

```
1 Lab. XML file 다루기
2
3 1. xml.ElementTree package
4 1)parse()로 ElementTree 클래스를 만든후, tree를 전부 조회하기
5     import xml.etree.ElementTree as ET
6
7     tree = ET.parse('books.xml')
8
9     print(type(tree))
10    s = tree.getiterator()
11    for i in s :
12        print(i)
13
14    print(type(tree))
15    root = tree.getroot()
16    print(root)
17    print(type(root))
18
19 2)Element 내부 속성과 method 확인하기
20     import xml.etree.ElementTree as ET
21
22     tree = ET.parse('books.xml')
23     root = tree.getroot()
24     print(root)
25     print(root.tag)
26     print(root.attrib)
27     print(root.tail)
28     print(root.text)
29
30     books = root.getchildren()
31     print(books[0].get("id")) #첫번째 book의 id속성값 구하기
32     print(books[1].get("id"))
33     print(books[2].get("id"))
34     print(books[0].keys())
35     print(books[0].items())
36
37     for i in books[0].getchildren():
38         print(i) #하위 element 읽어오기
39
40
41 2. XML 문서 순환 조회
42 1)반복형이나 반복자로 점검하면 False가 나온다.
43 2)하지만 Class내에 __getitem__()가 있기 때문에 반복가능하다.
44     import xml.etree.ElementTree as ET
45     import collections.abc as cols
46
47     tree = ET.ElementTree(file = 'books.xml')
48     root = tree.getroot()
49
50     print(issubclass(type(root), cols.Iterable)) #False
51     print(issubclass(type(root), cols.Iterator)) # False
```

```
52
53     for i in root:
54         print(i)
55
56 3)XML문서를 반복하면서 내부의 속성인 tag와 attrib 출력하기
57     import xml.etree.ElementTree as ET
58
59     tree = ET.ElementTree(file = 'books.xml')
60     root = tree.getroot()
61
62     for child_tag in root:
63         print(child_tag.tag, child_tag.attrib, child_tag.text)
64
65
66 3. xpath를 이용한 순환처리
67 1)특정 tag 찾기
68     import xml.etree.ElementTree as ET
69
70     tree = ET.ElementTree(file = 'doc.xml')
71     root = tree.getroot()
72
73     print(root.find('branch').tag)
74     print(root.findtext('branch'), end="")
75     b = root.findall('branch')
76     for child_tag in b:
77         a = child_tag.text
78         print(a, end = " ")
79
80 2)특정 tag의 text를 조회하기
81     import xml.etree.ElementTree as ET
82
83     tree = ET.ElementTree(file = 'doc.xml')
84     root = tree.getroot()
85
86     print(root.findtext('branch/sub-branch'))
87     print(root.find('branch/sub-branch').text)
88
89 3)반복자를 이용해서 tag 찾기
90     import xml.etree.ElementTree as ET
91
92     tree = ET.ElementTree(file = 'doc.xml')
93     root = tree.getroot()
94
95     for i in root.iterfind('branch'):
96         print(i.tag, i.attrib, i.text)
97
98 4)특정 하위 요소에서 찾기
99     import xml.etree.ElementTree as ET
100
101     tree = ET.ElementTree(file = 'doc.xml')
102     root = tree.getroot()
```

```
103
104     for i in root.iterfind('*//sub-branch'):
105         print(i.tag, i.attrib, i.text)
106
107 5)특성 속성을 갖고 있는 요소 찾기
108     import xml.etree.ElementTree as ET
109
110     tree = ET.ElementTree(file = 'doc.xml')
111     root = tree.getroot()
112
113     for i in root.iterfind("branch[@name='invalid']"):
114         print(i.tag, i.attrib, i.text)
115
116 6)문서에 있는 모든 text 조회해서 출력하기
117     import xml.etree.ElementTree as ET
118
119     tree = ET.ElementTree(file = 'doc.xml')
120     root = tree.getroot()
121
122     for i in root.itertext():
123         print(i, end="")
124
125
126 4. XML 문서 생성하기
127 1)Element 생성해서 요소 만들기(append() 이용)
128     import xml.etree.ElementTree as ET
129
130     root = ET.Element('root')
131     print(root)
132     print(root.tag)
133
134     child = ET.Element('child')
135     print(child)
136     print(child.tag)
137
138     root.append(child)
139
140     ET.dump(root)
141
142 2)Element 생성해서 요소 만들기(SubElement class 이용)
143     import xml.etree.ElementTree as ET
144
145     root = ET.Element('root')
146     print(root)
147     print(root.tag)
148
149     ET.SubElement(root, 'child')
150
151     ET.dump(root)
152
153 3)Element의 insert() 사용해서 요소의 특정 위치 지정하기
```

```
154     import xml.etree.ElementTree as ET
155
156     root = ET.Element('root')
157     print(root)
158     print(root.tag)
159
160     ET.SubElement(root, 'child1')
161     child2 = ET.Element('child2')
162     root.insert(2, child2)    # 2번째 위치에 삽입
163     ET.dump(root)
164
165 4)remove()를 이용하여 특정 요소 삭제하기
166     import xml.etree.ElementTree as ET
167
168     root = ET.Element('root')
169     print(root)
170     print(root.tag)
171
172     ET.SubElement(root, 'child1')
173     child2 = ET.Element('child2')
174     root.insert(2, child2)
175
176     root.remove(child2)
177     ET.dump(root)
178
179 5)Element 생성하며 속성 추가하기
180     import xml.etree.ElementTree as ET
181
182     #<book author="Michael Jackson" />
183     book = ET.Element('book', author = 'Michael Jackson')
184     print(sorted(book.keys()))
185
186     for name, value in sorted(book.items()):
