

ICT122 –8- Scripts avancé

Objectif

Connaitre les structures conditionnelles et les différentes boucles, ainsi que les exceptions

Transposer le diagramme de flux dans un script (et réciproquement).

Remarque

Cet exercice a pour but de vous faire progresser dans la conception de scripts avancés (séquences, structures de contrôles et exceptions).

Pour chacun des exercices, il vous sera demandé de respecter le diagramme de flux, car dans la vie d'entreprise, ce genre de schéma est l'aboutissement d'une réflexion. L'informaticien montre son professionnalisme en effectuant avec soin la tâche demandée et le client ne paie que ce qu'il a validé !

Pour tous les exercices, vous produirez un script PowerShell (*.ps1) compatible avec PowerShell v5.1 et/ou un diagramme de flux dessiné sur papier ou sur <https://draw.io>

Info

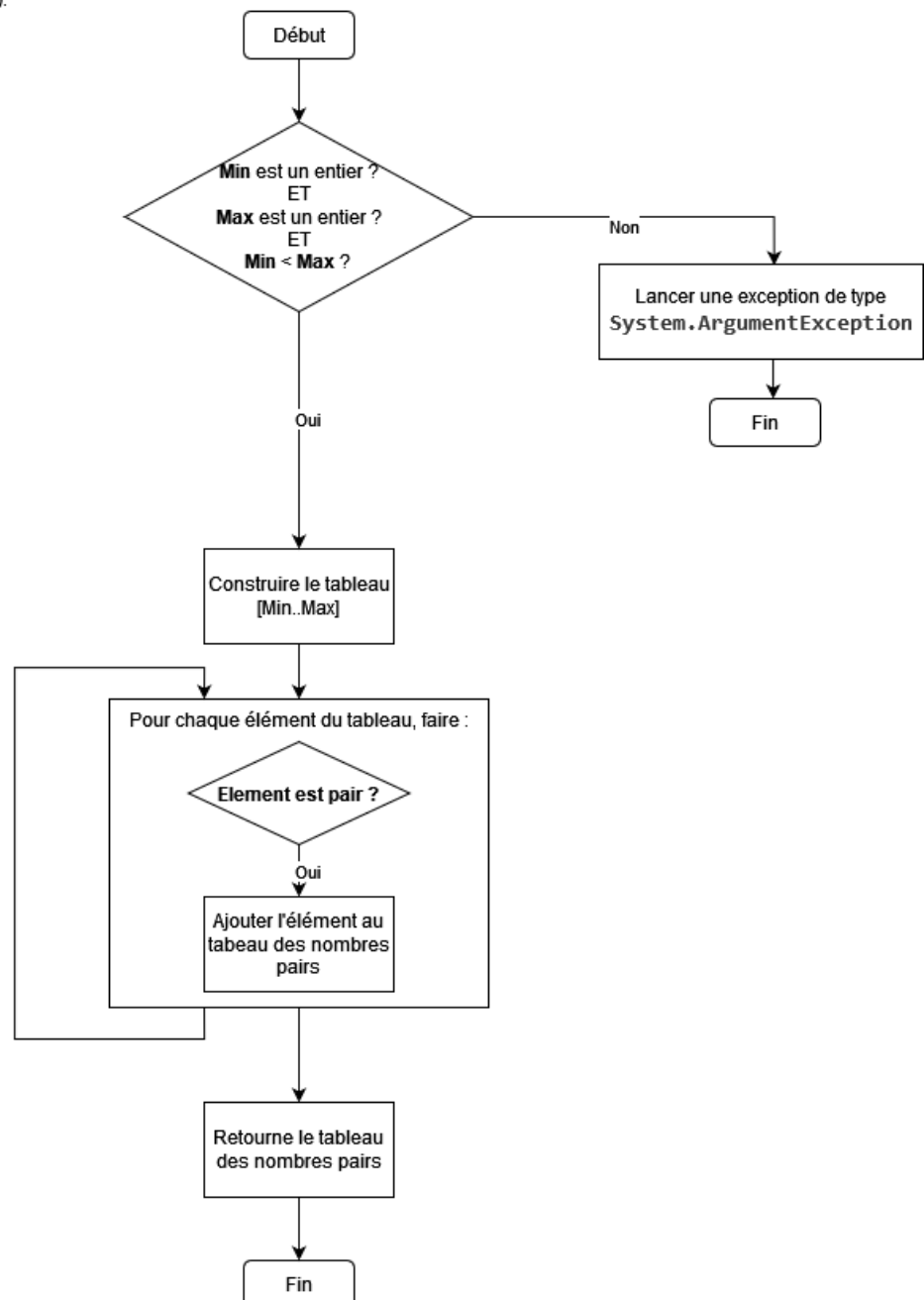
Il est possible de mettre des « .ps1 » sur Teams, mais dans le cas où les fichiers doivent être envoyés par e-mail, il faut ajouter l'extension « .txt », par exemple : « Prenom-Give-Name.ps1.txt »

Exercices

1.- Réalisez le script **Prenom-Write-Numbers.ps1** tel que décrit par le diagramme de flux ci-dessous.

