Python pour la science des données

Examen pratique de fin de semestre

Durée : 2h

Documents autorisés

Avertissement :

- *Comme pour tout examen universitaire, la communication avec une tierce personne pendant cet épreuve est strictement interdite quel qu'en soit le moyen.*
- *Les travaux fournis doivent être exclusivement originaux et personnels. Toute copie ou recopie de code sera considéré comme du plagiat qui sera sanctionné en conséquence. Nous rappelons que le plagiat est non seulement une fraude dans le cadre d'un examen, mais aussi un délit puni par la loi. Tous les codes fournis seront comparés entre eux à l'aide d'un outil automatique de détection de plagiat.*

Préliminaires :

- Télécharger l'archive *examen.zip* depuis l'espace-cours Cursus et extraire ses fichiers dans un répertoire personnel de travail.

Remarques importantes :

- Les deux exercices sont indépendants mais les données sont décrites dans l'exercice 1.
- Les deux exercices sont à réaliser dans deux fichiers Python séparés (*exo1.py* et *exo2.py*).

Exercice 1 : Extraction d'informations dans un fichier XML *(9 points)*

Présentation des données

Le fichier *rennes-logement.xml* contient des données au format XML sur les résidences du CROUS en Bretagne.

L'élément principal du document XML est *<root>*. Il contient une liste d'éléments *<residence ...>* contenant chacun les informations d'une des résidences proposées par le CROUS en Bretagne.

Informations de résidence (rattachées à un élément *<residence ...>*) :

- identifiant : attribut *id*
- nom : attribut *title*
- description courte : attribut *short_desc*
- coordonnées GPS : attributs *lat* (latitude) et *lon* (longitude)
- zone géographique : attribut *zone*
- informations générales : contenu de l'élément *<infos>*
- services proposés par la résidence : contenu de l'élément *<services>* (liste de services séparés par des virgules)
- informations de contact : contenu de l'élément *<contact>*

Remarque *: les éléments <infos>, <services> et <contact> sont destinés à être affichés sur une page web et peuvent contenir du code HTML.*

Question 1.1 *(0,5 points)* : à l'aide du module `etree` de `lxml`, charger les données du fichier *rennes-logement.xml* dans un arbre XML et récupérer le nœud racine de cet arbre.

Question 1.2 *(1,5 points)* : en utilisant une seule expression Xpath, extraire le texte des éléments *<contact>* pour les résidences dont la zone géographique contient « Villejean ». *Indication : 7 éléments correspondent.*

Question 1.3 *(2,5 points)* : en utilisant une seule expression Xpath, extraire les éléments *<residence>* correspondant à des résidences à Rennes proposant dans leurs services un local à vélos mais **pas** d'accès sécurisé. *Indication : 2 éléments correspondent. Remarque : lorsqu'un service est écrit en premier dans la liste de services, il commence par une majuscule*

Question 1.4 *(3,5 points)* : écrire le code Python permettant d'extraire les informations des résidences et d'obtenir une liste de dictionnaires où chaque dictionnaire représente une résidence avec les informations suivantes : le nom (clé `'nom'`), la description courte (clé `'description'`), la latitude (clé `'latitude'`), la longitude (clé `'longitude'`) et la zone géographique (clé `'zone'`).

```
Ex : [{'nom': 'LA GARE',
     'description': 'Résidence du centre ville de Rennes à
proximité immédiate de la gare SNCF, de la gare routière et
du métro.',
     'latitude': '48.1040703',
     'longitude': '-1.671259100000043',
     'zone': 'Rennes Centre'
     },
     {'nom: 'JULES FERRY',
     …
     …
        ]
```

Question 1.5 *(1 points)* : exporter le dictionnaire de la question précédente au format JSON dans un fichier nommé *resultat.json.* Votre code doit permettre d'obtenir le même contenu et le même formatage que celui du fichier *residences.json* disponible dans l'archive.

