Exercice 1 CYCON Thomas.ps1

```
Set-Location C:\
New-Item kprod -ItemType Directory
New-item -path 'C:\kprod\MAIN','C:\kprod\SECOND' -ItemType Directory
New-item -path 'C:\kprod\MAIN\test.txt' -ItemType File
Move-item -path C:\kprod\MAIN\test.txt -Destination C:\kprod\SECOND
Copy-Item -path C:\kprod\SECOND\test.txt -Destination C:\kprod\MAIN
Remove-Item -path C:\kprod\SECOND -Recurse
Rename-Item -Path C:\kprod\MAIN -NewName C:\kprod\MAIN_new
```

Exercice 2 CYCON Thomas.ps1

```
Set-Location ENV: #Se deplacer dans les envionnements de variable
New-Item "Mewo-promo" -Value "SISR" #Crée une variable Mewo-promo avec comme valeur SISR
Set-Location C: # Se deplacer dans C:
Get-Item -path 'ENV:Mewo-promo' #Mettre en evidence la variable Mewo-promo ainsi que sa valeur
Remove-Item -path ENV:\Mewo-promo #Supprimer la variable
```

Exercice 3 CYCON Thomas.ps1

```powershell
[int]$Res1 = [int]$test1+[int]$test2*[int]$test1
[int]$Res2 = ([int]$test1+[int]$test2)*[int]$test1
Write-Host "L'operation 1 est égale à :" $Res1
Write-Host "L'operation 2 est égale à :" $Res2
```

Exercice 4 CYCON Thomas.ps1

```
"2019: 01\03 - save ok" -match "\d{4}\:\s\d{2}\\d{2}\s\-\s\w{4}\s\w{2,3}"
"2020: 04\01 - save nok" -match "\d{4}\:\s\d{2}\\d{2}\s\-\s\w{4}\s\w{2,3}"
"2019: 01\03 - save ok" -notmatch "\d{4}\:\s\d{2}\\d{2}\s\-\s\w{4}\s\w{2,3}"
"2020: 04\01 - save nok" -notmatch "\d{4}\:\s\d{2}\\d{2}\s\-\s\w{4}\s\w{2,3}"
```

Exercice 5 CYCON Thomas.ps1

```
'8.8.8.8' -match "^([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])(\.([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])){3}$"
'10.1.0.2' -match "^([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])(\.([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])){3}$"
'192.168.2.3' -match "^([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])(\.([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])){3}$"
'A.8.8.8' -match "^([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])(\.([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])){3}$"
'2222.8.8.8' -match "^([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])(\.([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])){3}$"

```

Exercice 6 CYCON Thomas.ps1

```powershell
$Tableau1 = @(1..100)
Write-host "Valeur de l'indice 0 et 1 :" $Tableau1[0,1]
Write-host "Longueur du tableau :" $Tableau1.Length
$Tableau2 = @('Z','Y','X','W','V','U','T','S','R','Q','P','O','N','M','L','K','J','I','H','G','F','E','D','C','B','A')
Write-host "Valeur de l'indice 0 et 1 :" $Tableau2[0,1]
Write-host "Longueur du tableau :" $Tableau2.Length

```

Exercice 7 CYCON Thomas.ps1

```powershell
#Verifier si le dossier Mewo existe sinon le crée puis s'y deplacer
if (Test-Path -Path C:\MEWO) {
    write-host "Le dossier existe deja"}
Else { Write-host "Le dossier va être crée."
    New-item -path C:\MEWO -ItemType Directory
    Write-host "Dossier crée"
    Set-location -Path C:\MEWO\
    }
#Verifier si le fichier test.txt existe sinon le crée
if (Test-Path -Path C:\MEWO\test.txt) {
    write-host "Le dossier existe deja"}
Else { Write-host "Le fichier va être crée."
}

$x = @"
Name : Thomas
Version : 1.0
Description : fichier de test
"@

New-Item -itemtype file -Path "C:\MEWO\test.txt" -value $x

[String]$ReadH = Read-Host "Entrez un nom"
[String]$NameInfo = get-content -path C:\Mewo\test.txt
$ReadH
#Demander de saisir un prénom à l'utilisateur puis le comparer au prénom dans le fichier txt
if (($NameInfo -match "Name : $ReadH") ) {
Write-host "OK"}
Else { Write-host "NOK"}
```