Write-Numbers.ps1

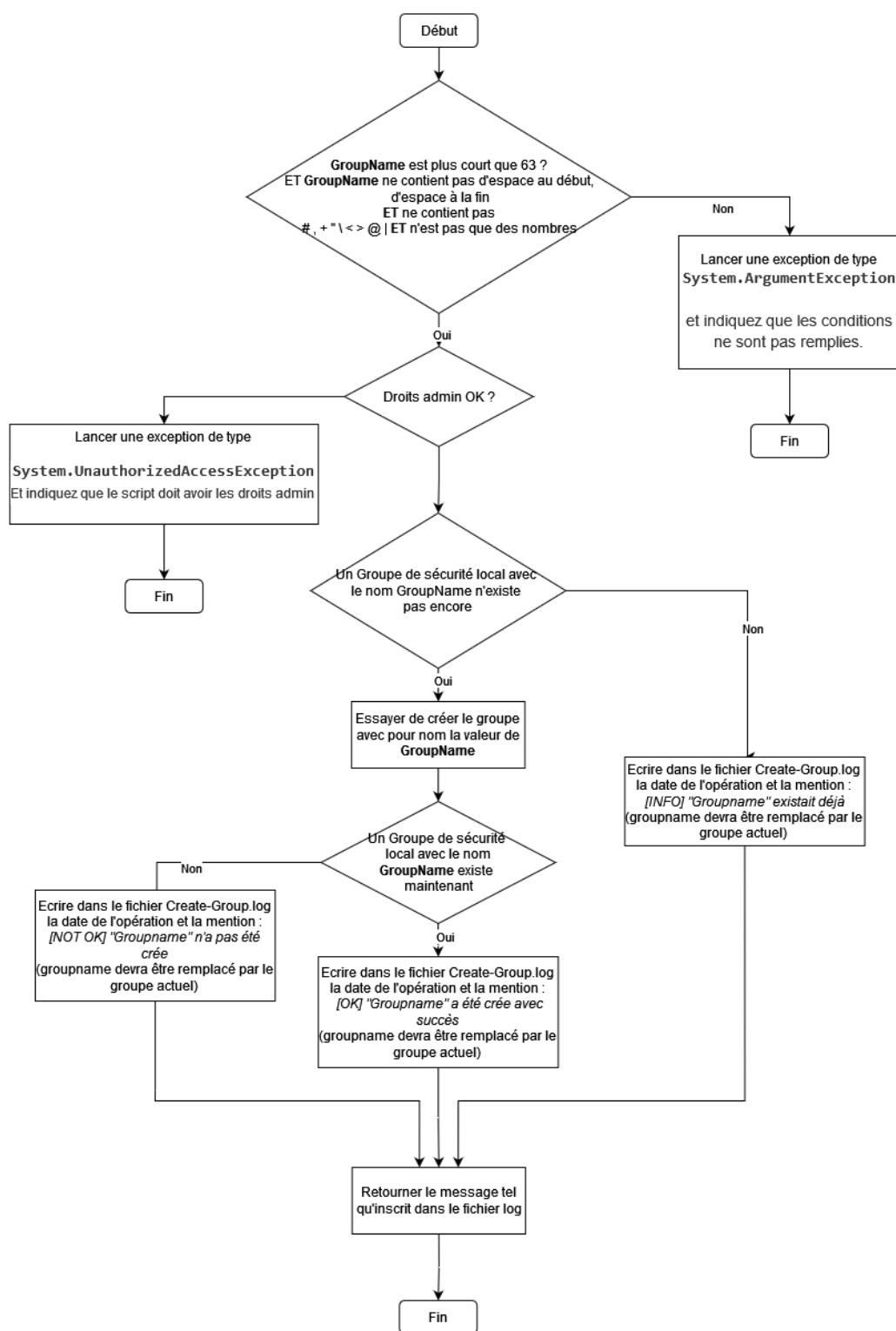
Retourne un tableau de nombre pairs
compris entre deux bornes (min et max).



