Python – XML

L3 Info & L3 MIASHS *TP4 – Corrigé*

Exercice 1.

Question 1. Le script `exercice1.py` affiche le nom des balises enfants de la racine ainsi que leurs attributs. À savoir clubs et journées (qui n'ont pas d'attributs).

Question 2.

exercice1-2.py

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  # Importation de l'API.
5  import xml.etree.ElementTree as ET
6
7  # Chemin vers le fichier xml.
8  FILE = 'championnat.xml'
9
10 # Création du parser et récupération de l'arbre XML du document.
11 root = ET.parse(FILE).getroot()
12
13 def afficher_enfants(subtree):
14     for child in subtree.findall('clubs/club'):
15         print (child.find('nom').text, child.attrib)
16
17     for child in subtree.findall('journees/journee'):
18         print (child.find('date').text, child.attrib)
19
20 afficher_enfants(root)
```

Exercice 2.

Question 1.

exercice2-1.py

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4  # Importation de l'API.
5  import xml.etree.ElementTree as ET
6
7  # Chemin vers le fichier xml.
8  FILE = 'championnat.xml'
9
10 # Création du parser et récupération de l'arbre XML du document.
11 root = ET.parse(FILE).getroot()
12
13 for child in root.findall('journees/journee'):
14     l = child.findall('rencontre')
15     if len(l) != 10:
16         print('La journée ' + child.attrib['num'] + ' n\'a pas exactement 10
               rencontres.')
```

Question 2.

exercice2-2.py

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

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

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

# Création du parser et récupération de l'arbre XML du document.
root = ET.parse(FILE).getroot()


# On récupère le sous-arbre des journees.
subroot = root.find('journees')

# Pour chaque journee dans journees, on supprime l'élément score.
for rencontre in subroot.findall('journee/rencontre'):
    score = rencontre.find('score')
    rencontre.remove(score)

# Écriture du nouveau document XML.
subtree = ET.ElementTree(subroot)
subtree.write('exercice2-2.xml')
```

Exercice 3.

Question 1.

exercice3-1.py

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

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

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

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

# Retourne la liste des clubs.
def get_list_club(r):
    # 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 r.findall('clubs/club'):

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

    return l

# Cette fonction retourne vrai si la paire ordonnée (id1, id2) est présente
# dans une rencontre.
def is_pair_in_rencontres(id1, id2, parser):

    # On itère sur toutes les rencontres.
    for r in parser.findall('journees/journee/rencontre'):

        # On récupère l'id du club receveur.
        current_id1 = r.find('clubReceveur').text

        # On récupère l'id du club invite.
        current_id2 = r.find('clubInvite').text

```

```
        # Si la paire ordonnée passée en paramètre est la même
        # que celle qu'on vient de trouver, on retourne vraie.
        if id1 == current_id1 and id2 == current_id2:
            return True

    # Si on arrive jusqu'ici, c'est que l'on a pas trouvé de paire
    # correspondante. On retourne alors faux.
    return False

def check_every_pair(list_ids, parser):
    for id1 in list_ids:
        for id2 in list_ids:
            if id1 != id2:
                if not is_pair_in_rencontres(id1, id2, parser):
                    print('Paire non présente : ' + id1 + ', ' + id2)


l = get_list_club(root)
check_every_pair(l, root)
```

Question 2.

exercice3-2.py

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

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

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

# Création du parser et récupération de l'arbre XML du document.
root = ET.parse(FILE).getroot()

# Retourne la liste des clubs.
def get_list_club(r):
    # 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 r.findall('clubs/club'):

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

    return l

# Pour un club donné, retourne l'équipe contre lequel les club a le plus marqué.
def get_best_enemy(club, dictionary, r):
    maxClub = club
    scoreMax = 0

    for rencontre in r.findall('journees/journee/rencontre'):
        clubReceveur = rencontre.find('clubReceveur').text
        clubInvite = rencontre.find('clubInvite').text
        scores = rencontre.find('score').text.split()
        score1 = scores[0]
        score2 = scores[1]

        # Si le club est receveur, il faut regarder score1.
        if club == clubReceveur:
            if score1 > scoreMax:
                scoreMax = score1
                maxClub = clubInvite

