Exercice 56 Écrivez une fonction qui crée une archive avec l'ensemble des fichiers ou répertoires passé par le pipe. Faites en sorte que cette fonction soit internationale (français & anglais) et qu'elle soit correctement commentée. Indice : vous pouvez lancer une application externe à PS (comme 7zip s'il est installé sur votre machine) par le biais de l'instruction suivante : `&"C:\Program Files\7-Zip\7z.exe"` ou en utilisant l'instruction `Start-Process "C:\Program Files\7-Zip\7z.exe"`. Comparez les deux commandes.

Réponse :

Exercice 57 Créer une fonction qui compare récursivement le contenu de deux répertoires passés en paramètre et crée un fichier de log en sortie avec le nom des fichiers qui ne sont pas identiques.

Réponse :

Exercice 58 Combiner les fonctions/scripts des Exercice 56 & Exercice 57 pour créer un script qui crée une archive qui contient les fichiers n'étant pas identique d'un répertoire à l'autre.

Réponse :

Exercice 59 Télécharger la librairie iTextSharp (http://sourceforge.net/projects/itextsharp/files/latest/download) et placer le fichier .dll dans votre répertoire de travail. Ensuite, trouver les instructions qui ajoute le type de données .NET contenue dans la librairie et qui instancie un objet de type `pdfreader`. Vous avez reçu le support de cours en fichier PDF modifiable *et* enregistrable. En utilisant les instructions précédentes, écrivez un script qui parcours le fichier PDF afin d'extraire dans un fichier texte toutes les réponses aux questions posées. Indices : Chaque zone de saisie a été nommée `Reponse` suivie du numéro de la question. Il est

conseillé d'utiliser un dictionnaire.

Réponse :

Exercice 60 Maintenant que vous savez faire des scripts assez conséquent, vous souhaitez dialoguer de manière plus professionnelle avec l'utilisateur en affichant des fenêtres popup. Pour cela, il est nécessaire de créer un nouvel objet nommé Wscript.Shell et d'utiliser une de ses méthodes pour afficher la popup. Trouver le code pour afficher un popup de warning.

Réponse :

4.11 Extensions : Snap-ins & Modules

Il existe deux types d'extensions en PS : les modules et les snap-ins.

4.11.1 Les Snap-ins

Le véritable nom est PSSnapin (composants logiciels enfichables) qui ont été créés pour PowerShell v1. Un PSSnapin est constitué généralement d'un ou plusieurs DLL accompagné de fichiers XML qui contiennent les paramètres de configuration et le texte de l'aide. Ces extensions doivent être installés et enregistrés pour que PS sache qu'ils existent.

Exercice 61 Trouver l'instruction qui permet de savoir quels snapins sont disponibles et non installés sur votre machine. Installez les.

Réponse :

4.11.2 Les modules

Les modules n'existent pas dans la v1 de PowerShell. Ils sont conçus pour être auto-suffisant et plus facile à distribuer; mais ils fonctionnent comme les Snapins. Par contre, ils n'ont pas besoin d'être installés. PS les installe automatiquement en utilisant le path `PSModulePath` de la variable d'environnement.

Exercice 62 Écrire l'instruction listant les modules disponibles à partir de la variable d'environnement puis en utilisant une cmdlet spécifique.

Réponse :

Enfin, pour ajouter un module, vous devez l'importer ; ensuite, vous pouvez utiliser la cmdlet Get-Module pour vérifier que le module est chargé.