SAE 105 : Traitement de données – CAHIER Des CHARGES

Objectifs

Compétences professionnelles

En tant que futur professionnel R&T, vous serez régulièrement amené à **traiter des données provenant du système d'information de votre entreprise** (extraction d'information comptable par exemple), pour **les présenter de façon pertinente et synthétique à votre hiérarchie ou à vos collaborateurs**. Ces traitements pouvant être **récurrents** (mensualisation de bilan par exemple), ils gagnent à être **automatisés** par des programmes/scripts informatiques pour les gérer de façon efficace.

La SAE 105 vous projette dans cette **situation professionnelle**. Elle vous propose d'exploiter des **données issues de fichiers csv**, de les traiter et de les présenter pour vos **collaborateurs universitaires** (services administratifs, financiers, collègues enseignants, collègues étudiants, …).

Contexte de réalisation

La **SAE 105** se déroulera selon le principe suivant :

- Vous êtes en charge, à titre **individuel**, d'un projet de développement informatique.
- Ce projet vous est confié par votre **hiérarchie** (l'enseignant mandataire en charge de votre groupe de TD).
- Votre hiérarchie **impose un cahier des charges fonctionnel et technique** (décrits par la suite), **qui devra être respecté avec des livrables fournis dans les temps** (**dépôt sur Moodle à la fin de chaque séance de TP**) ouverture du dépôt au début de la séance de TP et fermeture du dépôt 10 minutes après la fin du TP. Tout travail non rendu à temps sera sanctionné.

Organisation du temps de travail

Le projet devra être réalisé dans des **plages de travail réservées, obligatoires** et inscrites dans votre emploi du temps (charge homme/mois) :

- une partie des séances de travail seront **encadrées** (2 TPs) par des collaborateurs techniques (enseignants : Jean-Pierre et moi) que vous pourrez solliciter en cas de questions
- une autre partie des séances seront à faire **en autonomie** (7 TPs non encadrés) pour avancer votre travail et à l'issue desquelles un certain nombre de fonctionnalités devront être livrées (dépôt sur Moodle du programme développé en Python).

7 TPs de 3H non encadrés (sem48 à sem3)
2 TPs de 3h encadrés sem50 et sem2 pour répondre à vos questions
Rendre un petit rapport de 5 pages qui explique le projet et les fonctions qui sont utilisées. Pour chaque fonctions et procédures il faudra donner le nom des paramètres d'entrée et de sortie ainsi que leurs types. Il faudra également décrire brièvement ce que fait ce code. Ces informations seront à mettre dans le code en commentaires.

Pour l'**évaluation** en fin de la **semaine 3** : à voir …. soit

- 3h de TP : pour l'**évaluation** : environ **10 minutes** par étudiants.
- 2h de CTRL dans lequel on vous donne un fichier à traiter et on vous demande de coder différentes fonctions que vous avez écrites lors de cette SAE.

1 projet par étudiant, donc à faire tout seul, pour vous habituer à travailler en autonomie.

Le fichier **villes_france.csv** est disponible sur Moodle. L'extension csv = comma separated values, soit données séparées par une virgule.

C'est ce fichier qu'il faudra traiter. Il se trouve également : https://sql.sh/736-base-donnees-villes-francaises

Vous pourrez extraire une partie du fichier pour faire des tests de certaines fonctionnalités avant de les tester sur l'ensemble du fichier.

Le fichier à traiter comporte **27** champs (colonnes) et **36700** lignes :

1 ligne = 1 ville

"1","01","ozan","OZAN","ozan","Ozan","O250","OSN","01190","284","01284","2","26","6","618","469","500","93","6.6","4.91667","46.3833","2866","51546","+45456","462330","170","205"

