


https://fr.wikipedia.org/wiki/Windows_PowerShell

Windows PowerShell

[Article](#) [Discussion](#)

Windows PowerShell, anciennement **Microsoft Command Shell (MSH)**, nom de code *Monad*, est une [suite logicielle](#) développée par [Microsoft](#) qui intègre une [interface en ligne de commande](#), un [langage de script](#) nommé PowerShell ainsi qu'un [kit de développement](#). Il est inclus dans [Windows 7](#), [Windows 8.1](#), [Windows 10](#) et [Windows 11](#) (y compris les versions grand public) et s'appuie sur le [framework Microsoft .NET](#).

 **PowerShell - Maîtrisez les bases fondamentales**

Introduction à PowerShell : Présentation de PowerShell

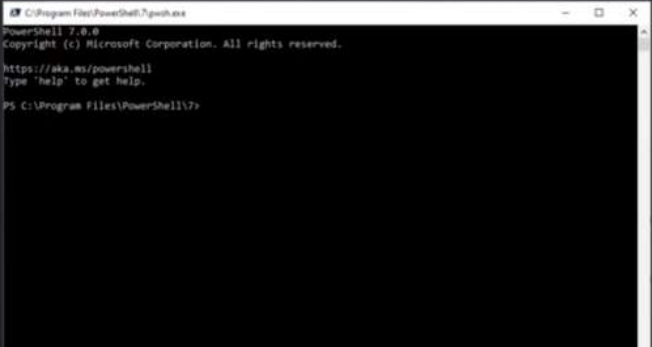
PowerShell est à la fois un interpréteur de commandes et un puissant langage de script orienté objet


S'appuie sur le Framework .Net Core


Administration des systèmes à travers des commandes et des scripts

Destiné à automatiser les tâches et permet de développer des outils

Multiplateforme et Open Source, les scripts PowerShell fonctionnent sur Windows, Linux et MacOS





 **PowerShell - Maîtrisez les bases fondamentales**

Introduction à PowerShell : Histoire de PowerShell

L'histoire de PowerShell commence en 2006 et à l'origine PowerShell était uniquement destiné pour Windows

De 2006 à 2016 PowerShell évolue normalement jusqu'à atteindre la version 5.1

En 2016, Microsoft décide de rendre PowerShell Open Source et Multiplateforme

Windows PowerShell

Première version sortie en 2006

Fonctionne uniquement sur Windows

N'est plus maintenu. Seul les bugs et les failles de sécurité seront corrigés.

PowerShell Core

Annoncé en 2016 et sortie en 2018

Multiplateforme (Windows, Linux, Mac)

OpenSource

PowerShell - Maîtrisez les bases fondamentales

Introduction à PowerShell : Notions d'objet

PowerShell est un langage de script orienté objet

Exemple d'objet de la vie courante :

Propriétés

Couleur : Jaune
Marque : Audi
Kilométrage : 50 000 KM

Méthodes

Rouler
Freiner



PowerShell - Maîtrisez les bases fondamentales

Introduction à PowerShell : Notions d'objet

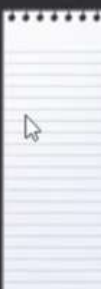
Exemple d'objet en informatique :

Propriétés

Taille du fichier : 1024 Ko
Date de création : 12/01/2020
Date de modification : 20/02/2020

Méthodes

Suppression fichier



PowerShell - Maîtrisez les bases fondamentales

Interpréteur de commandes : Structure des commandes

Les commandes PowerShell ont une structure cohérente et sont faciles à retenir

Structure commande : **Verbe-Nom** -Paramètre valeur

Exemple :

New-Item -ItemType **File** -Path **d:\test**

↑ ↑ ↑ ↑ ↑ ↑

Verbe Nom Paramètre Valeur Paramètre Valeur

Les commandes PowerShell se nomment « Cmdlet » et se prononcent Commandlet

Exécuter plusieurs Cmdlets : **Get-Date**; **Get-Item** C:\Windows

PowerShell - Maîtrisez les bases fondamentales

Interpréteur de commandes : Aide PowerShell

Pour obtenir de l'aide sur une Cmdlet, on utilise la Cmdlet **Get-Help**

Exemple :

