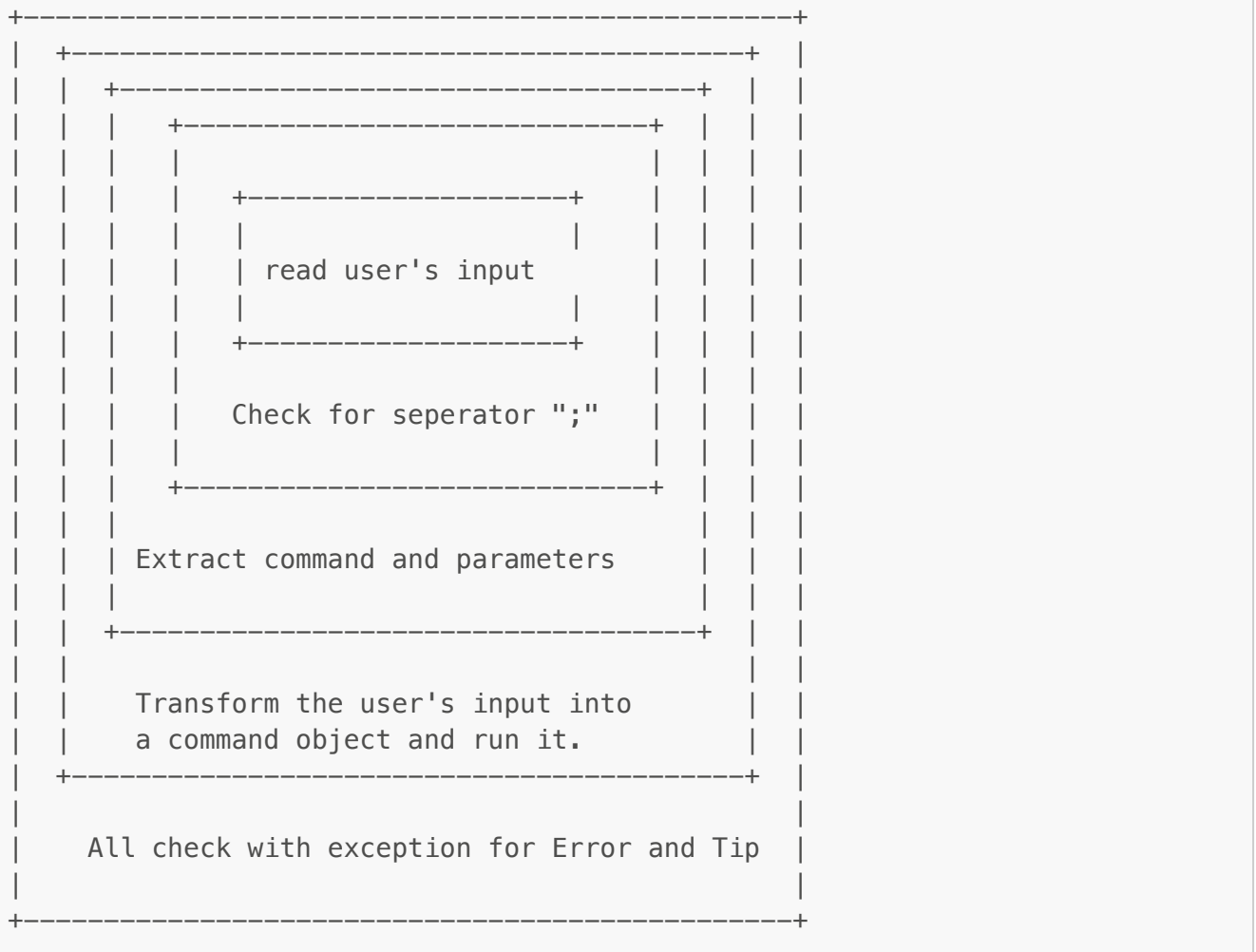
PyShell - TEO KALTRACHIAN

Ce projet a pour but de créer un Shell Python durant le 3ème semestre en ITI à HEPIA. Ce projet a été réalisé entièrement en Python 3. Il fallait reproduire le shell UNIX qui s'affiche sur le terminal pour un ensemble de commandes définis. Les commandes sont séparées par un ";" et elles sont exécutée à la pression de la touche ENTER.

Explication du PyShell

Après avoir entrée la ou les commandes, le user's input va etre parser et puis décomposé en commande et paramètres. On va vérifiée si cette commande ne s'agit pas d'un Alias, si oui l'Alias retournera la commande correspondante. Ensuite, pour un code plus claire et lisible, j'ai décidé d'utiliser un moyen de contourner cette verification enorme de "if - Elif - Else" du "match" entre la commande du user et la fonction qui la défini. J'ai donc crée des classes de commandes qui vont etre crée et appelé grace à leur nom avec le nom de la commande choisi par le user. Celà me permet de faire une seule vérification pour "matcher" la commande désirée avec sa fonctionnalité. De plus, le tout se trouve dans un "try - except" pour gérer toutes les erreurs de toutes les commandes plus facilement. Des "Tip" sont aussi affichés dans le terminal pour aider l'utilisateur lors de fautes de frappes ou pour la syntaxe d'une commande.



Commandes disponibles

Les commandes suivantes:

- pwd: Affiche le dossier courant
- cd PATH: Se déplace dans l'arborescence, de façon relative ou absolue
- mkdir PATH: Crée un dossier
- rm [-r] PATH: Supprime un fichier ou un dossier
- mv SOURCE DEST: Déplacement d'un fichier ou d'un dossier
- ls [-l]: Affiche le contenu du répertoire courant.
- echo PARAM: Affiche le paramètre
- cat FILE: Affiche le contenu d'un fichier
- touch FILE: Crée un fichier vide
- cp [-r] SOURCE DEST: Copie un fichier ou un dossier
- wc [-l -w -c] FILE: Affiche le nombre de ligne (-l), de mot (-w) ou de caractère dans un fichier
- alias WORD=COMMAND: Crée un alias WORD qui exécute la commande COMMAND
- tree PATH: Affiche l'arborescence sous forme d'arbre. Respectez exactement les caractères et les
- find REGEX PATH: Affiche récursivement tous les fichiers et dossiers dont le nom match l'expression
- find man: info sur l'utilisation de find
- grep REGEX FILE: Affiche toutes les ligne qui matche l'expression régulière dans le fichier passé en
- grep man: info sur l'utilisation de grep
- same FILE FILE: Affiche si deux fichiers sont identique
- duplicate PATH: Affiche le path des fichiers identiques dans une arborescence.
- exit: Quitte le Shell et ferme l'application

ainsi que redirection:

- Un fichier de sortie >

Problèmes

L'étape la plus dure était de réfléchir à comment optimiser mon code pour qu'il soit le plus lisible possible entre la gestion des "if - else" et "try - except". de plus, l'utilisation de la commande PIPE n'a pas pu etre réalisée. En effet, j'ai été rattrapé par le temps et la réflection pour optimiser mon code ma fait perde du temps. Cependant, je la ferai par la suite.

Problèmes

En somme, ce projet m'a permis d'apprendre des nouvelles fonctionnalités en Python3 qui ma permiss de coder très librement et de s'implifier et optimiser mon code le plus possible. Le fait d'avoir transformer tout un "if - elif - else" en 5 commandes, le tout dans unn "try - except" m'a fait gagner du temps et plus de lisibilité dans mon code. Ce projet ma aussi permis de comprendre comment fonnctionne les commandes UNIX derière ma console.