

## 10. Powershell II.

# Visszatekintés

- Számítógépek, számábrázolás, kódolás, fájlrendszerek
- Alapvető parancsok, folyamatok, reguláris kifejezések
- Változó, parancs behelyettesítés, aritmetikai, logikai kifejezések
- Script vezérlési szerkezetek, Sed, AWK
- Batch, WSH
- PS áttekintés, Powershell változók, műveletek
- Alapvető Powershell parancsok, vezérlési szerkezetek

# Mi jön ma?

- PowerShell függvények
- PowerShell könyvtári elemek
- Minden ami fontos.....

# Függvények PowerShellben

- `function név(par) { fvtörzs }`
  - Bárhol elhelyezkedhet, de csak a definíció után használható!
  - eredmény: `return` utasítás
  - `$lastexitcode` változó (utoljára végrehajtott külső program visszatérési értéke)
  - Hívás: `név 5` vagy `név(5)`
  - Több paraméter:
    - `név1 $x $y`

```
function nfaktor($n)
{
    $f=1
    for($i=2;$i -le $n;$i++) {$f*=$i}
    return $f
}
echo "N faktoriális"
nfaktor(5) vagy nfaktor 5
```

# Klasszikus ill. paraméter blokk

Klasszikus  
paraméter megadás:

```
function global:Where-BigProcess(  
    [int]      $meret = 200MB,  
    [String]   $property = "VirtualMemorySize",  
    [String]   $formatString = "Total {2} = {1}"  
) { body }
```

PARAMéter blokk  
megadás

```
function global:Where-BigProcess {  
    PARAM(  
        [int]      $meret = 200MB,  
        [String]   $property = "VirtualMemorySize",  
        [String]   $formatString = "Total {2} = {1}"  
    )  
    body  
}
```

# Függvény paraméterek

- Megadásuk jellemzően a klasszikus minta szerint:
  - `function alma($név,$méret) { ...}` vagy param blokk
- Alapértelmezett adatok jelzése:
  - `$név="zoli"`, mint pl. C#-ban
  - Kötelezően adandó paraméter:
    - `function alma( [parameter(Mandatory=$true)] $név) { ...}`
    - Korábban: `alma( $név=$(throw "Kérek nevet!!")) { ...}`
- Változószámú paraméteres függvény
  - Mint .NET-ben, `$args` tömb

# Függvény paraméterek megadása

- `Function alma($név,$ár,$szín) {...}`
- Pozíció szerinti megadás:
  - Hívás: `alma „jonatán” 150 ”piros”;`
- Név szerinti megadás:
  - Hívás: `alma –név golden –szín sárga`
    - Az árat nem adtuk meg, üres lesz!
    - Rövid forma: `alma –n golden –s sárga`
- Keverhetjük a két megadási módot!
  - Pl: `alma –s zöld zöld 250`
  - Lehetőleg ne használjuk ezt a kevert módot!

