)
  print(root.tag)

  ET.SubElement(root, 'child1')
  child2 = ET.Element('child2')
  root.insert(2, child2)

  root.remove(child2)
  ET.dump(root)
```

5)Element 생성하며 속성 추가하기

```
  import xml.etree.ElementTree as ET

  #<book author="Michael Jackson" />
  book = ET.Element('book', author = 'Michael Jackson')
  print(sorted(book.keys()))

  for name, value in sorted(book.items()):
      print('%s = %r' % (name, value))

  ET.dump(book)
```

6)Element 생성하며 속성 추가하기 - set() 사용하기

```
  import xml.etree.ElementTree as ET

  #<book author="Michael Jackson" />
  #<book price="25000" />
  book = ET.Element('book', author = 'Michael Jackson')
  book.set("price", '25000')
  print(sorted(book.keys()))

  for name, value in sorted(book.items()):
      print('%s = %r' % (name, value))

  ET.dump(book)
```

7)Element의 속성을 dict로 관리해서 검색 및 수정하기

```
import xml.etree.ElementTree as ET

book = ET.Element('book', author = 'Michael Jackson')
book.set("price", '25000')

print(sorted(book.keys()))

for name, value in sorted(book.items()):
    print('%s = %r' % (name, value))

ET.dump(book)

attributes = book.attrib
print(attributes['author'])
#print(attributes['title'])  #Error
attributes['title'] = 'Python Fundamental'
print(attributes['title'])
print(book.get('title'))

ET.dump(book)
```

5. XML을 문자열로 처리하기

1)XML 문서를 만들고 문자열로 보기

```
import xml.etree.ElementTree as ET

root = ET.Element('root')
print(root)
print(root.tag)
root.append(ET.Element('child1'))
child2 = ET.SubElement(root, 'child2')
child3 = ET.SubElement(root, 'child3')
print(ET.tostring(root))
```

2)fromstring()을 tostring()으로 읽기

```
import xml.etree.ElementTree as ET

xml_ = """<?xml version="1.0"?>
            <books><book id='test1'><title>Python Fundamental</title></book></books>"""
xml_str = ET.fromstring(xml_)
print(ET.tostring(xml_str))
```

3)BytesIO 이용하기

```
import xml.etree.ElementTree as ET
from io import BytesIO

file_like_object = BytesIO(b"<books><book id='test1'><title>Python Fundamental</title></book></books>")
tree = ET.parse(file_like_object)
```

```
        root = tree.getroot()
        print(ET.tostring(root))

    4)StringIO도 가능
        import xml.etree.ElementTree as ET
        from io import StringIO

        file_like_object = StringIO("<books><book id='test1'><title>Python 
        Fundamental</title></book></books>")
        tree = ET.parse(file_like_object)
        root = tree.getroot()
        print(ET.tostring(root))

        book = root.find('book')
        print(book.tag)

    5)iterparse()를 통한 parsing
        import xml.etree.ElementTree as ET
        from io import BytesIO

        file_like_object = BytesIO(b"<books><book id='test1'><title>Python 
        Fundamental</title></book></books>")

        for event, element in ET.iterparse(file_like_object):
            if element.tag == 'title':
                print(element.text)
            elif element.tag == 'book':
                print('It has subtrees.')
                element.clear()

    6)문자열로 XML을 만들고 특정 태그를 검색해서 조회해서 출력
        import xml.etree.ElementTree as ET

        input = """<books>
                        <book id='bk1'>
                            <title>Python Fundamental</title>
                            <author>Michael Jackson</author>
                            <price>25000</price>
                        </book>
                        <book id='bk2'>
                            <title>Machine Learning Fundamental</title>
                            <author>Sujan</author>
                            <price>35000</price>
                        </book>
                    </books>"""

        root = ET.fromstring(input)
        books = root.findall('book')
        print('Book count :', len(books))

        for book in books:
            print('Title =', book.find('title').text)
            print('Author =', book.find('author').text)
            print('Price =', book.find('price').text)
            print('-' * 20)
```

6. XML file로 내보내기

   1)write()로 파일 저장하기

```
    import xml.etree.ElementTree as ET

    input = """<books>
                        <book id='bk1'>
                           <title>Python Fundamental</title>
                           <author>Michael Jackson</author>
                           <price>25000</price>
                        </book>
                        <book id='bk2'>
                           <title>Machine Learning Fundamental</title>
                           <author>Sujan</author>
                           <price>35000</price>
                        </book>
                     </books>"""

    root = ET.fromstring(input)
    tree = ET.ElementTree(root)
    ver = ET.Element('docversion')
    ver.text = '1.2.1'
    root.append(ver)

    tree.write(open('mybook.xml', 'wb'))

    tree = ET.parse('mybook.xml')
    root_ = tree.getroot()
    print(ET.tostring(root_))
```