Bases de données et web – Python

Lucas Pastor

L3 Informatique & L3 MIASH

2018–2019

Python – XML

Utilisation de python et de l'API **ElementTree**.

ElementTree :

- API XML en python.
- Permet de parser un document XML.
- Facile à manipuler.

Cours extraits de `https://docs.python.org`

Document XML utilisé

```xml
<?xml version="1.0" encoding="utf-8"?>
<data>
    <country name="Liechtenstein">
        <rank>1</rank>
        <year>2008</year>
        <gdppc>141100</gdppc>
        <neighbor name="Austria" direction="E"/>
        <neighbor name="Switzerland" direction="W"/>
    </country>
    <country name="Singapore">
        <rank>4</rank>
        <year>2011</year>
        <gdppc>59900</gdppc>
        <neighbor name="Malaysia" direction="N"/>
    </country>
    <country name="Panama">
        <rank>68</rank>
        <year>2011</year>
        <gdppc>13600</gdppc>
        <neighbor name="Costa Rica" direction="W"/>
        <neighbor name="Colombia" direction="E"/>
    </country>
</data>
```

Lecture du document et récupération de la racine

```
1  # -*- coding: utf-8 -*-
2  #!/bin/python
3
4  # Importation de l'API.
5  import xml.etree.ElementTree as ET
6
7  # Lecture du fichier XML. L'arbre est
8  # stocké dans tree.
9  tree = ET.parse('country_data.xml')
10
11 # Récupération de la racine de l'arbre XML.
12 root = tree.getroot()
```

Lecture du document et récupération de la racine

```
1  # -*- coding: utf-8 -*-
2  #!/bin/python
3
4  # Importation de l'API.
5  import xml.etree.ElementTree as ET
6
7  # Lecture du fichier XML. L'arbre est
8  # stocké dans tree.
9  tree = ET.parse('country_data.xml')
10
11 # Récupération de la racine de l'arbre XML.
12 root = tree.getroot()
```

Récupération de la balise et des attributs

```
>>> root.tag
'data'

>>> root.attrib
{}
```

Lecture du document et récupération de la racine

```
1  # -*- coding: utf-8 -*-
2  #!/bin/python
3
4  # Importation de l'API.
5  import xml.etree.ElementTree as ET
6
7  # Lecture du fichier XML. L'arbre est
8  # stocké dans tree.
9  tree = ET.parse('country_data.xml')
10
11 # Récupération de la racine de l'arbre XML.
12 root = tree.getroot()
```

Récupération de la balise et des attributs

```
>>> root.tag
'data'

>>> root.attrib
{}
```

Itérer sur les nœuds enfants

```
# Itération sur les enfants d'un noeud.
>>> for child in root:
...     print child.tag, child.attrib
country {'name': 'Liechtenstein'}
country {'name': 'Singapore'}
country {'name': 'Panama'}
```

```
# On peut également accéder aux nœuds enfants via '[ ]'.
>>> root[0].attrib['name']
'Liechtenstein'

>>> root[1].attrib['name']
'Singapore'
```

```
# On peut également accéder aux nœuds enfants via '[ ]'.
>>> root[0].attrib['name']
'Liechtenstein'

>>> root[1].attrib['name']
'Singapore'
```

Remarque

Les attributs sont stockés sous forme de dictionnaire.

```
# On peut également accéder aux nœuds enfants via '[ ]'.
>>> root[0].attrib['name']
'Liechtenstein'

>>> root[1].attrib['name']
'Singapore'
```

Remarque

Les attributs sont stockés sous forme de dictionnaire.

```
# Exemple de dictionnaire.
>>> print dict
{'attr1': '1', 'attr2': '2'}

>>> dict['attr1']
'1'

>>> dict['attr2']
'2'
```

Trouver des éléments et des attributs

```
# Itération sur tous les éléments correspondant
# à une balise donnée.
>>> for country in root.findall('country'):
...     rank = country.find('rank').text
...     name = country.get('name')
...     print name, rank
Liechtenstein 1
Singapore 4
Panama 68
```

Trouver des éléments et des attributs

```
# Itération sur tous les éléments correspondant
# à une balise donnée.
>>> for country in root.findall('country'):
...     rank = country.find('rank').text
...     name = country.get('name')
...     print name, rank
Liechtenstein 1
Singapore 4
Panama 68
```

Remarque

La fonction ***find*** retourne un élément. La fonction ***get*** retourne la valeur d'un attribut. On accède à la valeur d'un élément avec ***.text***.

