NOTES DE COURS

Les variables

Automne 2019

Table des matières

1 Les variables

1.1 Créer une nouvelle variable

La commande New-Variable permet de créer une nouvelle variable. Si la variable existe déjà alors un message d'erreur est affiché.

```
New-Variable -Name prix
             -Value 40
             -Description 'prix du produit'
```

Il est aussi possible de créer une variable directement :

```
$z=30
```

Dans ce cas, il faut préfixer la variable par le caractère "**$**".

Si la variable contient des caractères spéciaux alors il faut placer le nom entre accolades : **${ma variable}**

Si la variable contient des accolades, il faut les placer entre accolades et précéder les accolades faisant partie du nom d'un accent grave :

```
${ma`{variable`}2}=23
```

Pour afficher la valeur de la variable :

```
${ma`{variable`}2}
23
```

1.2 Assigner une valeur à une variable

La commande Set-Variable permet d'assigner une valeur à une variable. Si la variable n'existe pas alors elle sera créée.

```
Set-Variable -Name prix
             -Value 1000
```

Il est aussi possible d'assigner une valeur à une variable en utilisant l'opérateur =

```
$prix=4000
```

La valeur assignée peut être :

- Numérique

```
PS C:\> $total=1020
```

```
PS C:\> $x -is [int]
True
```

```
PS C:\> $x -is [string]
False
```

Il est possible de définir de manière explicite le type de données d'une variable :

```
PS C:\> [int]$total=1020
```

- Chaîne de caractères

```
PS C:\> $cours="Power Shell"
```

```
PS C:\> $cours -is [int]
False
```

```
PS C:\> $cours -is [string]
True
```

- Booléenne

```
PS C:\> $recherche=$true

PS C:\> $recherche -is [int]
False

PS C:\> $recherche -is [string]
False

PS C:\> $recherche -is [boolean]
True
```

Pour les valeurs booléennes, on utilise les deux variables système **$true** et **$false**.

- Tableau (liste de valeurs séparées par des virgules)

```
PS C:\> $tableau=34,"java",$true
```

Un tableau peut contenir des valeurs de différents types.

Pour afficher la liste des valeurs :

```
PS C:\> $tableau
34
java
True
```

Pour afficher le premier élément du tableau, on utilise l'index 0 :

```
PS C:\> $tableau[0]
34
```

Pour afficher le deuxième élément du tableau, on utilise l'index 1 :

```
PS C:\> $tableau[1]
java
```

Pour afficher le troisième élément du tableau, on utilise l'index 2 :

```
PS C:\> $tableau[2]
True
```

Il est possible de modifier un élément du tableau :

```
PS C:\> $tableau[2]=1000
```

```
PS C:\> $tableau[2]
1000
```

Déclaration d'un tableau vide

```
PS C:\>  $tableau = @()
```

- Hashtable (tableau associatif de clé/valeur)

  Un tableau associatif doit être déclaré explicitement :

```
$a=@{}
```

```
$a=@{"cle1"="val1";"cle2"="val2"}
```

  Pour accéder aux valeurs on utilise une notation similaire aux tableaux ordinaires, mais avec une chaîne de caractères au lieu d'un index:

```
$a["cle1"]
```

  On peut aussi utiliser deux autres notations pour accéder aux valeurs:

```
$a.item("cle1") $a.cle1
```

  EXEMPLE

```
$capitale=@{"France"="Paris";"Mali"="Bamako"}

$capitale["Mali"]

$capitale.Values
```

Types de données

PowerShell support les variables du .Net

Type	Description
[string]	Chaîne de caractères Unicode
[char]	Caractère Unicode
[byte]	Octet
[int]	Nombre entier signé 32-bit
[long]	Nombre entier signé 64-bit
[bool]	Booléen
[decimal]	Valeur décimale sur 128-bit
[single]	Nombre à virgule flottante 32-bit
[double]	Nombre à virgule flottante 64-bit
[DateTime]	Date
[xml]	XML
[array]	Tableau
[hashtable]	Tableau associatif

Conversion de la valeur de retour

```
PS C:\> [int]$total=2000
```

```
PS C:\> [datetime]$total

January 1, 0001 12:00:00 AM
```