Get-Help Get-Verb

Get-Help possède plusieurs paramètres :

Pour obtenir de l'aide en ligne : **Get-Help** Get-Verb -Online

Pour obtenir de l'aide détaillée : **Get-Help** Get-Verb -Detailed

Pour obtenir des exemples d'utilisation : **Get-Help** Get-Verb -Example

Pour obtenir de l'aide sur les concepts de PowerShell, on utilise les about

Exemple :

Pour obtenir de l'aide sur les structures de contrôle if, on utilise **Get-Help** about_if

Pour lister tous les about, on utilise : **Get-Help** about *

```
PS C:\Users\Administrateur> Get-Help clear
```

NOM
Clear-Host
RÉSUMÉ
SYNTAXE
Clear-Host [<CommonParameters>]
DESCRIPTION
LIENS CONNEXES
https://go.microsoft.com/fwlink/?LinkID=225747
REMARQUES
Pour consulter les exemples, tapez : "get-help Clear-Host -examples".
Pour plus d'informations, tapez : "get-help Clear-Host -detailed".
Pour obtenir des informations techniques, tapez : "get-help Clear-Host -full".
Pour l'aide en ligne, tapez : "get-help Clear-Host -online"

```
PS C:\Users\Administrateur> Get-Help about
```

Name	Category	Module	Synopsis
about_BeforeEach_AfterEach	HelpFile		performed
about_Mocking	HelpFile		Pester pro
about_Pester	HelpFile		Pester is
about_should	HelpFile		Provides a
about_TestDrive	HelpFile		A PSDrive

Exemple de commande :

Liste des alias : `get-alias`

```
PS C:\Atelier\Powershell> get-alias
```

CommandType	Name	Version	Source
Alias	% -> ForEach-Object		
Alias	? -> Where-Object		
Alias	ac -> Add-Content		
Alias	asnp -> Add-PSSnapin		
Alias	cat -> Get-Content		
Alias	cd -> Set-Location		
Alias	CFS -> ConvertFrom-String	3.1.0.0	Microsoft.PowerShell.Utility
Alias	chdir -> Set-Location		

On sélectionne uniquement les commandes qui contiennent alias dans leur titre :

```
PS C:\Atelier\Powershell> get-command -Noun alias
```

CommandType	Name	Version
Cmdlet	Export-Alias	3.1.0.0
Cmdlet	Get-Alias	3.1.0.0
Cmdlet	Import-Alias	3.1.0.0
Cmdlet	New-Alias	3.1.0.0
Cmdlet	Set-Alias	3.1.0.0

```
PS C:\Atelier\Powershell> get-command -Module Microsoft.PowerShell.Utility -Verb get -TotalCount 1
```


CommandType	Name	Version	Source
Function	Get-FileHash	3.1.0.0	Microsoft.PowerShell
Cmdlet	Get-Alias	3.1.0.0	Microsoft.PowerShell
Cmdlet	Get-Culture	3.1.0.0	Microsoft.PowerShell
Cmdlet	Get-Date	3.1.0.0	Microsoft.PowerShell
Cmdlet	Get-Event	3.1.0.0	Microsoft.PowerShell
Cmdlet	Get-EventSubscriber	3.1.0.0	Microsoft.PowerShell
Cmdlet	Get-FormatData	3.1.0.0	Microsoft.PowerShell
Cmdlet	Get-Host	3.1.0.0	Microsoft.PowerShell
Cmdlet	Get-Member	3.1.0.0	Microsoft.PowerShell
Cmdlet	Get-PSBreakpoint	3.1.0.0	Microsoft.PowerShell

Pour mettre plusieurs commande à la suite : on utilise le ;

```
PS C:\Atelier\Powershell> Get-date;get-service
lundi 6 mars 2023 15:22:22

Status      : Stopped
Name        : AarSvc_8b294
DisplayName : Agent Activation Runtime_8b294
```