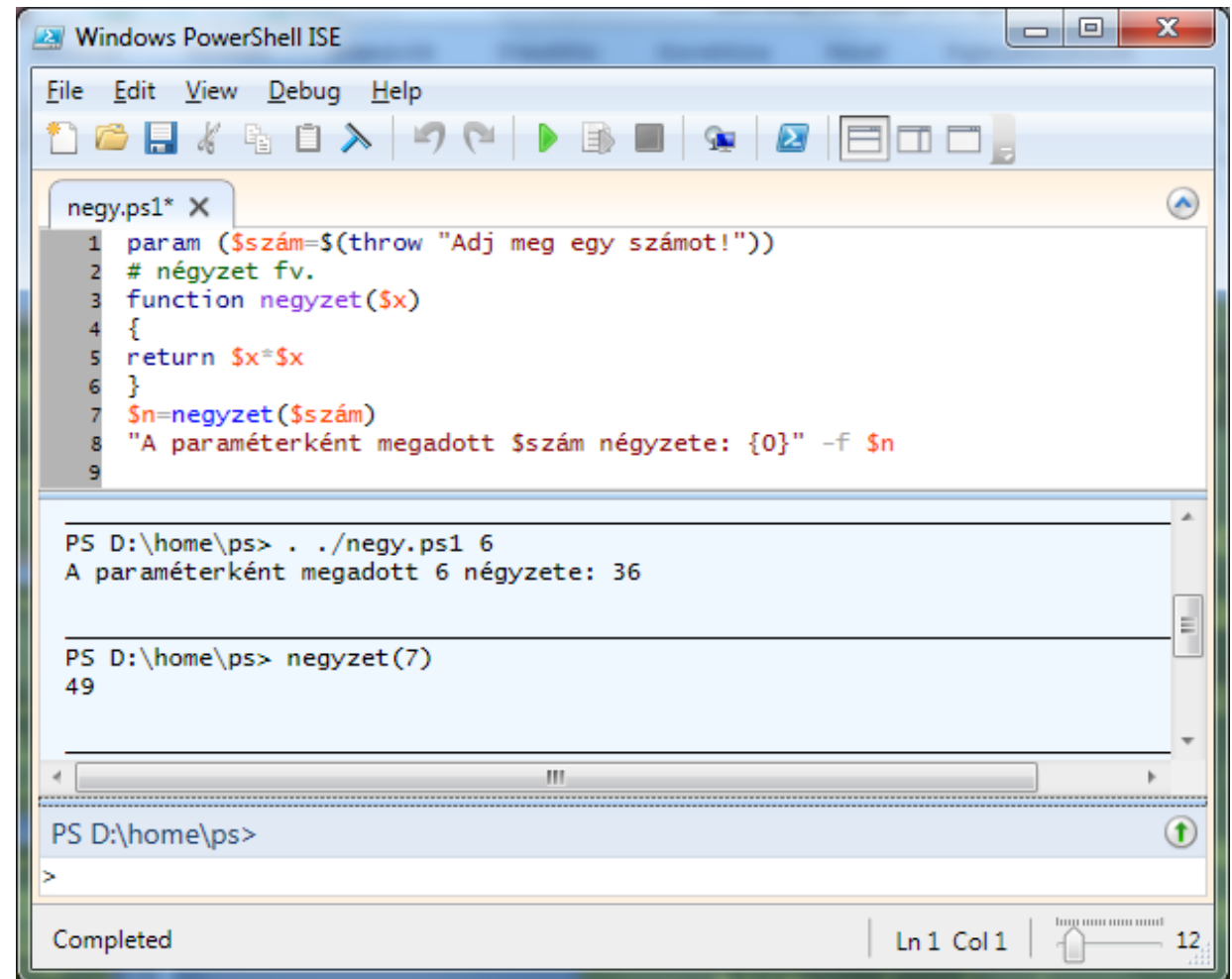
# Függvény, változó szint módosítás (dot sourcing)

- Lehet függvényen belül is függvényt definiálni. Belső függvény nem hívható közvetlenül!
  - Pontos indítás: . Fv
    - Azt eredményezi, hogy ezentúl az Fv-n belüli függvények is láthatók, használhatók közvetlenül.
- Függvény lokális változó nem látható kívül.
  - Pontos indítás: . Fv
  - Lokális változó kívül is látszik!
- Óvatosan a használattal!!



# Függvény közvetlen használat

- Ez példa az előző „szint” módosító . használatra!
  - „Dot-Sourcing” operátor
  - Shell script használat esetén is ugyanez!



```
negy.ps1 X
1 param ($szám=$(throw "Adj meg egy számot!"))
2 # négyzet fv.
3 function negyzet($x)
4 {
5     return $x*$x
6 }
7 $n=negyzet($szám)
8 "A paraméterként megadott $szám négyzete: {0}" -f $n
9

PS D:\home\ps> . ./negy.ps1 6
A paraméterként megadott 6 négyzete: 36

PS D:\home\ps> negyzet(7)
49

PS D:\home\ps>
>
```