Version 1 du 21.07.2023

2.- Create-Group.ps1

Réalisez le script **Prenom-Create-Group.ps1** que décrit par le diagramme de flux ci-dessous.



3- Les informaticiens doivent être capable de comprendre le code existant dans une entreprise (en anglais : *legacy code*). Dessinez le diagramme de flux (papier/crayon) ou en utilisant le site <https://draw.io> tel qu'exprimé par le script ci-dessous :

```
[CmdletBinding()]
param (
    [Parameter(Mandatory=$true)]
    [ValidateRange(0, 100)]
    [double]$FreeSpaceThreshold,
    [switch]$IncludeUsbDrives
)

if ($null -eq $FreeSpaceThreshold -or $FreeSpaceThreshold -lt 0 -or $FreeSpaceThreshold -gt 100) {
    throw [System.ArgumentException]::new("La valeur du paramètre 'FreeSpaceThreshold' doit être comprise entre 0 et 100.")
}

$partitions = Get-CimInstance -Class Win32_LogicalDisk

foreach ($partition in $partitions) {
    if (-Not $IncludeUsbDrives -and $partition.Description -match 'USB') {
        continue
    }

    $driveLetter = $partition.DeviceID
    $freeSpaceGB = [math]::Round($partition.FreeSpace / 1GB, 2)
    $totalSpaceGB = [math]::Round($partition.Size / 1GB, 2)
    $freeSpacePercent = [math]::Round(($freeSpaceGB / $totalSpaceGB) * 100, 2)

    if ($freeSpacePercent -ge $FreeSpaceThreshold) {
        Write-Host "Partition : $driveLetter"
        Write-Host "Espace libre : $freeSpaceGB Go"
        Write-Host "Espace total : $totalSpaceGB Go"
        Write-Host "Taux d'espace libre : $freeSpacePercent %"
        Write-Host "-----"
    }
}
```

Bonus :

Modifiez le script pour afficher le message en rouge si l'espace disque restant vaut moins de 20%. En orange si moins de 40%. Sinon en vert.

4.- Get-Mystery2.ps1

Comme pour l'exercice 3, dessinez le diagramme de flux correspondant au code suivant :

```
if (-Not ([Security.Principal.WindowsPrincipal]
[Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Principal.WindowsBuiltI
nRole]::Administrator)) {
    Write-Host "Ce script doit être exécuté avec des droits d'administration."
    Exit 1
}
$NewUserName = Read-Host "Entrez le nom d'utilisateur du nouvel utilisateur :"

if ([string]::IsNullOrEmpty($NewUserName)) {
    Write-Host "Le nom d'utilisateur ne peut pas être vide. Veuillez entrer un nom
d'utilisateur valide."
    Exit 1
}

if (Get-LocalUser -Name $NewUserName -ErrorAction SilentlyContinue) {
    Write-Host "Le nom d'utilisateur '$NewUserName' existe déjà. Veuillez choisir un autre nom
d'utilisateur."
    Exit 1
}

$NewUserPassword = Read-Host "Entrez le mot de passe du nouvel utilisateur :"
$SecureNewUserPassword = ConvertTo-SecureString $NewUserPassword -AsPlainText -Force

try {
    New-LocalUser -Name $NewUserName -Password $SecureNewUserPassword -ErrorAction Stop
}
catch {
    Write-Host "Une erreur s'est produite lors de la création du nouvel utilisateur
'$NewUserName'."
    Write-Host "Erreur : $_"
    Exit 1
}

Write-Host "Le nouvel utilisateur '$NewUserName' a été créé avec succès !"

$GroupName = Read-Host "Entrez le nom du groupe où ajouter l'utilisateur '$NewUserName' :"

if (-Not (Get-LocalGroup -Name $GroupName -ErrorAction SilentlyContinue)) {
    Write-Host "Le groupe '$GroupName' n'existe pas. Veuillez vérifier le nom du groupe et
réessayez."
    Exit 1
}

try {
    Add-LocalGroupMember -Group $GroupName -Member $NewUserName -ErrorAction Stop
}
catch {
    Write-Host "Une erreur s'est produite lors de l'ajout de l'utilisateur '$NewUserName' au
groupe '$GroupName'."
    Write-Host "Erreur : $_"
    Exit 1
}

Write-Host "L'utilisateur '$NewUserName' a été ajouté au groupe '$GroupName' avec succès !"
```

Bonus :

Ce script comporte un risque. Quel est-il ?

Proposer un nouveau diagramme de flux qui limite les interactions avec la console et qui contrôle l'existence de l'utilisateur et du groupe avant la création et l'ajout (comportement atomique du script).