Pour voir

```
PS C:\Atelier\Powershell> Get-PSReadLineOption

EditMode           : Windows
AddToHistoryHandler :
HistoryNoDuplicates : True
HistorySavePath     : C:\Users\Administrateur\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt
HistorySaveStyle    : SaveIncrementally
```

Normalement, toutes les commandes sont stockées dans un fichier txt : C:\Users\Administrateur\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt

```
PS C:\Atelier\Powershell> cat C:\Users\Administrateur\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt
py --help
py -v
py --v
py
py -v
py --help
python -v
& C:/Users/Administrateur/AppData/Local/Programs/Python/Python311/python.exe c:/Atelier/Python/Exercice/hello_world.py
py
clear
p
py
*
& C:/Users/Administrateur/AppData/Local/Programs/Python/Python311/python.exe -m pip install -U autopep8
& C:/Users/Administrateur/AppData/Local/Programs/Python/Python311/python.exe c:/Atelier/Python/Exercice/hello_world.py
py .\hello_world.py
py .\hello_world.py & C:/Users/Administrateur/AppData/Local/Programs/Python/Python311/python.exe c:/Atelier/Python/Exercice/hello_world.py
& C:/Users/Administrateur/AppData/Local/Programs/Python/Python311/python.exe c:/Atelier/Python/Exercice/hello_world.py
py .\hello_world.py
& C:/Users/Administrateur/AppData/Local/Programs/Python/Python311/python.exe c:/Atelier/Python/Exercice/hello_world.py
py .\hello_world.py
& C:/Users/Administrateur/AppData/Local/Programs/Python/Python311/python.exe c:/Atelier/Python/Exercice/hello_world.py
& 'C:/Users/Administrateur/AppData/Local/Programs/Python/Python311/python.exe' 'c:/Users/Administrateur/.vscode/extensions/ms-python
her' '63447' '--' 'c:/Atelier/Python/Exercice/hello_world.py'
c:; cd 'c:/Atelier/Python/Exercice'; & 'C:/Users/Administrateur/AppData/Local/Programs/Python/Python311/python.exe' 'c:/Users/Admini
n\debugpy\adapter/../../debugpy/launcher' '63458' '--' 'c:/Atelier/Python/Exercice/hello_world.py'
& C:/Users/Administrateur/AppData/Local/Programs/Python/Python311/python.exe c:/Atelier/Python/Exercice/exercice1_nom_prenom.py
& C:/Users/Administrateur/AppData/Local/Programs/Python/Python311/python.exe c:/Atelier/Python/Exercice/exercice1_nom_prenom.py
& C:/Users/Administrateur/AppData/Local/Programs/Python/Python311/python.exe c:/Atelier/Python/Exercice/exercice1_race_animaux.py
& C:/Users/Administrateur/AppData/Local/Programs/Python/Python311/python.exe c:/Atelier/Python/Exercice/exercice2_ouiMajeur_nonMineur
& C:/Users/Administrateur/AppData/Local/Programs/Python/Python311/python.exe c:/Atelier/Python/Exercice/exercice1_nom_prenom.py
```

```
& C:/Users/Administrateur/AppData/Local/Programs/Python/Python311/python.exe c:/Atelier/Python/Exercice/exercice1_nom_prenom.py
& C:/Users/Administrateur/AppData/Local/Programs/Python/Python311/python.exe c:/Atelier/Python/Exercice/exercice1_race_animaux.py
& C:/Users/Administrateur/AppData/Local/Programs/Python/Python311/python.exe c:/Atelier/Python/Exercice/exercice1_nom_prenom.py
& C:/Users/Administrateur/AppData/Local/Programs/Python/Python311/python.exe c:/Atelier/Python/Exercice/exercice3_aireCarre.py
```

PowerShell - Maîtrisez les bases fondamentales

Interpréteur de commandes : Pipeline

Le Pipeline est symbolisé par le caractère `|` et il permet de connecter plusieurs Cmdlets entre elles

