

Projet Transport en Commun

Le but du projet :

I. Apprendre à programmer

Le but principale est de se détacher de la façon très académique d'apprendre à programmer vue en cours et d'apprendre à plus se débrouiller.

De ce fait le projet est très libre et le sujet pas très cerné, fait ce que vous voulez (niveau de précision, configurabilité, organisation).

4 point que je considère comme essentiel pour apprendre à coder :

1. Programmer

pour apprendre il faut pratiquer c'est comme du sport.

2. Tester


une question ? un doute ? => fait un petit bout de code pour tester, c'est l'avantage du virtuel

3. Être Rigoureux

c'est comme des math, pour faire un code propre et apprendre correctement il faut être rigoureux

Par exemple : enlever les bouts de code qui marche pas / qui sont inutile, mettre quelque commentaire (doc des fonctions), nommer correctement ses variables & cie,

4. Être curieux

Il faut se demander comment ça fonctionne, allez lire de la doc / demander à des ia (ex: Mistral ) , bon demander à des gens c'est mieux et ça pollue moins.

Le but c'est que au fur et à mesure vous me montrez vos codes, de cette façon je peux vous aider sur des choses qui bloquent / donner des conseils sur la façon de coder et etc.

Votre première réaction face à une erreur ne doit pas être d'aller voir sur internet ou demander à quelqu'un mais plutôt d'essayer de trouver une solution, comprendre.

II. Le sujet

Le but est de créer un simulateur de transport en commun, c'est à dire des lignes de transport, et simuler (donc modéliser) le nombre de passager au cours de la journée.

On pourra rajouter un nombre de transport par ligne qui varie au cours du temps, de même pour la demander en transport.

Le but ici est pas de faire une interface graphique, donc la simulation se résumera sûrement a des nombres dans la consoles (ou des fichiers) et des graphiques (`matplotlib` par exemple) mais libre a vous de faire une interface si vous le souhaitez.

III. Pour aller plus loin

Quel que idées pour approfondir la simulations :

- un algo qui va gérer le nombre de transport en fonction de la demande
 - un réseau de neurone pour effectuer la même tâche
 - simuler sur une semaine (ou plus => donc faire des variation entre les jours)
 - interface graphique
-

Conseils :

Les conseils que je vous donne sont globalement général.

I. Organisation

C'est bien d'organiser son espace de travail, donc diviser la simulation en plusieurs fichiers (vous pouvez importer les fonctions des autre fichier comme pour n'importe quel modules), il faut essayer de séparer les fichiers où est coder la simulation et ceux où sont mit les paramètres (nombre de ligne, paramètre des transport, nombre de personnes, etc).

De plus, je vous conseil de séparer en plusieurs sous-dossier, par exemple `data` (pour y exporter des données), `src` (pour *source*, pour mettre le code de la simulation) et `simulation` (pour mettre les fichiers de paramétrage).

II. Documentation

Pensez a mettre de la `docstring` (pas besoin d'un gros pâté, une ligne qui explique ce que fait la fonction est suffisant).

Hésitez pas à mettre des commentaires à coté des lignes dont vous savez que vous allez oublié la signification, de même pour des parties de code où vous n'êtes pas sure de fiabilité / de si sa marche comme vous le souhaité.

Je vous conseil de faire un fichier (par exemple `.md` donc du markdown) pour expliquer le fonctionnement :

- quels sont les buts (et donc les choix / interprétation du sujet que vosu avez faits)
- quel(s) fonction(s) un utilisateur doit-il appeler pour lancer une simulation
- quels sont les paramètres, comment les modifiers
-

III. Préparation

Comme pour n'importe quel travail, il est important de de voir où on vas, donc il faut préparer des "tâches" pour ne pas se perdre.

Donc commencer par choisir (et écrire quel que pars) les buts que vous vous donner, attention n'essayer pas de faire tout les but à la fois, il faut y aller par étapes.

Commencer toujours par quelque chose de simple (le début peut-être le plus dure car c'est pas encore possible de voire le résultat) puis augmentez la complexité au fur et à mesure.

Libre à vous de faire ou non du pseudo code / ou autre, personnellement j'aime pas trop car c'est trop proche du code, donc on a tendance à déjà prévoir comment on vas coder tel ou tel truc, alors que le but est de savoir globalement la tête du programme.

IV. Sauvegarde

Pensez à sauvegardez votre code (dans la plus part des éditeurs de code il y a une option sauvegarde automatique).

Il y a 2 solutions :

Je vous conseil d'utiliser [github](#), sa permet d'avoir un historique de ce que vous avez fait, de ce fait il faire des *commits*^[1] suffisamment souvent et suffisamment détailler pour ne pas s'y perdre. De plus, c'est l'occasion d'apprendre à utiliser cet incontournable outils du développement. Enfin, pour moi c'est plus simple car de cette façon je peux aussi télécharger votre code pour le regarder en détails.

Si vous décider d'utiliser github, fait un *fork*^[2] du *repositorie*^[3] (*repo* pour les intimes) et je vous conseil d'installer l'application Github Desktop pour faciliter l'usage de l'outils.

Ou bien vous télécharger juste le code de ce repo et vous faite avec.

1. C'est les "sauvegardes" des modifications fait au code (par fichier par exemple) ↩

2. Cela vous permet de récupérer tout ce qui est présent dans ce repo et de créer le votre, pensez a le mettre en public. ↩

3. C'est l'espace ou est stocker votre code ainsi que l'historique des modifications. ↩