Exercice 8 CYCON Thomas.ps1

```powershell

 $CJ1S = Read-Host "Donne un chiffre entre 1 et 10" -AsSecureString
 $CJ1 = [System.Net.NetworkCredential]::new('', $CJ1S).Password
if ($CJ1 -ge 1 -and $CJ1 -le 10){
    Write-host "Bien"}
 Else { Write-host "Non ce chiffre n'est pas compris entre 1 et 10"}


 # Le joueur 2 doit deviner le chiffre
 [int]$RH2 = Read-Host "Joueur 2, devinez le chiffre choisi par Joueur 1"

 # Vérifier si le joueur 2 à deviné
if ($RH2 -eq $CJ1) {
    Write-Host "Gagné"
} else {
    Write-Host "Perdu"
    if ($RH2 -gt $CJ1) {
        Write-Host "Le chiffre donné par Joueur 2 est plus grand que celui choisi par Joueur 1."
    } else {
        Write-Host "Le chiffre donné par Joueur 2 est plus petit que celui choisi par Joueur 1."
    }
}
```

Exercice 9 CYCON Thomas.ps1

```powershell
#Variable RH pour saisir un chiffre
[int]$RH = Read-Host "Saisir un Chiffre"
#Puis condition pour valider la réponse
if ( $RH -eq 8){
    Write-Host " Le chiffre 8 était le bon chiffre"
    }
Elseif ($RH -eq 12) {
    Write-Host "Le chiffre 12 était le bon chiffre"
    }
Elseif ($RH -eq 18) {
    Write-Host "Le chiffre 18 était le bon chiffre"
    }
Else {
    Write-Host "Le chiffre $RH n'est pas le bon"
    }
```

Exercice 10 CYCON Thomas.ps1

```powershell
[int]$test = 0
while ($test -le 10) {
    Write-host $test
    $test++
}
```

Exercice 11 CYCON Thomas.ps1

```powershell
[int]$test = 0
Do {
    $test++
    Write-host $test
}
While ($test -ne 10)
```

Exercice 12 CYCON Thomas.ps1

```powershell
#Creation de la variable
[int]$test = 0

# Nom de l'application
$NA = "wireguard"

# Boucle pour l'application en cours d'exécution
do {
    # Ajouter 1 à la variable
    $test++

    # Vérifier si le processus de l'application est en cours d'exécution
    $AC = Get-Process -Name $NA -ErrorAction SilentlyContinue
} while ($AC)

# Afficher le message final lorsque l'application est fermée
Write-Host "L'application '$NA' est fermée. Valeur finale de compteur : $test"
```

Exercice 13 CYCON Thomas.ps1

```powershell
do {
    # Saisir un chiffre entre 1 et 10
    [Int]$test = Read-Host "Entrer un chiffre entre 1 et 10"

    # Vérifier si le chiffre est entre 1 et 10
    if ($test -ge 1 -and $test -le 10) {
        Write-Host "Le chiffre est correct"
    } else {
        Write-Host "Le chiffre '$test' n'est pas valide."
    }
} until ($test -ge 1 -and $test -le 10)
```

Exercice 14 CYCON Thomas.ps1

```powershell
#Création de la variable
$test = 0
#Incrementation
for ($test = 1; $test -le 10; $test++) {
    Write-Output $test
}
```

Exercice 15 CYCON Thomas.ps1

```powershell
# $disques pour obtenir les infos des disques
$disques = Get-WmiObject -Class Win32_LogicalDisk

$espaceLibreTotal = 0

# foreach pour parcourir les disques
foreach ($disque in $disques) {

    # Récupérer l'espace libre en octets
    $espaceLibre = $disque.FreeSpace

    # Convertir l'espace libre en (gigaoctets)
    $espaceLibreGB = [math]::Round($espaceLibre / 1GB, 2)

    # Afficher les informations du disque et de l'espace libre
    Write-Host "Disque : $($disque.DeviceId)"
    Write-Host "Espace libre : $espaceLibreGB Go"

      # Ajout de l'espace libre à la variable totale
      $espaceLibreTotal += $disque.FreeSpace

}

# Convertir l'espace libre en format plus lisible (gigaoctets)
$espaceLibreTotalGB = [math]::Round($espaceLibreTotal / 1GB, 2)

# Affichage de l'espace libre total
Write-Host "Espace libre total : $espaceLibreTotalGB Go"
```