Une Cmdlet peut recevoir en entrée la sortie d'une autre Cmdlet

Exemple d'utilisation du Pipeline avec la Cmdlet `Get-Command` :



PowerShell - Maîtrisez les bases fondamentales

Interpréteur de commandes : Pipeline

Pipeline - Canalisation

Get-Command → Résultat → Out-File

Get-Content → Sort-Object → Out-File

`Get-Content h:\non_trier.txt | Sort-Object | Out-File h:\fichier_trier.txt`

Exercice : on a besoin de transférer les données d'un fichier CVS dans un autre fichier pour le trier dans l'ordre alphabétique :

Sort-object :

```
PS C:\Atelier\Powershell> Get-help Sort-Object

NOH
Sort-Object

RÉSUMÉ
Sorts objects by property values.

SYNTAXE
Sort-Object [[-Property] <System.Object[]>] [-CaseSensitive] [-Culture <System.String>] [-Descending] [-InputObject
<System.Management.Automation.PSObject>] [-Unique] [<CommonParameters>]

DESCRIPTION
The 'Sort-Object' cmdlet sorts objects in ascending or descending order based on object property values. If sort properties aren't included in a
command, PowerShell uses default sort properties of the first input object. If the input object's type has no default sort properties, PowerShell
attempts to compare the objects themselves. For more information, see the Notes (#notes) section.

You can sort objects by a single property or multiple properties. Multiple properties use hash tables to sort in ascending order, descending
order, or a combination of sort orders. Properties are sorted as case-sensitive or case-insensitive. Use the Unique parameter to remove duplicates
```

```
from the output.
```

Out-File :

```
PS C:\Atelier\Powershell> Get-help Out-File

NOM
    Out-File

RÉSUMÉ
    Sends output to a file.
```

Correction :

```
Get-Content -Path .\data-NBKwx54O92ekpK2aTq5oL.csv | Sort-Object | Out-File C:\Atelier\Powershell\test_trie_nom.csv
```

```
PS C:\Atelier\Powershell> Get-Content -Path .\data-NBKwx54092ekpK2aTq5oL.csv | Sort-Object | Out-File C:\Atelier\Powershell\test_trie_nom.csv
PS C:\Atelier\Powershell> cat .\test_trie_nom.csv
Amelia Odum
Andrew Bolton
Anika Mccoy
Astra Stevens
Austin Murray
Barry Hubbard
Beatrice Good
Benedict Carpenter
```

Autre commande : extraire les processus en cours dans un fichier CSV

Get-Process | Select-Object Name | Out-File ./test_extract_process.csv

```
PS C:\Atelier\Powershell> Get-Process | Select-Object Name | Out-File ./test_extract_pro
```

```
PS C:\Atelier\Powershell> cat .\test_extract_process.csv
```

Name

AnyDesk
AnyDesk
ApplicationFrameHost
armsvc
audiodg

PowerShell - Maitrisez les bases fondamentales

Interpréteur de commandes : Format d'affichage

Format-Table

Format-List

Exemple :

La Cmdlet **Get-Host** envoie par défaut sa sortie sous forme de liste

Pour envoyer le résultat de Get-Host sous forme de tableau : [Get-Host](#) | [Format-Table](#)





Correction TP :

#TP POWERSHELL

Question 1 : Trouver les services qui sont démarrés !

```
Get-Service | Where-Object {$_.status -eq "running"}
```

Question 3 - Afficher toutes les CMDLETs disponibles en PS

```
Get-Command | Where-Object {$_.CommandType -eq "Cmdlet"}
```

Question 3 - Afficher seulement les CMDLETs qui commence par Get

```
Get-Command -Name *get* | Where-Object {$_.CommandType -eq "Cmdlet"}
```

Question 4 - Trouver comment naviguer entre les répertoires en PS

```
PS C:\Users\aresv> cd .\OneDrive\  
PS C:\Users\aresv\OneDrive> ls
```