- **Département :** numéro du département.
- **Slug :** identifiant unique en minuscule, sans accent et sans espace. Peut servir pour faire les URLs d'un site web.
- **Nom :** nom en majuscule et sans accent.
- **Nom simple :** nom en minuscule, sans accent et avec les tirets remplacés par des espaces. Peut être utilisé pour faire une recherche lorsqu'on ne sait pas si le nom de ville possède un tiret ou des espaces (ex : "Saint-Étienne" possède un tiret comme séparateur, tandis que "Le Havre" possède un espace comme séparateur)
- **Nom reel :** nom correct avec les accents
- **Nom soundex :** soundex du nom de la ville (permet de trouver des villes qui se prononcent presque pareil) **[ajouté le 31/10/2013]**
- **Nom metaphone :** metaphone du nom de la ville (même utilité que le soundex) **[ajouté le 31/10/2013]**
- **Code postal :** code postal de la ville. Si la ville possède plusieurs codes postaux, ils sont tous listés et séparés par un tiret **[ajouté le 31/10/2013]**
- **Numéro de commune :** numéro de la commune dans le département. Combiné avec le numéro de département, il permet de créer le code INSEE sous 5 caractères.
- **Code commune (ou code INSEE) :** identifiant unique sous 5 caractères
- **Arrondissement :** arrondissement de la ville
- **Canton :** canton de la ville
- **Population en 2010 :** nombre d'habitants lors du recensement de 2010
- **Population en 1999 :** nombre d'habitants lors du recensement de 1999
- **Population en 2012 (approximatif) :** valeur exprimée en centaine
- **Densité en 2010 :** nombre d'habitants au kilomètre carré arrondie à l'entier. Calculé à partir du nombre d'habitant et de la surface de la ville **[corrigé le 02/07/2014]**
- **Surface / superficie :** surface de la ville en kilomètre carrée **[corrigé le 02/07/2014]**
- **Longitude/latitude en degré :** géolocalisation du centre de la ville. Permet de localiser la ville sur une carte (exemple : carte Google Maps) **[ajouté le 31/10/2013, corrigé le 07/11/2013]**
- **Longitude/latitude en GRD :** géolocalisation exprimée en GRD
- **Longitude/latitude en DMS (Degré Minute Seconde) :** géolocalisation exprimée en Degré Minute Seconde
- **Altitude minimale/maximale :** hauteur minimum et maximum de la ville par rapport au niveau de l'eau

De ce fichier on va extraire les **12 informations** suivantes sur lesquelles nous allons travailler :

Qui sont :
Numéro du département : 01
Nom de la ville : OZAN
Code postal : 01190
nbre d'habitants en 2010 : 618
nbre d'habitants en 1999 : 469
nbre d'habitants en 2012 : 500
densité : 93
surface : 6.6 km^2
longitude : 4.91667
lattitude : 46.3833
altitude min : 170 m
altitude max : 205 m

TRAVAIL A REALISER :

Il faut impérativement respecter les notations données

Pour réaliser cette SAE, n'utiliser que les notions vues en Cours et TP, donc pas de dictionnaire

- **Etape 1 :**
  - **1.1** Faire un programme qui utilise **un menu** : taper 1 pour extraire les villes du fichier, taper 2 pour extraire des ***statistiques*** sur les villes d'un département, taper 3 pour visualiser un histogramme, taper 4 pour calculer la distance entre 2 villes, taper 5 pour le plus court chemin, taper …. (menu provisoire, à améliorer en fonction de l'avancement)
  - **1.2** Pour les ***statistiques*** liées aux villes d'un département on sera redirigé sur un **sous-menu** lié au **4 fonctions** à réaliser (décrites ci-dessous). Prendre exemple sur test_Affiche_Menu_V0.py

  - **1.3** faire une **fonction** **extraction_villes_csv(fichier)** qui extrait toutes **les lignes** du fichier csv et qui les sauvegardent dans une liste. Cette liste sera composée d'une chaîne de caractères (str = string). Une ligne c'est :
    "1","01","ozan","OZAN","ozan","Ozan","O250","OSN","01190","284","01284","2","26","6","618","469","500","93","6.6","4.91667","46.3833","2866","51546","+45456","462330","170","205"

  - **1.4** A partir de la liste précédente nommée **uneListe**, faire une fonction **extract_infos_villes(uneListe)** qui retourne la liste **listeInfo** des **12 informations** retenues pour chaque ville. Chaque élément de la liste doit être au bon format ou type pour pouvoir l'utiliser par la suite.
    Info = [1, 'OZAN',"01190", 618, 469, 500, 93, 6.6, 4.91667, 46.3833, 170, 205]

