R405 TP1 - Eliott PARADIS

🖱️ Interaction Utilisateur

Exercice 1

Script :

```powershell
$nom = Read-Host "Quel est votre nom ?"
$naissance = Read-Host "Bonjour $nom, quelle est votre année de naissance ?"
$age = Read-Host "Quel sera votre âge lors anniversaire cette année ?"
$annee = [int]$naissance + [int]$age
Write-Host "Nous sommes en $annee"
```

Explication :

- Utilise `Read-Host` pour saisir nom, année de naissance et âge.
- Calcule l'année courante avec une addition simple.

📊 Manipulation d'objets (Chaînes)

Exercice 2

Script :

```powershell
$texte = "Coder en Powershell c'est simple"
Write-Host "Longueur : $($texte.Length)"
Write-Host "Caractères : $($texte.ToCharArray())"
Write-Host "Premier 'P' : $($texte.IndexOf('P'))"
Write-Host "Remplacement e→i : $($texte.Replace('e','i'))"
Write-Host "Dernier 'p' : $($texte.LastIndexOf('p'))"
Write-Host "Découpage : $($texte.Split(' '))"
Write-Host "Extrait (2,5) : $($texte.Substring(2,5))"
Write-Host "---"
```

Explication :

- Exploite les méthodes `Length`, `Replace`, `Split`, etc. de l'objet `System.String`.

🕰️ Manipulation de dates

Exercice 3

Script :

```
$date = Get-Date
Write-Host "Date : $($date.ToString('dd/MM/yyyy'))"
Write-Host "Heure : $($date.ToString('HH:mm:ss'))"
Write-Host "Dans 5 jours : $($date.AddDays(5).ToString('dddd dd MMMM yyyy'))"
Write-Host "Dans 1 mois : $($date.AddMonths(1).ToString('dddd dd MMMM yyyy'))"
Write-Host "Il y a 10 jours : $($date.AddDays(-10).ToString('dd/MM/yyyy'))"
```

Explication :

- Utilise `AddDays`, `AddMonths` et le formatage personnalisé (`dddd` pour le jour en entier).

Exercice 4

Script :

```
$date = Get-Date
$culture = [System.Globalization.CultureInfo]::GetCultureInfo("fr-FR")
Write-Host "Jour : $($date.ToString('dddd', $culture))"
Write-Host "Abrégé : $($date.ToString('ddd', $culture))"
Write-Host "Mois : $($date.ToString('MMMM', $culture))"
Write-Host "Abrégé : $($date.ToString('MMM', $culture))"
```

Explication :

- Formatage localisé en français avec `ToString()` et `CultureInfo`.

🔁 Conversion de données

Exercice 5

Script :

```
$input = Read-Host "Entrez une date"
$date = [System.DateTime]::Parse($input)
Write-Host "Date convertie : $date"
```

Explication :

- Conversion d'une chaîne en `DateTime` avec la méthode statique `Parse()`.

Exercice 6

Script :

```
try {
    $input = Read-Host "Entrez une date"
    $date = [System.DateTime]::Parse($input)
    Write-Host "Date valide : $date"
} catch {
    Write-Host "Erreur : Formats acceptés -> '2/5/2024', '3 mars 2024'"
}
```

Explication :

- Gestion d'erreur avec `try/catch` pour guider l'utilisateur.

🖥️ Gestion des processus

Exercice 7

Script :

```
Get-Process | Select-Object ProcessName, StartTime, CPU
```

Explication :

- Affiche le nom, l'heure de démarrage et l'utilisation CPU des processus.

Exercice 8

Script :

```
Get-Process | Where-Object { $_.CPU -gt 10 }
```

Réponse à la question PDF :

- Filtre les processus avec `Where-Object` pour afficher ceux utilisant >10s CPU.

Exercice 9

Script :

```
Get-Process | Sort-Object CPU -Descending
```

Explication :

- Trie les processus du plus gourmand au moins gourmand en CPU.

Exercice 10

Script :

```
Get-Process | Sort-Object CPU -Descending | Select-Object -First 10
```

Explication :

- Affiche les 10 premiers processus après tri.

Exercice 11

Script :

```
$process = Get-Process | Sort-Object CPU -Descending | Select-Object -First 10
$process | Out-File -FilePath "process.txt"
```

Explication :

- Sauvegarde le résultat dans `process.txt` avec `Out-File`.

🌐 Réseau

Exercice 12

Script :

```
Get-NetTCPConnection | Where-Object { $_.State -eq "Listen" }
```

Réponse à la question PDF :

- La propriété `OwningProcess` permet de retrouver le processus associé à un port.

---**Réponse à la question PDF :**

Exercice 13

Script :

```

```

```
Get-Process -IncludeUserName | Where UserName -NE
$([System.Security.Principal.WindowsIdentity]::GetCurrent().Name) # Méthode la
plus efficace
#
Get-Process -IncludeUserName | Where UserName -NE $(whoami) # Meme résultat (et
même type) mais appel d'un autre binaire
```

Explication :

- Utilise `-IncludeUserName` pour afficher les processus non lancés par l'utilisateur courant.
- Utilisation de `[System.Security.Principal.WindowsIdentity]::GetCurrent().Name`
    - *Contrairement à $env:USERNAME (variable modifiable), cette méthode .NET récupère l'identité Windows réelle de l'utilisateur qui à executé le script.*