Question 5 - Trouver comment afficher le contenu du répertoire courant en PS

```
ls
```

Question 6 – Changer la couleur du fond d'ecran lorsqu'un message d'erreur apparait, en blanc.

```
$Host.PrivateData.ErrorBackgroundColor = "White"
```

```
PS C:\Users\aresv\OneDrive> $Host.PrivateData.ErrorBackgroundColor = "White"  
PS C:\Users\aresv\OneDrive> $Host.PrivateData
```

Question 7 – Ecrire dans un fichier qui va s'appeler fichier.txt, l'ensemble des commandes de type « Function », avec le verbe « Get » et appartenant au module Storage.

```
Get-Command -Name *get* | Where-Object {$_.CommandType -eq "Function"} | Out-File  
.\fichier.txt
```

Question 8 – Afficher uniquement le numéro du processus system : Le resultat devra ressembler à cela :

Id : 4

```
Get-Process | Select-Object ID | Format-List
```

Question 9 – Afficher le details des propriétés du processus explorer sous forme de list.

```
Get-Process | Where-Object ProcessName -eq Explorer | Format-List
```


Question 10 – Afficher la liste des services au format table en ne gardant que leur nom et en les regroupant par leur status.

```
Get-Service | Sort-Object -Property Status | Format-Table
```

Question 11 - Chercher sur votre OS tous les fichiers supérieur à 300 Mb

```
Get-ChildItem -Force -Recurse -Path "C:\" | Where-Object -FilterScript {  
>> ($_.Length -ge 300mb)  
>> }
```

Question 12 - vous devez trouver tous les fichiers de plus de 300 Mb et les mettre dans un fichier .csv

```
Get-ChildItem -Force -Recurse -Path "C:\" | Where-Object -FilterScript {  
>> ($_.Length -ge 300mb) } | Out-File .\fichier.txt
```

Question 13 : Afficher la liste des fichiers et dossiers dans c:\windows qui sont inférieure 1GB.

```
Get-ChildItem -Force -Recurse -Path "C:\Windows\" | Where-Object -FilterScript {  
>> ($_.Length -le 1Gb)  
>> }
```

Question 14 : Créer un alias pour lancer un processus.

New-Alias ping Test-Connection

L'option **Group-Object** Nous permet de regrouper les objet par un attribut -> ici leur status

```
Windows PowerShell  
PS C:\Users\User1> get-service | Group-Object -Property status  
  
Count Name Group  
-----  
119 Running {AarSvc_757ed, AdobeARMSvc, AESMSvc, AnyDesk...}  
168 Stopped {AJRouter, ALG, AppIDSvc, AppMgmt...}  
  
PS C:\Users\User1>
```

Sélectionner un objet -> **Select-Object** : nous permet de sélectionner les services en fonction des arguments

```
PS C:\Users\User1> get-service | Select-Object -Last 3 -Property DisplayName  
  
DisplayName  
-----  
Jeu sauvegardé sur Xbox Live  
Xbox Accessory Management Service  
Service de mise en réseau Xbox Live
```

Attribut **-ErrorAction** : gestion des erreurs

<https://www.tutorialspoint.com/how-to-use-the-erroraction-parameter-in-powershell>

```
PS C:\Users\User1> get-service lol0 -ErrorAction Inquire; get-date  
Confirmer
```

```

Impossible de trouver un service assorti du nom « lolo ».
[O] Oui [I] Interrompre la commande [S] Suspendre [?] Aide (la valeur par défaut est « 0 »)
get-service : Impossible de trouver un service assorti du nom « lolo ».
Au caractère Ligne:1 : 1
+ get-service lolo -ErrorAction Inquire; get-date
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (lolo:String) [Get-Service], ServiceCommandException
+ FullyQualifiedErrorId : NoServiceFoundForGivenName,Microsoft.PowerShell.Commands.GetServiceCommand