  - **1.5** A partir de la liste des départements (**listeDept**) pour chaque indicatif téléphonique et de la liste précédente (**uneListe**), faire une fonction **extract_villes_depart_indicatif(listeDept, listeInfo)** qui extrait l'ensemble des villes des **xx** départements en fonction de **l'indicatif téléphonique** de France (01 = île de France, 02 = Nord-Ouest, 03 = Nord-Est, 04 = Sud-Est, 05 = Sud-Ouest). La fonction devra retourner le **nombre de villes** par indicatif téléphonique. Elle

devra également **sauvegarder dans un fichier texte** (voir ci-dessous) la liste des villes. Par exemple le **02** comporte les **22 départements**.
Donc ici on aura 1 groupe de TP par indicatif téléphonique (01 = TP-A1, 02 = TP-A2, 03 = TP-B1, 04 = TP-B2, 05 = TP-C1).

Si vous avez des difficultés pour gérer l'information du **département**, traiter la comme une **chaîne str**, puisque les départements de Corse sont "2A" et "2B".

https://fr.wikipedia.org/wiki/Liste_des_indicatifs_t%C3%A9l%C3%A9phoniques_en_France

La fonction doit retourner un **fichier texte**, contenant le nom des villes pour chaque département.

Les fichiers texte auront les noms suivants :

INDICATIF 01 : **IF01.txt** (Iles de France 01) : 8 départements
INDICATIF 02 : **NO02.txt** **(**Nord-Ouest 02) : 22 départements
INDICATIF 03 : **NE03.txt** (Nord-Est 03) : 23 départements
INDICATIF 04 : **SE04.txt** (Sud-Est 04) : 25 départements
INDICATIF 05 : **SO05.txt** (Sud-Ouest 05) : 27 départements

01 : [75,77,78,91,92,93,94,95]

02 : [14,18,22,27,28,29,35,36,37,41,44,45,49,50,53,56,61,72,76,85,974,976]

03 : [02,08,10,21,25,39,51,52,54,55,57,58,59,60,62,67,68,70,71,80,88,89,90]

04 : [01,03,04,05,06,07,11,13,15,2A,2B,26,30,34,38,42,43,48,63,66,69,73,74,83,84]

05 [09,12,16,17,19,23,24,31,32,33,40,46,47,64,65,79,81,82,86,87,971,972,973,975,977,978]

source : https://fr.wikipedia.org/wiki/Liste_des_indicatifs_téléphoniques_en_France

Par exemple, les 7 premières lignes des fichiers doivent être :

IF01.txt	NO02.txt	NE03.txt	SE04.txt	SO05.txt
1 PARIS (75)	1 SAINT-PIERRE-CANIVET (14)	1 GERCY (02)	1 OZAN (01)	1 BEDEILHAC-ET-AYNAT (09)
2 JOUY-LE-CHATEL (77)	2 GRANGUES (14)	2 VILLEMONTOIRE (02)	2 CORMORANCHE-SUR-SAONE (01)	2 CONTRAZY (09)
3 LESCHES (77)	3 VIEUX-BOURG (14)	3 BRANCOURT-LE-GRAND (02)	3 PLAGNE (01)	3 SUC-ET-SENTENAC (09)
4 SAINT-GERMAIN-SOUS-DOUE (77)	4 ENGLESQUEVILLE-LA-PERCEE (14)	4 LICY-CLIGNON (02)	4 TOSSIAT (01)	4 VENTENAC (09)
5 JUILLY (77)	5 RANCHY (14)	5 PROISY (02)	5 POUILLAT (01)	5 THOUARS-SUR-ARIZE (09)
6 LES CHAPPELLES-BOURDON (77)	6 CASTILLON-EN-AUGE (14)	6 CHAMBRY (02)	6 TORCIEU (01)	6 ARTIX (09)
7 MARCILLY (77)	7 GRENTHEVILLE (14)	7 MONT-SAINT-JEAN (02)	7 REPLONGES (01)	7 TOURTOUSE (09)

- **Etape 2 :**

Pour chaque groupe de TP, nous avons réparti **1 département** par étudiant (voir fichier Doc2 : attribution des départements)

