Rappels Python

L3 Info & L3 MIASHS *Rappels*

1 Indentation du code

En Python la délimitation des bloques de code est indiqué par l'indentation. **Attention**, il ne faut pas mixer sur la même ligne de code le caractère `tab` et les espaces.

```
1     for i in range(0, 10):
2         # Les deux instructions qui suivent sont dans la boucle for.
3         k = i + 1
4         print(k)
5
6     # Cette instruction est en dehors de la boucle for.
7     print('Hello World!')
```

2 Obtenir la taille d'une liste

```
1     myList = []
2
3     # Retourne la taille de la liste.
4     len(myList)
```

3 Supprimer un sous-arbre

```
1     # Racine du document.
2     root = ET.parse(FILE).getroot()
3
4     # Sous-racine.
5     subroot = root.find('journees')
6
7     # Suppression de la sous-racine.
8     root.remove(subroot)
```

4 Écrire un document XML

```
1     # Transformation de la racine en arbre.
2     tree = ET.ElementTree(root)
3
4     # Écriture du document.
5     tree.write('document.xml')
```

5 Utilisation d'un dictionnaire

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

    # Ajout d'une liste de deux éléments ayant comme clef
    # 'score1'.
    dictionnaire['score1'] = [1, 0]

    # Affichage du premier élément correspondant à la clef 'score1'.
    print( dictionnaire['score1'][0] )
```

6 Créer un élément sans parent

```
    # Création d'un élément sans parent.
    root = ET.Element('root')
```

7 Ajouter un nœud enfant

```
    # Création d'une balise child enfant de root ayant comme valeur la chaîne
    # caractères 'valeur'. Le noeud nouvellement créé est également récupéré
    # dans la variable child.
    child = ET.SubElement(root, 'child').text = 'valeur'

    # Il est également possible d'ajouter directement un noeud enfant existant.
    # Ajout de child2 à root.
    root.append(child2)
```

8 Conversion entiers et chaînes de caractères

```
    # Transformation d'un entier en chaîne de caractères.
    s = str(1)

    # Transformation d'une chaîne de caractères en entier.
    i = int('1')
```

9 Concaténation de chaînes de caractères

```
    s1 = 'texte 1'
    s2 = 'texte 2'

    # Concaténation à l'aide de l'opérateur +.
    s = s1 + s2
```

10 Séparation d'une chaîne de caractères

```
    # Scores de jeu séparés par un espace.
    s = '4 2'

    # On sépare la chaîne au niveau des espaces. Cela renvoie une liste de chaînes
    # caractères.
    scores = s.split()

```

```
    # Affichage du premier score.
    print(scores[0])

    # Affichage du deuxième score.
    print(scores[1])
```

11 Récupérer la valeur d'un attribut

```
    valeur = club.get('id')
```

12 Ajouter un attribut à une balise

```
    # Ajoute un attribut nommé 'nom' et ayant pour valeur
    # '1' à la balise root.
    root.set('nom', '1')
```

13 Échappement des quotes dans les requêtes XPath

```
    # XPath avec prédicat. Il faut échapper les quotes.
    resultat = root.find('clubs/club[@id=\'OM\']')
```

14 Comparaison des dates

```
    # Il faut importer la bibliothèque de fonctions.
    from datetime import datetime

    # Chaînes de caractères contenant des dates.
    date1 = '2018-11-25'
    date2 = '2018-12-11'

    # Il faut transformer les chaînes de caractères en variables datetime.
    # Pour cela on indique à datetime le format des dates contenues dans
    # les chaînes # de caractère. En l'occurence nos dates sont de la
    # forme année-mois-jour.
    dateTime1 = datetime.strptime(date1, '%Y-%m-%d')
    dateTime2 = datetime.strptime(date2, '%Y-%m-%d')

    # Ensuite on compare les variables datetime avec les opérateurs
    # classiques =, !=, <, >.
    if (dateTime1 < dateTime2):
        return True
```