Fonctionnement:

Le point d’entrée est le fichier treeDisplay.py qui prend en entrée un type de fichier et un fichier source qui contient une représentation d’arbre.

treeDisplay commence par parser le fichier source selon son type pour en créer une représentation interne d’arbre (tree.py).

treeDisplay calcule ensuite les coordonnées de l’arbre (appelle à la méthode computerCoord de la classe Tree)

treeDisplay affiche finalement le résultat.

-> treeDisplay légèrement modifié. Pour lancer le programme :

python3 treeDisplay.py format entree [-labels]

Il teste si on a au moins 2 paramètres (format d’entrée + fichier d’entrée), puis si le tag “-labels” est présent. Si oui, alors on affiche les labels.

A faire:

1. Il faut donc écrire les différentes fonctions de parsing. J’ai présentement créé des fichiers différents pour que l’on puisse facilement coder chacune de notre côté les différentes fonctions qui permettent de le faire mais on pourra tout regrouper dans un fichier à la fin.
   1. XML à éventuellement perfectionner si jamais ce n’est pas efficace pour les arbres de trop grande taille (<http://effbot.org/zone/element-iterparse.html>)
   2. Dot à modifier pour prendre en compte toute la grammaire dot
2. Ajuster toNetworkX pour, au lieu d’afficher le graphe, générer l’image directement. Nécessite alors de déterminer la résolution de l’image pour que l’on puisse voir les détails. Idée : Si h et l sont respectivement la hauteur et la largeur en pixels de l’image finale :  
   h=depth(tree)\*A ; l = Xmax(tree)\*B ; avec A et B à fixer par des tests.
3. Ajuster toNetworkX pour que les labels s’affichent mieux.
4. Déterminer à quoi sert l’attribut “type” des noeuds des arbres générés par arbogen.
5. Créer un module parseIntoTree.py qui rassemblera les actuels dotParser.py, strParser.ypy, ...

Fait:

* Ajuster computeCoord
  + dans la méthode computeCoord il y a avait self.addOffsets → self.addOffsets
  + dans la méthode addOffsets il y avait tree.offset → self.offset
  + dans la méthode setup il y avait nexts[depth] += 1 → nexts[depth] = self.x +

Père toujours non centré. (cf test.py) Version à jour ?

Mise à jour effectuée :D

* Parser de String terminé (normalement). Il lit le fichier caractère par caractère pour construire récursivement l’arbre représenté. Tous les espaces sont zappés, même ceux des labels (parce que sinon, comment accepter les espaces dans le label mais pas ceux avant ou après ? A moins qu’on fasse un accu d’espaces, et qu’on ne l’ajoute au label que si trouve un caractère != ()espaces … Mais je trouve ça un peu casse couille pour pas grand chose. Qui irait mettre des espaces dans ses labels ?!). testStr montre l’exemple le plus complexe que j’ai testé.
* Parser de XML terminé et fonctionnel. Je me suis basée sur les sources de arbogen pour la structure supposée du fichier xml. J’ai utilisé la librairie ElementTree interne à Python pour parser le fichier XML. Ensuite, on n’a plus qu’à parcourir l’arbre qui en résulte. Le fait qu’on traite le fichier d’un coup peut poser des problèmes de mémoire -> à tester et à débattre avec le prof.
* Ajuster toNetworkX pour que les labels s’affichent mieux. -> Labels placés légèrement au-dessus des noeuds, mais il faut zoomer un peu pour quand même arriver à lire. Peut-être faudrait-il songer à agrandir l’espace entre deux noeuds quand on sait qu’on a des labels à afficher ?
* Nouvelle version de treeDisplay.py qui utilise le module argparse.
  + Permet de spécifier les paramètres et checker leur bon format automatiquement
  + <http://docs.python.org/3.4/howto/argparse.html#id1>