187         print('%s = %r' % (name, value))
188
189     ET.dump(book)
190
191 6)Element 생성하며 속성 추가하기 - set() 사용하기
192     import xml.etree.ElementTree as ET
193
194     #<book author="Michael Jackson" />
195     #<book price="25000" />
196     book = ET.Element('book', author = 'Michael Jackson')
197     book.set("price", '25000')
198     print(sorted(book.keys()))
199
200     for name, value in sorted(book.items()):
201         print('%s = %r' % (name, value))
202
203     ET.dump(book)
204
```

```
205 7)Element의 속성을 dict로 관리해서 검색 및 수정하기
206     import xml.etree.ElementTree as ET
207
208     book = ET.Element('book', author = 'Michael Jackson')
209     book.set("price", '25000')
210
211     print(sorted(book.keys()))
212
213     for name, value in sorted(book.items()):
214         print('%s = %r' % (name, value))
215
216     ET.dump(book)
217
218     attributes = book.attrib
219     print(attributes['author'])
220     #print(attributes['title']) #Error
221     attributes['title'] = 'Python Fundamental'
222     print(attributes['title'])
223     print(book.get('title'))
224
225     ET.dump(book)
226
227
228 5. XML을 문자열로 처리하기
229 1)XML 문서를 만들고 문자열로 보기
230     import xml.etree.ElementTree as ET
231
232     root = ET.Element('root')
233     print(root)
234     print(root.tag)
235     root.append(ET.Element('child1'))
236     child2 = ET.SubElement(root, 'child2')
237     child3 = ET.SubElement(root, 'child3')
238     print(ET.tostring(root))
239
240 2)fromstring()을 tostring()으로 읽기
241     import xml.etree.ElementTree as ET
242
243     xml_ = """<?xml version="1.0"?>
244             <books><book id='test1'><title>Python
                Fundamental</title></book></books>"""
245     xml_str = ET.fromstring(xml_)
246     print(ET.tostring(xml_str))
247
248 3)BytesIO 이용하기
249     import xml.etree.ElementTree as ET
250     from io import BytesIO
251
252     file_like_object = BytesIO(b"<books><book id='test1'><title>Python
Fundamental</title></book></books>")
253     tree = ET.parse(file_like_object)
```

```
254     root = tree.getroot()
255     print(ET.tostring(root))
256
257 4)StringIO도 가능
258     import xml.etree.ElementTree as ET
259     from io import StringIO
260
261     file_like_object = StringIO("<books><book id='test1'><title>Python
Fundamental</title></book></books>")
262     tree = ET.parse(file_like_object)
263     root = tree.getroot()
264     print(ET.tostring(root))
265
266     book = root.find('book')
267     print(book.tag)
268
269 5)iterparse()를 통한 parsing
270     import xml.etree.ElementTree as ET
271     from io import BytesIO
272
273     file_like_object = BytesIO(b"<books><book id='test1'><title>Python
Fundamental</title></book></books>")
274
275     for event, element in ET.iterparse(file_like_object):
276         if element.tag == 'title':
277             print(element.text)
278         elif element.tag == 'book':
279             print('It has subtrees.')
280             element.clear()
281
282 6)문자열로 XML을 만들고 특정 태그를 검색해서 조회해서 출력
283     import xml.etree.ElementTree as ET
284
285     input = """<books>
286             <book id='bk1'>
287                 <title>Python Fundamental</title>
288                 <author>Michael Jackson</author>
289                 <price>25000</price>
290             </book>
291             <book id='bk2'>
292                 <title>Machine Learning Fundamental</title>
293                 <author>Sujan</author>
294                 <price>35000</price>
295             </book>
296         </books>"""
297
298     root = ET.fromstring(input)
299     books = root.findall('book')
300     print('Book count :', len(books))
301
302     for book in books:
```

```
303     print('Title =', book.find('title').text)
304     print('Author =', book.find('author').text)
305     print('Price =', book.find('price').text)
306     print('-' * 20)
307
308
309 6. XML file로 내보내기
310 1)write()로 파일 저장하기
311     import xml.etree.ElementTree as ET
312
313     input = """<books>
314             <book id='bk1'>
315                 <title>Python Fundamental</title>
316                 <author>Michael Jackson</author>
317                 <price>25000</price>
318             </book>
319             <book id='bk2'>
320                 <title>Machine Learning Fundamental</title>
321                 <author>Sujan</author>
322                 <price>35000</price>
323             </book>
324         </books>"""
325
326     root = ET.fromstring(input)
327     tree = ET.ElementTree(root)
328     ver = ET.Element('docversion')
329     ver.text = '1.2.1'
330     root.append(ver)
331
332     tree.write(open('mybook.xml', 'wb'))
333
334     tree = ET.parse('mybook.xml')
335     root_ = tree.getroot()
336     print(ET.tostring(root_))
337
338
339
340
341
342
343
344
345
346
```