Incrémenter une variable (++)

```
PS C:\> $i=1
```

```
PS C:\> $i=$i+1
```

```
PS C:\> $i
2
```

```
PS C:\> $i++
```

```
PS C:\> $i
3
```

Décrémenter une variable (--)

```
PS C:\> $j=5
```

```
PS C:\> $j=$j-1
```

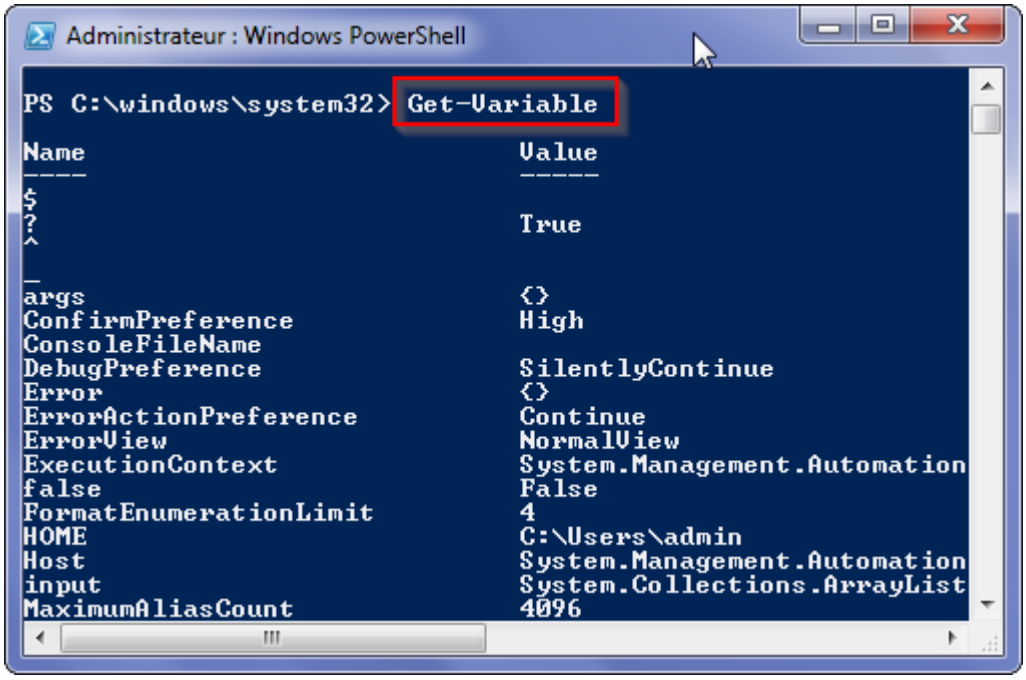
```
PS C:\> $j
4
```

```
PS C:\> $j--
```

```
PS C:\> $j
3
```

1.3 Afficher les propriétés des variables

La commande Get-Variable permet d'afficher les propriétés des variables :



Get-Variable suivi du nom d'une variable (sans le $) permet de recevoir un objet englobant la variable.

```
Get-Variable -Name prix

Name                           Value
----                           -----
prix                           4000
```

Déterminer le type d'une variable

La fonction GetType retourne le type de données d'une variable.

```
New-Variable -Name salaire -Value 2000
```

Afficher le type de données en utilisant la méthode GetType :

```
$salaire.GetType()

IsPublic IsSerial Name     BaseType
-------- -------- ----     --------
True     True     Int32    System.ValueType
```

```
$salaire.GetType().Name

Int32
```

1.4 Supprimer la valeur d'une variable

La commande Clear-Variable permet de supprimer la valeur d'une variable sans supprimer la variable.

```
Clear-Variable -Name prix
```

Afficher la valeur de la variable prix :

```
Get-Variable -Name prix

Name                           Value
----                           -----
prix
```

1.5 Supprimer une variable

La commande Remove-Variable permet de supprimer une variable.

```
Remove-Variable -Name prix
```

Vérifier si la variable existe :

```
Get-Variable -Name prix


Get-Variable : Cannot find a variable with the name 'prix'.
At line:1 char:1
+ Get-Variable -Name prix
+ ~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (prix:String) [Get-
Variable], ItemNotFoundException
    + FullyQualifiedErrorId :
VariableNotFound,Microsoft.PowerShell.Commands.GetVariableCommand
```