Description de projet:

- Le but de ce projet est de prendre la base données des lignes de métro de Paris en format .csv et les transformer en .xml et puis les représenter dans un fichier .svg.

- Le fichier '.csv' est sous la forme suivante:
"`Station_id;station_name;station_desc;station_lat;station_lon;stop_sequence;route_id;service_id;direction_id;service_short_name;long_name_first;long_name_last`"

1- CSV → XML

- Lors de la transformation de .csv -> .xml on garde que les infos nécessaires pour représenter les lignes dans le .svg, on ignore donc les attributs "`route_id`" et "`direction_id`"
- On a utilisé la librairie `xml.etree` pour créer le document XML après le traitement des données csv

2- Structure XML

- <metro> root
  - <metro_lines> contient tous les lignes
    - <line> représente une des ligne de metro
      - @service_id identifiant de la ligne

- <short_name> numéro de la ligne metro (1→13 + 3B et 7B)
- <long_name_first> terminus 1
- <long_name_last> terminus 2 / 3
- <stops> les arrêts de la ligne
  - <station> contient le nom de l'arrêt
    - @station_id identifiant de l'arrêt
    - @stop_sequence le numéro d'arrêt dans la ligne
    - @links les autres lignes qu’on peut accéder depuis l'arrêt
  - <fork> la ligne se branche en 2
    - <left> branche gauche, contient une suits des arrêts
      - <station>
    - <right> branche droite, même chose que la branche droite
      - <station>

- <stations> Tous les arrêts dans le métro de Paris avec tous leurs détails
  - <station>
    - @station_id identifiant
    - <station_name> nom

    - <station_desc> description = addresse
    - <station_lat> latitude
    - <station_lon> longitude

Avec cette structure on voulait faire quelque chose simple et logique qui contient tous les infos qu'on a besoin pour faire la transformation en .svg, mais on voulait aussi une structure avec les moindre de redondance

- Dans cette partie, la transformation est simple:
    1. on parcours les lignes (on appliquant le template "//line")
    2. Pour chaque arrêt on crée une cercle pour le représenter, les coordonnées du cette dernière sont calculés avec sa position() (pour x) ou avec le stop séquence de la dernière station avant la fork pour les stations dans <fork><left>/<right> ET le numéro de ligne (pour y)
    3. On attache le nom de l'arrêt qu'on obtient avec `key('station', @station_id)` au cercle
    4. Si la position de l'arrêt est supérieure à 1 (c'est pas le premier arrêt) on dessine un <line> du dernier cercle au nouveau

5. Si l’attribute @links existe dans <station> courant on tokenize ces links et on creer une cercle pour chaque link dans links