Exercice 16 CYCON Thomas.ps1

```powershell
#Game Start
#Commencer
[string]$Start = Read-host = "Voulez-vous jouer (oui ou non) ?"
New-item -path "C:\MEWO\" -ItemType Directory
$Tablscore = @()

while ($start -eq "oui"){

# Définir le chemin du fichier texte
$fichier = "C:\MEWO\tableau_scores.txt"

# Lire le tableau de scores depuis le fichier texte
$scores = Get-Content -Path $fichier

# Afficher le tableau de scores
$scores | Format-Table -AutoSize

#Pour réinitialiser le tableau des scores
#$Tablscore = @()

#Entrez un pseudo
[string]$pseudo = Read-host "Entrez votre Pseudo."

#Choix du sujet
[int]$sujet = Read-Host "Choix des sujets (noter le chiffre) : 1 L'Empire Romain, 2 L'Egypte Antique, 3 Les Mayas, 4 Les découvertes scientifiques, 5 Les Dieux de toutes les Mythologies"

#Question
$questions1 = @(...)

$questions2 = @(...)

$questions3 = @(...)

$questions4 = @(...)

$questions5 = @(...)

#Choix de la difficulté
[int]$Difficult = Read-Host "Combien de questions voulez-vous (3, 5, ou 10)"

# Sélectionner des questions aléatoires
If($sujet -eq 1){
    $QuestionsSelectionne = $questions1 | Get-Random -Count $difficult
}
Elseif($sujet -eq 2){
    $QuestionsSelectionne = $questions2 | Get-Random -Count $difficult
}
Elseif($sujet -eq 3){
    $QuestionsSelectionne = $questions3 | Get-Random -Count $difficult
}
Elseif($sujet -eq 4){
    $QuestionsSelectionne = $questions4 | Get-Random -Count $difficult
}
Elseif($sujet -eq 5){
    $QuestionsSelectionne = $questions5 | Get-Random -Count $difficult
}
Else{
    write-host = "Choix non valide"
    }
```

Exercice 16 CYCON Thomas.ps1

```
#Jeu

# Initialiser le score
$score = 0

# Boucle pour les questions
foreach ($question in $QuestionsSelectionne) {
    $chance = 0
    $correct = $false

    while ($chance -lt 3 -and -not $correct) {
        $reponse = Read-Host $question.Question
        if ($reponse -eq $question.rep) {
            Write-Host "Correct!"
            $score++
            $correct = $true
        } else {
            $chance++
            if ($chance -lt 3) {
                Write-Host "Incorrect. Essayez encore. Il vous reste $((3 - $chance)) tentative(s)."
            } else {
                Write-Host "Incorrect. La bonne réponse est $($question.rep)."
            }
        }
    }
}

#Sujet pour le tableau des scores
if($sujet -eq 1){
$nsujet = "L'Empire Romain"
}
Elseif($sujet -eq 2){
$nsujet = "L'Egypte Antique"
}
Elseif($sujet -eq 3){
$nsujet = "Les Mayas"
}
Elseif($sujet -eq 4){
$nsujet = "Les découvertes scientifiques"
}
Else{$nsujet = "Les Dieux de toutes les Mythologies"
}

# Ajouter le score au tableau des scores
$Tablscore += [PSCustomObject]@{
    Pseudo = $pseudo
    Sujet = $nsujet
    Difficultés = $Difficult
    Score  =$score
}

# Définir le chemin du fichier texte
$fichier = "C:\MEMO\tableau_scores.txt"

# Enregistrer le tableau de scores dans le fichier texte
$Tablscore | Out-File -FilePath $fichier

# Afficher le score final
Write-Host "$pseudo, votre score final est $score sur $difficult."

# Lire le tableau de scores depuis le fichier texte
$scores = Get-Content -Path $fichier

# Afficher le tableau de scores
$scores | Format-Table -AutoSize

# Afficher un message de confirmation
Write-Output "Le tableau de scores a été enregistré dans $fichier"
#Recommencer
$Start = Read-host = "Voulez-vous recommencer ?"
}

##Fin
```