- **2.1** En fonction du département qui vous a été attribué, faire une **fonction** nommé extract_villes_NumDepart(numDept, listeVilles) qui extrait de la liste toutes les villes du numéro de département passé en paramètre, et retourne le **nombre de villes** de ce département et **une liste des villes** de ce département qui contient les 12 informations précédemment extraites. Cette fonction sauvegarde ces villes dans le fichier nommé villes_n°Dept.txt (*exemple villes_74.txt*).
- **2.2** Faire **4 fonctions** qui sauvegardent dans **4 fichiers** :
  Pour trier la liste des villes on utilisera l'algorithme **tri bulle** étudié dans le TP7.
  - **a)** Ecrire la fonction **MinMax5Villes_Habitants(numDept, listeVillesDept)** qui permet de sauvegarder le nom des **5 villes** qui ont **le plus (respectivement le moins) d'habitant**s en **2010**, ainsi que le nombre d'habitants, la superficie et la densité, ... Nom des fichiers Top5Villes_ n°Dept.txt **(nomfich1)**, (respectivement Min5Villes_ n°Dept.txt **(nomfich2)**)
  - **b)** Ecrire la fonction **mapTenVilles(nomfich1, nomfich2)** qui permet d'**afficher** ces **10 villes** sur OpenStreetMap, avec des couleurs différentes en fonction des densités : *un cercle centré sur la ville avec une couleur plus ou moins intense en fonction de la densité*. **nomfich1** et **nomfich2** étant les 2 noms de fichiers de la partie **a)**.
  - **c)** Le nom des 10 villes qui ont eu **le plus fort accroissement** de sa population entre 1999 et 2012. Nom du fichier TopAcc10Villes_ n°Dept.txt.
  - **d)** Le nom des 10 villes qui ont eu **la plus forte baisse** de sa population entre 1999 et 2012, avec la valeur de cette baisse. Nom du fichier TopBaisse10Villes_ n°Dept.txt.

- **ATTENTION, tout ce qui suit est PROVISOIRE, mais vous pouvez tout de même avancer**

- **Etape 3 : (Provisoire)**
  - Pour le département que vous devez traiter, faire un **histogramme des villes** en fonction du nombre d'habitant en 2010, calculer la **moyenne** du nombre d'habitants, **l'écart-type** (*dispersion autour de la moyenne*). Vous répartirez les nombres d'habitants en **10 classes** de **même amplitude**. Utiliser la bibliothèque matplotlib.pyplot pour faire ce tracé.
  - Pour le département faire 2 fonctions, l'une qui répertorie les **5 villes** qui ont **la plus forte différence d'altitude**, l'autre les **5 villes** qui ont **la plus faible différence d'altitude**.
  - Faire une fonction qui permet d'**afficher** ces **10 villes** sur OpenStreetMap, avec des couleurs différentes en fonction ces différences d'altitude.

- **Etape 4 :**
  - A faire en fonction de votre groupe de TP
  - On demande une fonction qui retourne la **distance euclidienne** entre les villes suivantes : a) Paris – Marseille (TP-A1), b) Lille – Bordeaux (TP-A2), c) Strasbourg – Brest (TP-B1), d) Calais – Toulouse (TP-B2), e) Nantes – Nice (TP-C1).
  - On attribue 2 villes pas étudiant, et chaque étudiant calcul la **distance euclidienne** entre ces 2 villes.
  - On demande une **fonction** qui permet de calculer la **distance géodésique** entre ces 2 villes éloignées et 2 villes proches (*50 km*) (pour le calcul voir Doc3 : calcul de la distance géodésique sur Moodle)

- **Etape 5 :**

- Pour chaque étudiant, on demande une fonction qui permet de sauvegarder l'ensemble des villes traversées pour aller de la **ville1** à la **ville2** en utilisant un algorithme basé sur les 10 plus proches voisins. On recherchera les **10 villes** appartenant au disque de **centre ville1** et de **rayon R** (pour les tests, on prendra R = 50 km). On ne conservera que la **villeX** qui est la plus proche de la **ville2**, au sens de la distance euclidienne. **La villeX** devient alors le nouveau centre et on procède de proche en proche jusqu'à arriver à la **ville2**. L'ensemble des villes traversées seront sauvegardées dans un fichier nommé **parcours.txt**.
- Utiliser OpenStreetMap pour tracer le parcours et afficher les villes traversées.
- **Optimiser le trajet** en recherchant le **chemin le plus court** pour aller de ville1 à ville2, en passant par les villes qui ont le plus d'habitants dans un rayon de 100 km.

- **Etape 6 :**
  - Pour les plus rapides, à compléter avec des fonctions de votre choix …