```

mardi 7 mars 2023 11:34:26

```

PS C:\Users\User1> get-service lolo -ErrorAction Continue; get-date
get-service : Impossible de trouver un service assorti du nom « lolo ».
Au caractère Ligne:1 : 1
+ get-service lolo -ErrorAction Continue; get-date
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (lolo:String) [Get-Service], ServiceCommandException
+ FullyQualifiedErrorId : NoServiceFoundForGivenName,Microsoft.PowerShell.Commands.GetServiceCommand

```

mardi 7 mars 2023 11:35:40

```

PS C:\Users\User1> get-service lolo -ErrorAction Ignore; get-date

```

mardi 7 mars 2023 11:35:48

.CreationTime pour voir la date de création des éléments d'un dossier

```

PS C:\Users\User1> cd .\Documents\
PS C:\Users\User1\Documents> (Get-ChildItem C:\Users\User1\Documents\Angular\).CreationTime

```

lundi 22 août 2022 11:17:17
vendredi 19 août 2022 10:25:37
lundi 10 octobre 2022 14:09:43

.LastWriteTimeUtc.DayOfWeek : voir la date de la dernière modification affiché sur les fichiers d'un dossier :

```

PS C:\Users\User1\Documents> (Get-ChildItem C:\Users\User1\Documents\Angular\).LastWriteTimeUtc.DayOfWeek
Wednesday
Friday
Monday
PS C:\Users\User1\Documents>

```

Select-String permet de chercher un résultat avec l'attribut -SimpleMatch

```

Windows PowerShell
PS C:\Users\User1\Documents> Get-service | Select-Object -Property DisplayName | Select-String -SimpleMatch "Windows"

```

@{DisplayName=Générateur de points de terminaison du service Audio Windows}
@{DisplayName=Audio Windows}
@{DisplayName=Journal d'événements Windows}
@{DisplayName=Service de cache de police Windows}
@{DisplayName=Serveur de trame de la Caméra Windows}
@{DisplayName=Service Point d'accès sans fil mobile Windows}
@{DisplayName=Serveur Gestionnaire de licences Windows}
@{DisplayName=Windows Mixed Reality OpenXR Service}
@{DisplayName=Pare-feu Windows Defender}
@{DisplayName=Windows Installer}
@{DisplayName=Service de simulation de perception Windows}
@{DisplayName=Service PushToInstall de Windows}
@{DisplayName=Expérience audio-vidéo haute qualité Windows}
@{DisplayName=Sauvegarde Windows}

New-Item permet de créer des dossiers/document mais aussi de vérifier s'ils existent ou non avec l'attributs .Exists ou encore le supprimer avec .Delete

```

PS C:\Users\User1\Documents> (Get-Item C:\Users\User1\Documents\01c.pdf).Delete()
PS C:\Users\User1\Documents> New-Item -Name titi.txt -ItemType file -Value "salut à toi"

```

Répertoire : C:\Users\User1\Documents

Mode	LastWriteTime	Length	Name
-a----	07/03/2023 11:57	17	titi.txt


```
PS C:\Users\User1\Documents> New-Item -Name dossier67 -ItemType Dir
```

Répertoire : C:\Users\User1\Documents

Mode	LastWriteTime	Length	Name
d----	07/03/2023 11:58		dossier67

```
C:\Users\User1\Documents\01c.pdf
PS C:\Users\User1\Documents> (Get-Item C:\Users\User1\Documents\01c.pdf).Ex
True
PS C:\Users\User1\Documents> (Get-Item C:\Users\User1\Documents\01c.pdf).De
PS C:\Users\User1\Documents>
```

Attribut ItemType : permet de dire si c'est un fichier (file) ou un dossier (directory)

Copy-Item : Pour copier un dossier/files

```
PS C:\Users\User1\Documents> Copy-Item .\testo.txt -Destination lolo.txt
```

Tp :

TP Powershell Pipeline

Le fichier dont vous auriez besoin : https://mohamed-m2i-bucket.s3.eu-west-1.amazonaws.com/fake_directory_tp+2.zip

Exercice 1 : Écrivez une commande PowerShell qui récupère tous les fichiers .bin dans un répertoire spécifié, les trie par taille décroissante, puis affiche les 10 plus gros fichiers.