script DNS CYCON Thomas.ps1

```powershell
$start = Read-Host "Lancer le script ? (oui ou non)"

while($start -eq "oui"){

[int]$opt = Read-host "`n`[1] Récupérer le/les serveur(s) DNS de l'ordinateur`n`[2] Changement du serveur DNS de l'ordinateur parmis une liste définit`n`[3] Création d'un enregistrement DNS (option proposée pour le type)`n`[4] Test d'un enregistrement DNS (Ip vers nom/ nom vers IP)`n`[5] Suppression d'un enregistrement`n`[6] Visualisation des dernières erreurs DNS`n`Choisir une option (noter le chiffre)"
$dns = Get-DnsClientServerAddress -AddressFamily IPV4 |Select-Object -ExpandProperty ServerAddresses


    If($opt -eq 1){

     write-host $dns ; ping $dns

    }

    Elseif($opt -eq 2){

     $dnssw = read-host -Prompt "Quelle DNS choisissez vous (1.1.1.1, 8.8.8.8, 10.1.1.1)"

        switch ($dnssw){

         "1.1.1.1" {Set-DnsClientServerAddress -InterfaceAlias Ethernet0 -ServerAddresses ("1.1.1.1")}
         "8.8.8.8" {Set-DnsClientServerAddress -InterfaceAlias Ethernet0 -ServerAddresses ("8.8.8.8")}
         "10.1.1.1"{Set-DnsClientServerAddress -InterfaceAlias Ethernet0 -ServerAddresses ("10.1.1.1")}
         }
    }

    Elseif($opt -eq 3){
        $dnsRR = read-host -Prompt "Quelle type d'enregistrement choisissez-vous [A] ou [Cname] ?"
     switch ($dnsRR){
         "A" {invoke-command -Computername DC1 -Scriptblock {Add-DnsServerResourceRecordA}}
         "Cname" {invoke-command -Computername DC1 -Scriptblock {Add-DnsServerResourceRecordCname -HostNameAlias "learn.local" -ZoneName "learn.local"}}
         }
    }

    Elseif($opt -eq 4){
    $reponse = read-host = "choisissez un DNS"
    Resolve-DnsName -Name $reponse
    }

    Elseif($opt -eq 5){
        invoke-command -Computername DC1 -Scriptblock {Get-DnsServerResourceRecord -ZoneName "learn.local"}

        invoke-command -Computername DC1 -Scriptblock {
        [String]$nname = read-host "Choisir le nom de l'enregistrement à supprimer"
        [String]$type = Read-Host "Choisir le type de l'enregistrement"
        Remove-DnsServerResourceRecord -ZoneName "learn.local" -RRType $type -Name $nname}
    }
    Elseif($opt -eq 6){

    invoke-command -Computername DC1 -Scriptblock {Get-WinEvent -FilterHashTable @{LogName='DNS Server';Level='2'}}

    }
}
```