        # Si le club est invité, il faut regarder score2.
        if club == clubInvite:
            if score2 > scoreMax:
```

```
                scoreMax = score2
                maxClub = clubReceveur

    return maxClub

# Récupération de la liste des clubs.
clubList = get_list_club(root)

# Création d'un dictionnaire vide.
butDict = {}

# Création d'un dictionnaire ayant comme clefs les IDs des clubs et comme
# valeur un couple d'entiers (x, y) où x est le nombre de buts marqués
# et y le nombre de buts reçus.
for c in clubList:
    butDict[c] = [0, 0]


# On itère sur chaque rencontre.
for rencontre in root.findall('journees/journee/rencontre'):
    clubReceveur = rencontre.find('clubReceveur').text
    clubInvite = rencontre.find('clubInvite').text
    scores = rencontre.find('score').text.split()
    score1 = scores[0]
    score2 = scores[1]

    # Mise à jour du dictionnaire.
    #print(butDict[clubReceveur])
    butDict[clubReceveur][0] += int(score1)
    butDict[clubReceveur][1] += int(score2)

    butDict[clubInvite][0] += int(score2)
    butDict[clubInvite][1] += int(score1)


for c in clubList:
    bestEnemy = get_best_enemy(c, butDict, root)
    print('Le club ' + c + ' a pour marqué le plus de but contre ' + str(bestEnemy)
        )
```

Exercice 4.

exercice4.py

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

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

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

# Création du parser et récupération de l'arbre XML du document.
root = ET.parse(FILE).getroot()


# On récupère le sous-arbre des journees.
subroot = root.find('journees')

# On peut supprimer les journées de l'arbre à écrire en XML.
root.remove(subroot)

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

```

```
25      # On va itérer sur tous les clubs.
26      for c in r.findall('clubs/club'):
27
28          # On ajoute l'id trouvé dans la liste.
29          l.append(c.attrib['id'])
30
31      return l
32
33  # Ajoute la rencontre à la racine.
34  def ajouter_rencontre(rencontre, r):
35      newRencontre = ET.SubElement(r, 'rencontre')
36      ET.SubElement(newRencontre, 'clubReceveur').text = rencontre.find('clubReceveur
            ').text
37      ET.SubElement(newRencontre, 'clubInvite').text = rencontre.find('clubInvite').
            text
38      ET.SubElement(newRencontre, 'score').text = rencontre.find('score').text
39
40  # On récupère la liste des clubs.
41  clubList = get_list_club(root)
42
43  for club in clubList:
44      # Ajout d'une balise rencontres au club.
45      currentClub = root.find('clubs/club[@id=\'' + str(club) + '\']')
46      rencontres = ET.SubElement(currentClub, 'rencontresReceveur')
47
48      for rencontre in subroot.findall('journee/rencontre[clubReceveur=\'' + str(club
            ) + '\']'):
49          ajouter_rencontre(rencontre, rencontres)
50
51  # Écriture du nouveau document XML.
52  tree = ET.ElementTree(root)
53  tree.write('exercice4.xml')
```

Exercice 5.

exercice5.py

```
1   #!/usr/bin/env python
2   # -*- coding: utf-8 -*-
3
4   # Importation de l'API.
5   import xml.etree.ElementTree as ET
6   from datetime import datetime
7
8   # Chemin vers le fichier xml.
9   FILE = 'championnat.xml'
10
11  # Création du parser et récupération de l'arbre XML du document.
12  root = ET.parse(FILE).getroot()
13
14  def is_pair_correct(journee1, journee2):
15      id1 = int(journee1.attrib['num'])
16      id2 = int(journee2.attrib['num'])
17
18      date1 = journee1.find('date').text
19      date2 = journee2.find('date').text
20
21      dateTime1 = datetime.strptime(date1,'%Y-%m-%d')
22      dateTime2 = datetime.strptime(date2,'%Y-%m-%d')
23
24      # On vérifie que si id1 est inférieur à id2, alors il en est de même pour
25      # la date.
26      if (id1 < id2) and (dateTime1 < dateTime2):
27          return True
28      elif (id2 < id1) and (dateTime2 < dateTime1):
29          return True
30      else:
31          return False
```

```
32
33
34 # On récupère le sous-arbre des journees.
35 subroot = root.find('journees')
36
37 # On récupère les journées dans une liste.
38 listJournees = subroot.findall('journee')
39
40 for i in range(0, len(listJournees)):
41     for j in range(i+1, len(listJournees)):
42         if not(is_pair_correct(listJournees[i], listJournees[j])):
43             print('Problème de chronologie entre journée ' + str(i+1) + ' et journé
                  e ' + str(j+1))
```