Exercice 2 : Écrivez une commande PowerShell qui récupère la liste de tous les processus en cours d'exécution, trie les résultats par utilisation de la mémoire en Mo, puis affiche les 5 processus les plus gourmands.

Exercice 3 : Écrivez une commande PowerShell qui récupère la liste de tous les fichiers et dossiers dans un répertoire spécifié, les trie par date de dernière modification (du plus récent au plus ancien), puis affiche les 5 éléments les plus récents.

Exercice 4 : Écrivez une commande PowerShell qui récupère la liste de tous les utilisateurs de l'ordinateur, trie les résultats par ordre alphabétique du nom, puis affiche le nom et l'adresse e-mail de chaque utilisateur.

Exercice 5 : Écrivez une commande PowerShell qui récupère la liste de tous les services en cours d'exécution, trie les résultats par état (en cours d'exécution, en pause ou arrêté), puis affiche le nom et l'état de chaque service.

Exercice 6 : Écrivez une commande PowerShell qui récupère la liste de tous les fichiers .log dans un répertoire spécifié, filtre les résultats pour ne conserver que ceux qui contiennent le

mot-clé "erreur", puis trie les résultats par date de dernière modification (du plus récent au plus ancien).

Exercice 7 : Écrivez une commande PowerShell qui récupère la liste de tous les processus en cours d'exécution, filtre les résultats pour ne conserver que ceux qui ont été lancés il y a plus de 24 heures, puis affiche le nom et l'ID de chaque processus.

Exercice 1 :

```
Get-ChildItem -Name *.bin* | Sort-Object -property Length | Select-Object -First 10
```

Exercice 2 :

```
Get-Process | Sort-Object PM -Descending | Select-Object -First 5
```

Exercice 3 :

```
Get-ChildItem -Name -Recurse | Sort-Object LastWriteTime | Select-Object -First 5
```

Exercice 4 :

```
Get-LocalUser | Sort-Object
```

Exercice 5 :

```
Get-Service | Sort-Object -Property Status | Format-List Name,Status
```

Exercice 6 :

```
Select-String -Path *.log -Pattern 'Erreur' | Sort-Object -Property LastWriteTime
```

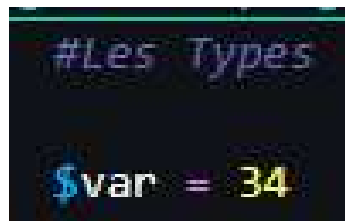
Exercice 7 :

https://docs.oracle.com/cd/E24843_01/html/E23087/spprocess-1.html

```
Get-Process | Where-Object {$_.STIME -gt "24:00"} | Format-List Id,Name
```

Les variables :

Déclarer variable :



```
scripting > > scripting_tp_powerhsell.ps1 > ...
1 #Les Types
2
3 $var = 34
4
5 $var.GetType()
6
7 $var = $var +3
8
9 Write-Host "La valeur de var est : " $var

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL CODEWHISPERER REFERENCE LOG
PowerShell Extension v2023.2.1
Copyright (c) Microsoft Corporation.

https://aka.ms/vscode-powershell
Type 'help' to get help.

PS C:\Users\aresv\Documents\atelier\powershell\fake_directory_tp>
PS C:\Users\aresv\Documents\atelier\powershell\fake_directory_tp> . 'C:\Users\aresv\Documents\atelier\powershell\fake_directory_tp\scripting\scripting_tp_powerhsell.ps1'

IsPublic IsSerial Name BaseType
-----
True True Int32 System.ValueType
La valeur de var est : 37

PS C:\Users\aresv\Documents\atelier\powershell\fake_directory_tp>
```


Dans le cas de figure défini plus bas : les deux variables se sont concaténé car ce ne sont pas les même types de variable (**string** et **int**)

```
7
8 $var2 = "67"
9 $var2 = $var2 + 4
10
11 Write-Host "la valeur de var est de : " $var
12
13 Write-Host $var2
14
15
16
17
18
19
```

PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL

```
PS C:\Users\User1\Documents\powershell> . 'C:\Users\User1\Documents\powershell\demo.ps1'
```

IsPublic	IsSerial	Name	BaseType
True	True	Int32	System.ValueType

```
la valeur de var est de : 37
674
```

```
Write-Host $var2

[long]$var3 = 6543322

$var3.GetType()

$var3 = "lolo"
```

Exemple de programme :

```
$demo = 45

if($demo -lt 89){
    Write-Host "Oui la variable demo est plus petite, elle est de $demo"
}else{
    Write-Host "Non"
}
```

Autre exemple de programme qui vérifie si un fichier existe et retourne son contenu s'il t'existe:

```
if(Test-Path C:\Users\aresv\Documents\atelier\powershell\fake_directory_tp\file1.log){
    Get-Content C:\Users\aresv\Documents\atelier\powershell\fake_directory_tp\file1.log
}else{
    Write-Host "Le fichier n'existe pas !"
}
```

Création d'un nouveau répertoire en script :

```
if(!(Test-Path C:\Users\aresv\Documents\atelier\powershell\fake_directory_tp\aws)){
    New-Item -ItemType Directory -Path C:\Users\aresv\Documents\atelier\powershell\fake_directory_tp\aws
    Write-Host "répertoire créer"
}else{
    Write-Host "répertoire existe déjà"
}
```

Création de fichier dans un dossier :

```
#Création de fichier dans un dossier :
```

```
for ($i=1; $i -le 10; $i++){
    New-Item -ItemType file -Name file$i.txt -value "je suis un fichier $i" -Path .\aws
}
```

Script pour afficher les processus :

```
#Afficher Les processus

$processes = Get-Process
foreach($process in $processes){
    Write-Host "$($process.ProcessName) $($process.Id)"
}
```

```
# Script pour compter le nombre de ligne de chacun des fichiers d'un dossier
$files = Get-ChildItem -Path .\dossier -Filter *.txt
foreach($file in $files){
    $lines = Get-Content $file.FullName | Measure-Object -Line
    Write-Host "$($file.Name) : $($lines.Lines) lignes"
}
```

```
$res = Read-Host "donne moi un nombre ? "

Write-Host "resultat : $res"
```

Tp :

TP-Powershell-Script

Exercice 1 : Créer un script qui va vous permettre de créer 10 dossiers et chacun de ces dossiers va contenir 5 fichiers avec une extension différentes : .csv, .txt, .ppt, .doc et .xls.

Exercice 2 : Ecrire un script qui va demander le chemin d'un dossier et qui nous donner le nom du fichier le plus volumineux de ce dossier.

Exercice 3 : Créer un script qui va déplacer des fichiers qui se trouvent sur le bureau avec une extension .jpeg, .png, .jpg dans le dossier Pictures du Home Directory de l'utilisateur, les fichiers avec une extension .doc, .ppt, .xls, .csv et .csv dans le dossier Dossier TP du Dossier Documents du Home Directory de l'utilisateur

Exercice 1 :

```
$var_file = "test_file"
$var_csv = csv
$var_txt = txt
$var_ppt = ppt
$var_doc = doc
$var_xls = xls
```

```
for ($i=1; $i -le 5; $i++){  
    New-Item -ItemType Directory -Name Directory$i -Path .\aws  
    New-Item -ItemType File -Name $var_file$i.$var_csv -Path .\aws\Directory$i  
    New-Item -ItemType File -Name $var_file$i+1.$var_txt -Path .\aws\Directory$i  
    New-Item -ItemType File -Name $var_file$i+2.$var_ppt -Path .\aws\Directory$i  
    New-Item -ItemType File -Name $var_file$i+3.$var_doc -Path .\aws\Directory$i  
    New-Item -ItemType File -Name $var_file$i+4.$var_xls -Path .\aws\Directory$i  
}
```