Exercice 2 : Données JSON et programmation objet *(11 points)*

Le but de l'exercice est de créer et d'utiliser des objets « `Résidence` » représentant les résidences du CROUS décrites à l'exercice 1. Les données utiles pour cet exercice sont contenues dans le fichier *residences.json* au format JSON. Leur structure est décrite à la question 1.4 de l'exercice 1.

Question 2.1 *(0,5 points)* **:** à l'aide du module `json` de Python charger les données du fichier *residences.json* dans une variable `jsonData`.

Question 2.2 *(2 points)* : créer une classe `Residence` et son constructeur qui, à partir d'un dictionnaire contenant les données d'une résidence, crée un objet `Residence` et valorise ses attributs d'instances avec les correspondances suivantes :

attribut d'instance de *Residence*	clé de dictionnaire d'une résidence
`nom` (type *string*)	`'nom'`
`desc` (type *string*)	`'description'`

`lat` (type *string*)	`'latitude'`
`lon` (type *string*)	`'longitude'`
`zone` (liste de *string*)	`'zone'`

Question 2.3 *(1,5 points)* : ajouter à la classe `Residence` une méthode spéciale d'affichage pour qu'une résidence s'affiche avec un `print()` de la manière suivante :

`Résidence <NOM> (<zone géographique>) : <description>`

Exemple d'affichage de la première résidence du fichier residences.json :

```
Résidence LA GARE (Rennes centre) : Résidence du centre ville
de Rennes à proximité immédiate de la gare SNCF, de la gare
routière et du métro.
```

Question 2.4 *(3 points)* :

2.4.1 : ajouter à la classe `Residence` une méthode `distanceDe()` qui prend comme premiers paramètres deux coordonnées GPS (une longitude et une latitude, toutes deux sous forme d'une chaîne de caractères), en troisième paramètre un profil de parcours (par exemple `'driving-car'`) et qui renvoie la distance en mètres entre la résidence et ces coordonnées, selon le profil de parcours spécifié. Vous pouvez utiliser dans cette méthode la fonction `getJsonResponse()` qui, à partir d'une url de base et un dictionnaire de paramètres, renvoie la réponse au format JSON d'une requête à l' API Matrix du site *openrouteservice.org*. Pour cela il faudra renseigner votre propre clé d'API dans la variable `apiKey` du programme *exo2.py*.

2.4.2 : sur le même principe, ajouter à la classe `Residence` une méthode `dureeDe()` qui renvoie la durée de parcours en secondes entre la résidence et des coordonnées GPS fournies en paramètres, selon un profil de parcours spécifié lui aussi en paramètre.

Remarque pour les questions suivantes 2.5 et 2.6 : l'API Matrix d'openroutservice.org limite le nombre de requêtes par minute à 40. Il faudra veiller à bien respecter les contraintes données pour éviter d'avoir un nombre trop important de requêtes par minute. D'autre part le nombre max de requêtes par jour est de 500.

Question 2.5 *(2 points)* : dans le programme principal, écrire le code Python permettant d'afficher les résidences de Rennes se trouvant à moins de 20 minutes à pied du campus de Villejean (coordonnées : *longitude : -1.702147 ; latitude : 48.118737*) avec la durée correspondante en minutes. Pour ces calculs de durée on se limitera aux objets `Residence` dont l'attribut `zone` contient `'Rennes'`)

Exemple d'affichage pour une des résidences correspondantes :

```
A 14 minutes : Résidence VILLEJEAN OUEST (Rennes ouest-Villejean)
:  Résidence proche de l'université de Rennes II et campus santé
```

Question 2.6 *(2 points)* : dans le programme principal, écrire le code Python permettant d'afficher les résidences en dehors de Rennes (l'attribut `zone` ne contient pas `'Rennes'`), et la distance en voiture entre la résidence et le campus de Villejean, en kilomètres.

Exemple d'affichage pour une des résidences correspondantes :

```
A 239 km : Résidence KERGOAT (Brest) : A proximité des UFR Droit/
Sciences Economiques, AES, STAPS et de l'IUT.
```

Fin de l'examen :

Déposez sur l'espace-cours Cursus vos fichiers ***exo1.py*** et ***exo2.py***.