Moad RAZZAKI / Paul BERRA

Mini-projet : Traiter des données

Diapo 2 :

Ce projet aura été organisé en 3 principales parties : Etude et apprentissage, mise en pratique et organisation ainsi que programmation et débogage.

Lors de la 1ere partie qui a était répartie sur environ 4h, nous avons étudié les consignes et les livrables attendus, les spécificités et prérequis du projet, cela a duré environ 1heure.

Nous avons ensuite dû découvrir et nous instruire sur les différentes librairies recommandées par le corps enseignant, les spécificités de chaque librairie, leurs facilités d'intégration et si le résultat fourni par une librairie est bien ce que nous voulions et avons besoin.

Lors de la 2eme partie, nous avons appliqué concrètement les librairies sur des exercices précis donnés par Mr Haddab, nous avons mit en place les fonctions mathématiques permettant de calculer une moyenne, un écart type…

Nous avons ensuite réalisé un schéma de l'organisation du programme : quelles fonctions nous allons faire, ou est-ce-que nous allions les appeler, mais aussi lesquels seront dans des boucles, à quel endroit... nous avons pré étudié en détail le fonctionnement des boucles afin de simplifier au maximum le programme, toutes les boucles ne font pas x tours, seuls celle nécessaires parcourent les listes.

Enfin, lors de la 3eme partie, nous avons mis en place concrètement tout le travail préparatoire réalisé lors des 2 premières parties. Nous avons répartie la dernière étape en environ 13h de programmation pure, mais nous avons aussi passé un peu de temps sur le débogage car entre le schéma et la réalité tout n'était pas applicable exactement comme nous le souhaitions, de plus nous avons du revoir en conséquence le schéma afin d'être sur de rester sur quelque chose de clair, pratique et fonctionnel.

Diapo 3 :

Le script a un fonctionnement plutôt simple et organisé ; dans un premier temps nous importons les modules, ensuite nous définissons des variables utiles (le i qui sert de métronome au programme par ex) ainsi que des listes, les tags de parking, l'url commune. Mais aussi des listes vides ayant pour but d'accueillir les valeurs récupérées sur les différents parking.

La partie violette correspond a notre init () ou main(), elle est en charge de gérer le déploiement des différentes fonctions, dans un premier temps, une fonction va sous gérer les différentes fonctions permettant de récupérer les informations url contenu sur une page xml.

Dans un premier temps, l'url est construite sur un système de i qui se balade dans la liste des tags, ensuite cette url est transmise a la fonction requête qui va effectuer une requête et extraire la réponse puis la stocke dans « texte » qui est lui-même envoyer a la fonction de parsing qui va extraire les données utiles qui se trouvent dans les balises, elle va retourner le nom, le nombre de places libres et total ainsi que le statu du parking. Suite a ca une fonction crée un nom pour un fichier sous le format « places parkings » + la date et l'heure tronquer, les minutes n'apparaissent pas, ainsi lors de l'écriture des fichiers il n'en crée pas un par minute : si un dossier existe pour l'heure en cours, il ajoute a la suite, sinon il en crée un.

Enfin toutes ces données sont écrites dans des fichiers csv afin de constituer des logs.

Diapo 4 :

La partie marron concerne le cas de figure ou i aurais parcouru l'intégralité de la liste des tags, dans ce cas la, une fonction de soustraction places total-places libres nous donnent les places occupés par parking, ensuite cette liste est transmise a la fonction math qui va gérer l'organisation de toutes les fonctions mathématique et de l'écriture de logs sous un format exploitable par Gnu plot.

Dans un premier temps, les donnés utiles sont transmise aux fonction en charge de la moyenne et de l'écart type, qui seront ainsi retourné a la fonction en charge de l'écriture du fichier .Dat et a celles ayant pour but d'afficher proprement la moyenne ainsi que tracer les courbes.

Si jamais la liste n'a pas était complètement parcouru, c'est simple, le programme saute simplement au prochain tour sans retourner aucunes valeurs.

La partie bleue, a pour charge de remplir les listes vides déclarés a la racine avec les différentes valeurs récuperes par « url_requete_ecriture » de manière a ce que la fonction math puisse les exploités, de plus dans le cas ou un lien aurais bugger, un message d'erreur est retourné en tant que variable afind'eviter l'erreur « NoneType » lié a l'absence de données.

Cette partie est déclenchée a la suite de « url_requete_ecriture ».

Diapo 5 :

Enfin, la partie verte est une boucle infinie ayant deux possibités : réalisé toute la boucle init sans erreur critiques, puis est effectué un sleep d'une valeur rentré par l'utilisateur au début, car selon la nature des mesures souhaités, l'échantillonnage doit être réglable

Mais dans l'autre cas, si une erreur réellement critique est remontée, i est tout simplement incrémenté afin de passer a l'url de parking suivant et ainsi ne pas fausser les mesures

Diapo 6 :

Afin de mener a bien ce projet en conservant des nuits de sommeil de plus de 6h, nous avons utilisé des librairies pour nous simplifier le travail et automatiser certaines tâches complexe comme le parsing.

Les librairies utilisés sont :

Math (sqrt) pour kes différents calcul (principalement sigma)

Request, pour effectuer les requetes vers le serveur et récupérer le contenu xml

BeautifulSoup, afin d'effectuer un parsing propre, sans bricolage

DateTime afin d'vaoir l'heure et la date pour le renommage des fichiers

Os, afin de pouvoir crée des dossier pour les logs, et avoir la commande chdir.

Documentation utilisés :

Afin de nous débrouiller au mieux, nous avons recouru a des sites internet pour trouvés certaines informations, notamment stackoverflow pour le parsing, formatage de données et affichage des courbes

Askcodez pour gnuplot aussi

Reddit pour les commandes OS

Docs.python et pypy pour la documentation des librairies

Et enfin moodle qui aura était essentiel.