Les fonctions

Contenu

- **Retour sur les commandes en PowerShell**
- **Les fonctions en mathématiques**
- **Les fonctions en programmation**
- **Les fonctions en PowerShell**
- **Fonctions vs Cmdlets**

Retour sur les commandes en PowerShell

Retour sur les commandes en PowerShell

- **Syntaxe d'une commande PowerShell:**
  - Verbe-Nom (ex: Get-Help)
  - Le mot avant un « - » s'appelle toujours un verbe, même lorsqu'il n'en est pas techniquement un

Les arguments

- **Une commande peut aussi prendre des arguments**
- **Exemples d'arguments:**
  - Set-Location C:\
  - New-Item **-Path** . **-Name** "PowerShell" **-ItemType** "directory"

Cmdlet

- **Une commande PowerShell est aussi appelée « cmdlet » (prononcé « *commandlet* »)**
  - Et ce, même si la commande est belle et qu'elle n'est donc pas une *commande laite*

https://knowyourmeme.com/memes/comedy-genius

Les fonctions en mathématiques

Les fonctions en mathématiques

- **En mathématiques, une fonction associe une valeur (x) à une autre (y)**

Exemple de fonction en mathématiques

- **y = f(x)**
- **f(x) = 2x + 7**

- **Pour x = 42:**
  - On remplace x par 42 dans l'expression « 2x + 7 »
  - y = f(42)
  - = 2(42) + 7
  - = 91

- **Cette fonction associe donc la valeur 42 à la valeur 91**

Les fonctions en mathématiques

- **x est une variable**
- **En donnant une valeur à cette variable, on peut résoudre l’expression mathématique qui détermine la valeur de f(x)**

Les fonctions en programmation

Les fonctions en programmation

- **Une fonction en programmation ressemble à une fonction en mathématiques**
  - Fonction en mathématiques: f(x) = expression mathématique utilisant x
  - Fonction en programmation: f(x) -> bloc de code utilisant x

Pseudocode d’une fonction

FONCTION NomDeLaFonction (paramètre 1, paramètre 2, ...)

DÉBUT

    Bloc de code utilisant les paramètres

FIN

Appel de fonction

- **Pour exécuter une fonction, on doit l'appeler en lui passant des arguments**
  - maFonction(argument1, argument2, …)
- **Le corps de la fonction (bloc de code) est alors exécuté, en remplaçant les paramètres par les arguments correspondants**

Exemple

Déclaration d'une fonction Factorielle:

```
FONCTION Factorielle (nombre)
DÉBUT
    resultat <- 1
    POUR i de 1 à nombre
    DÉBUT
        resultat <- resultat * i
    FIN
    RETOURNER resultat
FIN
```

Exemple

- **Appel de la fonction Factorielle avec l'argument 7:**

```
factorielleDeSept <- Factorielle(7)
```

- **Le résultat de la fonction (valeur de retour) est assigné à la variable « factorielleDeSept »**

À quoi ça sert?

- **Une fonction a deux utilités principales:**
  - Éviter les répétitions de code
  - Rendre le code plus facile à lire

Éviter les répétitions de code

- **Principe *DRY* (*Don’t Repeat Yourself*)**
- **Si un bloc de code identique se retrouve à plusieurs endroits dans notre programme, on en fait une fonction**

Exemple

```
Écrire "Entrer votre prénom"
prenom <- Lire au clavier
Écrire "Votre prénom est " + prenom
Écrire "Entrer votre nom"
nom <- Lire au clavier
Écrire "Votre nom est " + nom
Écrire "Vous vous appelez " + prenom + " " + nom
```

Exemple

```
FONCTION demanderInfo(descriptionInfo)
DÉBUT
    Écrire "Entrer votre " + descriptionInfo
    info <- Lire au clavier
    Écrire "Votre " + descriptionInfo + " est " + info
    RETOURNER info
FIN

prenom <- demanderInfo("prénom")
nom <- demanderInfo("nom")
Écrire "Vous vous appelez " + prenom + " " + nom
```

Rendre le code plus facile à lire

- **En donnant des noms significatifs à nos fonctions, on donne plus d’informations sur ce que le code fait**

Exemple

```
# nombres: tableau de nombres entiers déjà initialisé


somme <- 0
POUR CHAQUE nombre DANS nombres
DÉBUT
    somme <- somme + nombre
FIN
ÉCRIRE "Moyenne: " + somme / taille du tableau nombres
```