Script AD CYCON Thomas.ps1*

```powershell

$start = Read-Host = "Lancer le script ? (oui ou non)"

while($start -eq "oui"){
    [Int]$ChoixG = Read-Host "[1]Créer un utilisateur, une OU ou un groupe`n`[2]Désactiver un utilisateur`n`[3]Modifier un utilisateur`n`[4]Récupérer la liste des utilisateurs d'un OU
     ou d'un groupe`n`[5]Récupérer les informations d'un user`n`Que choisissez-vous ?"


    If($choixG -eq 1){
       [Int]$choix = Read-Host "[1]Un utilisateur`n`[2]Une OU`n`[3]Un groupe`n`Que voulez-vous créer"

        If($choix -eq 1){

        Write-Host "# Veuillez renseigner les différents champs ci-dessous" -ForegroundColor Cyan
        invoke-command -Computername DC1 -Scriptblock {
        $prenom = read-host "Prénom"
        $nom= Read-Host "Nom"
        $path = Read-Host -Prompt "DC1/Lyon`n`DC1/Marseille`n`Quelle OU choisissez-vous "
        $psw = Read-Host -Prompt "L'utilisateur change son mot de passe à la connexion (oui ou non)"
        Switch ($psw){
         "oui" {$psw = 1}
         "non" {$psw = 0}
         }
        New-ADUser -Name "$prenom $nom" -DisplayName "$nom $prenom" -GivenName "$prenom" -Surname "$nom" -SamAccountName "$prenom.$nom" -UserPrincipalName "$prenom.$nom@learn.local"
        -Path "OU=$path,OU=Domain Controllers,DC=LEARN,DC=LOCAL" -AccountPassword(Read-Host -AsSecureString "Mot de passe ?") -ChangePasswordAtLogon $psw -Enabled 1}

        Write-Host "# Formulaire complété, le compte va être créé dans l'Active Directory..." -ForegroundColor Green
        }


        If($choix -eq 2){
        Write-Host "# Veuillez renseigner les différents champs ci-dessous" -ForegroundColor Cyan
        invoke-command -Computername DC1 -Scriptblock {
        $nomOU = Read-Host -Prompt "Nom de l'OU"
        New-ADOrganizationalUnit -Name $nomOU -Path "OU=Domain Controllers,DC=LEARN,DC=LOCAL"}
        Write-Host "# Formulaire complété, le compte va être créé dans l'Active Directory..." -ForegroundColor Green
        }

        If($choix -eq 3){
        Write-Host "# Veuillez renseigner les différents champs ci-dessous" -ForegroundColor Cyan
        invoke-command -Computername DC1 -Scriptblock {
        $namegrp = read-host "Nom du groupe"
        $pathgrp = Read-Host -Prompt "DC1/Lyon`n`DC1/Marseille`n`Quelle OU choisissez-vous "
        [String]$description = Read-Host "Donner une description"

        New-ADGroup -Name $namegrp -Path "OU=$pathgrp,OU=Domain Controllers,DC=LEARN,DC=LOCAL" -GroupScope Global -Description $description}

        Write-Host "# Formulaire complété, le compte va être créé dans l'Active Directory..." -ForegroundColor Green
        }
```

Script AD CYCON Thomas.ps1*

```
    Write-Host "# Veuillez renseigner les différents champs ci-dessous" -ForegroundColor Cyan
    invoke-command -Computername DC1 -Scriptblock {
    $SamAccount = Read-Host "Donnez le nom de l'utilisateur (prénom.nom)"
    Set-ADUser -Identity $SamAccount -Enabled:$false}
    Write-Host "# Formulaire complété, le compte va être désactivé dans l'Active Directory..." -ForegroundColor Green
    }

    If($choixG -eq 3){
        Write-Host "# Veuillez renseigner les différents champs ci-dessous" -ForegroundColor Cyan
        invoke-command -Computername DC1 -Scriptblock {
        $ChoixC = Read-Host "[1]Reset du Mot de Passe`n`[2]Changer le chemin`n`Que voulez-vous faire"
        $PathC = Read-Host "Quelle chemin choisir DC1/Lyon ou DC1/Marseille"
        $SamAccount = Read-Host "Quelle utilisateur changer (prénom.nom)"
        $user = Get-ADUser -Identity $SamAccount -ErrorAction SilentlyContinue
        If($ChoixC -eq 1){
        Set-AdUser -Identity $SamAccount -ChangePasswordAtLogon 1 -Enabled 1}
        If($ChoixC -eq 2){
        Move-AdObject -Identity $user.DistinguishedName -TargetPath "OU=$pathC,OU=Domain Controllers,DC=LEARN,DC=LOCAL"
        }
         Write-Host "# Formulaire complété, le compte va être modifié dans l'Active Directory..." -ForegroundColor Green
        }
     }

     If($ChoixG -eq 4){
        invoke-command -Computername DC1 -Scriptblock {
            $ch = Read-Host "[1]Liste des Utilisateurs d'un Groupes`n`[2]Liste des Utilisateurs d'une OU"
                switch ($ch){
                    "1" {
                            $ndgrp = Read-Host "Nom du groupe"
                            $lsgrp = Get-ADGroupMember -Identity $ndgrp
                               Write-Host $lsgrp.name
                        }
                    "2" {
                            $pathF = Read-Host "Choisir une OU DC1/Lyon ou DC1/Marseille"
                            $nam = Get-ADUser -Filter * -SearchBase "OU=$pathF,OU=Domain Controllers,DC=LEARN,DC=LOCAL"
                            Write-Host $nam.name
                        }
            }
        }
     }

     If($ChoixG -eq 5){
        invoke-command -Computername DC1 -Scriptblock {
            $namG = Read-Host "Choisir un utilisateur (prénom.nom)"
            Get-ADUser -Identity $namG
         }
     }
}

```