Création de document XML

```
1  # Importation de l'API.
2  import xml.etree.cElementTree as ET
3
4  # Création de la racine.
5  root = ET.Element("root")
6
7  # Ajout d'un noeud enfant de la racine.
8  doc = ET.SubElement(root, "doc")
9
10 ET.SubElement(doc, "field1", attr1="blah", attr2="2").text =
       "some value1"
11 ET.SubElement(doc, "field2", attr1="asdfasd").text = "some
     value2"
12
13 tree = ET.ElementTree(root)
14 tree.write("filename.xml")
```

Création de document XML

```python
# Importation de l'API.
import xml.etree.cElementTree as ET

# Création de la racine.
root = ET.Element("root")

# Ajout d'un noeud enfant de la racine.
doc = ET.SubElement(root, "doc")

ET.SubElement(doc, "field1", attr1="blah", attr2="2").text =
    "some value1"
ET.SubElement(doc, "field2", attr1="asdfasd").text = "some
    value2"

tree = ET.ElementTree(root)
tree.write("filename.xml")
```

```xml
<root>
    <doc>
        <field1 attr1="blah" attr2="2">some value1</field1>
        <field2 attr1="asdfasd">some value2</field2>
    </doc>
</root>
```

Fonction

```
1  # Définition d'une fonction prenant deux paramètres x et y et retournant la somme.
2  def somme(x, y):
3      result = x + y
4      return result
5
6  # Appel de fonction.
7  somme(1, 2)
```

Fonction

```
1  # Définition d'une fonction prenant deux paramètres x et y et retournant la somme.
2  def somme(x, y):
3      result = x + y
4      return result
5
6  # Appel de fonction.
7  somme(1, 2)
```

Remarque

L'indentation est **importante** en Python. C'est ce qui délimite un bloque de code.

Fonction

```
1  # Définition d'une fonction prenant deux paramètres x et y et retournant la somme.
2  def somme(x, y):
3      result = x + y
4      return result
5
6  # Appel de fonction.
7  somme(1, 2)
```

Remarque

L'indentation est **importante** en Python. C'est ce qui délimite un bloque de code.

Liste

```
1  # Création d'une liste vide.
2  myList = []
3
4  # Ajout d'un élément dans une liste.
5  myList.append('element')
6
7  # Itération sur une liste.
8  for myElement in myList:
9      print myElement
```

Boucle

```
1 # Boucle for.
2 for i in range(1, 10)
3     print i
```

Boucle

```
1          # Boucle for.
2          for i in range(1, 10)
3              print i
```

Remarque

La boucle ci-dessous affichera 1, 2, 3, 4, 5, 6, 7, 8, 9.

Boucle

```
1 # Boucle for.
2 for i in range(1, 10)
3     print i
```

Remarque

La boucle ci-dessous affichera 1, 2, 3, 4, 5, 6, 7, 8, 9.

Conditions

```
1 # Conditions.
2 if a == b:
3     print a
4 elif a > 3:
5     print b
6 elif a < 2 and b > 3:
7     print a, b
8 else:
9     print a+b
```

Exercice – Validation du championnat

```xml
<?xml version="1.0" encoding="utf-8"?>
<championnat>
    <clubs>
        <club id="LOSC">
            <nom>LOSC Lille Metropole</nom>
            <ville>Lille</ville>
        </club>
        <club id="SRFC">
            <nom>Stade Rennais FC</nom>
            <ville>Rennes</ville>
        </club>
    </clubs>
    <journees>
        <journee num="1">
            <date>2013-08-10</date>
            <rencontre>
                <clubReceveur>ACA</clubReceveur>
                <clubInvite>ASSE</clubInvite>
                <score>0 1</score>
            </rencontre>
            <rencontre>
                <clubReceveur>GB</clubReceveur>
                <clubInvite>ASMFC</clubInvite>
                <score>0 2</score>
            </rencontre>
        </journee>
        <journee num="2">
            <date>2013-08-17</date>
            <rencontre>
```

Exercice

Vérifier que les clubs se rencontrent tous deux fois. Une fois à domicile et une fois à l'extérieur. Afficher les couples de clubs manquants.

```python
#!/bin/python
# -*- coding: utf-8 -*-

# Importation de l'API.
import xml.etree.ElementTree

# Chemin vers le fichier xml.
FILE = 'championnat.xml'

# Création du parseur.
document = xml.etree.ElementTree.parse(FILE)
p = document.getroot()

# Retourne la liste des clubs.
def get_list_club(parser):
    # La liste des IDs que l'on va mettre à jour au fur et à mesure.
    l = []

    # On va itérer sur tous les clubs.
    for c in parser.findall('clubs/club'):

        # On ajoute l'id trouvé dans la liste.
        l.append(c.attrib['id'])

    return l
```

```
26
27 # Cette fonction retourne vrai si la paire ordonnée (id1, id2) est présente
28 # dans une rencontre.
29 def is_pair_in_rencontres(id1, id2, parser):
30
31     # On itère sur toutes les rencontres.
32     for r in parser.findall('journees/journee/rencontre'):
33
34         # On récupère l'id du club receveur.
35         current_id1 = r.find('clubReceveur').text
36
37         # On récupère l'id du club invite.
38         current_id2 = r.find('clubInvite').text
39
40         # Si la paire ordonnée passée en paramètre est la même
41         # que celle qu'on vient de trouver, on retourne vraie.
42         if id1 == current_id1 and id2 == current_id2:
43             return True
```

```
44
45      # Si on arrive jusqu'ici, c'est que l'on a pas trouvé de paire
46      # correspondante. On retourne alors faux.
47      return False
48
49  def check_every_pair(list_ids, parser):
50      for id1 in list_ids:
51          for id2 in list_ids:
52              if id1 != id2:
53                  if not is_pair_in_rencontres(id1, id2, parser):
54                      print('Paire non présente : ' + id1 + ', ' + id2
                            )
55
56
57  l = get_list_club(p)
58  check_every_pair(l, p)
```

Exercice

Recopier championnat.xml en enlevant le score.