Exemple

```
FONCTION calculerMoyenne(tableau)
DÉBUT
    somme <- 0
    POUR CHAQUE nombre DANS nombres
    DÉBUT
        somme <- somme + nombre
    FIN
    moyenne <- somme / taille de tableau
    RETOURNER moyenne
FIN
```

Exemple

 # nombres: tableau de nombres entiers déjà initialisé

Écrire "Moyenne: " + calculerMoyenne(nombres)

Écrire des fonctions courtes

- **Idéalement, une fonction devrait:**
  - Être courte
  - Faire une seule chose
    - Si une fonction fait plusieurs choses, elle devrait faire appel à d'autres fonctions
- **Principe *KISS* (Keep it simple, stupid)**

Exemple

```
# Cette fonction calcule la somme des nombres d'un tableau,
# puis calcule une moyenne à partir de cette somme
FONCTION calculerMoyenne(tableau)
DÉBUT
    somme <- 0
    POUR CHAQUE nombre DANS nombres
    DÉBUT
        somme <- somme + nombre
    FIN
    moyenne <- somme / taille de tableau
    RETOURNER moyenne
FIN
```

Exemple

```
# Cette fonction calcule la somme des nombres d'un tableau
FONCTION calculerSomme(tableau)
DÉBUT
    somme <- 0
    POUR CHAQUE nombre DANS nombres
    DÉBUT
        somme <- somme + nombre
    FIN
    RETOURNER somme
FIN
```

Exemple

```
# Cette fonction calcule la moyenne des nombres d'un tableau
FONCTION calculerMoyenne(tableau)
DÉBUT
    RETOURNER calculerSomme(tableau) / taille du tableau
FIN
```

Valeur de retour et effets de bord

- **Une fonction peut retourner une valeur**
- **Une fonction peut aussi effectuer des opérations qui modifient son environnement**
  - Exemples:
    - Écrire dans la console
    - Écrire des données dans un fichier
  - De telles opérations s'appellent des « effets de bord » (*side effects*)

Les fonctions en PowerShell

Les fonctions en PowerShell

- **Par convention, les noms des fonctions respectent le même format que les cmdlets (Verbe-Nom)**
  - Liste des verbes approuvés
  - Le langage ne nous force cependant pas à respecter cette convention
  - Dans le cadre du cours, le respect de cette convention est à votre discrétion
- **On peut, ou non, préciser les types des paramètres de la fonction**

Syntaxe

```
function Write-HelloWorld {
  Write-Output "Hello World!"
}

Write-HelloWorld
```

Syntaxe

```
function Write-HelloName {
  param ($name)
  Write-Output "Hello $name!"
}

Write-HelloName -name "Bob"
# Ou bien
Write-HelloName "Bob"
```

Syntaxe

```
function Write-HelloName {
  param ($firstName, $lastName)
  Write-Output "Hello $firstName $lastName!"
}

Write-HelloName -firstName "Bob" -lastName "L'éponge"
# Ou bien
Write-HelloName "Bob" "L'éponge"
```

Syntaxe

```
# Syntaxe alternative pour les paramètres
function Write-HelloName($firstName, $lastName) {
  Write-Output "Hello $firstName $lastName!"
}
```

Syntaxe

```
# Préciser les types des paramètres
function Write-HelloName([string]$firstName, [string]$lastName) {
  Write-Output "Hello $firstName $lastName!"
}
```

Syntaxe

```
function Write-HelloName {
  Write-Output ("Hello " + $args[0] + " " + $args[1] + "!")
}

Write-HelloName "Bob" "L'éponge"
```

Syntaxe

```
function concatenerPrenomNom($prenom, $nom) {
  return "$prenom $nom"
}


Write-Output (concatenerPrenomNom -prenom "Bob" -nom "L'éponge")
```

Fonctions vs Cmdlets

Fonctions vs Cmdlets

- **L'interpréteur du langage PowerShell a lui-même été développé dans un autre langage de programmation**
  - Langage C#
  - Environnement .NET de Microsoft
- **Les fonctions sont écrites en PowerShell directement**
- **Tandis que les cmdlets sont implantés en C#/.NET directement dans le code de l'interpréteur**

Fin de la présentation

Des questions?