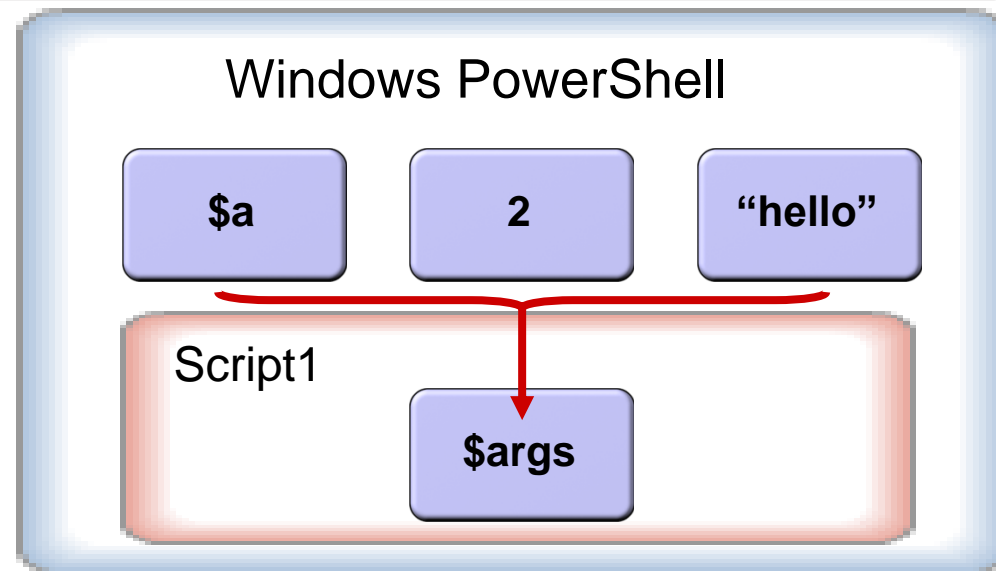
Completed | Ln 1 Col 1 | 12

# Script paraméterek mint tömb

```
# script file argtest.ps1  
foreach( $i in $args ){“Paraméterek {0:D};” -f $i }
```

```
PS> $a = 10
```

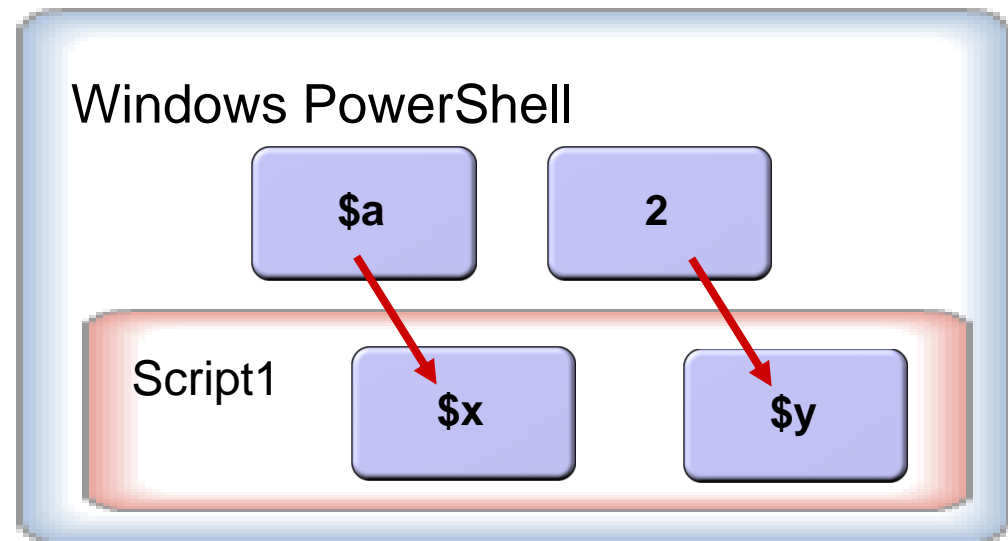
```
PS> .\argtest.ps1 $a 2 “hello”
```



# Nevesített script paraméterek

```
# nevesített paraméterek  
param( $x, $y )  
"A `$x={0}" -f $x  
"A `$y={0}" -f $y
```

```
PS> $a = 10  
PS C:\> .\paramtest.ps1 $a 2
```



# Nevesített és normál paraméterek közös használata

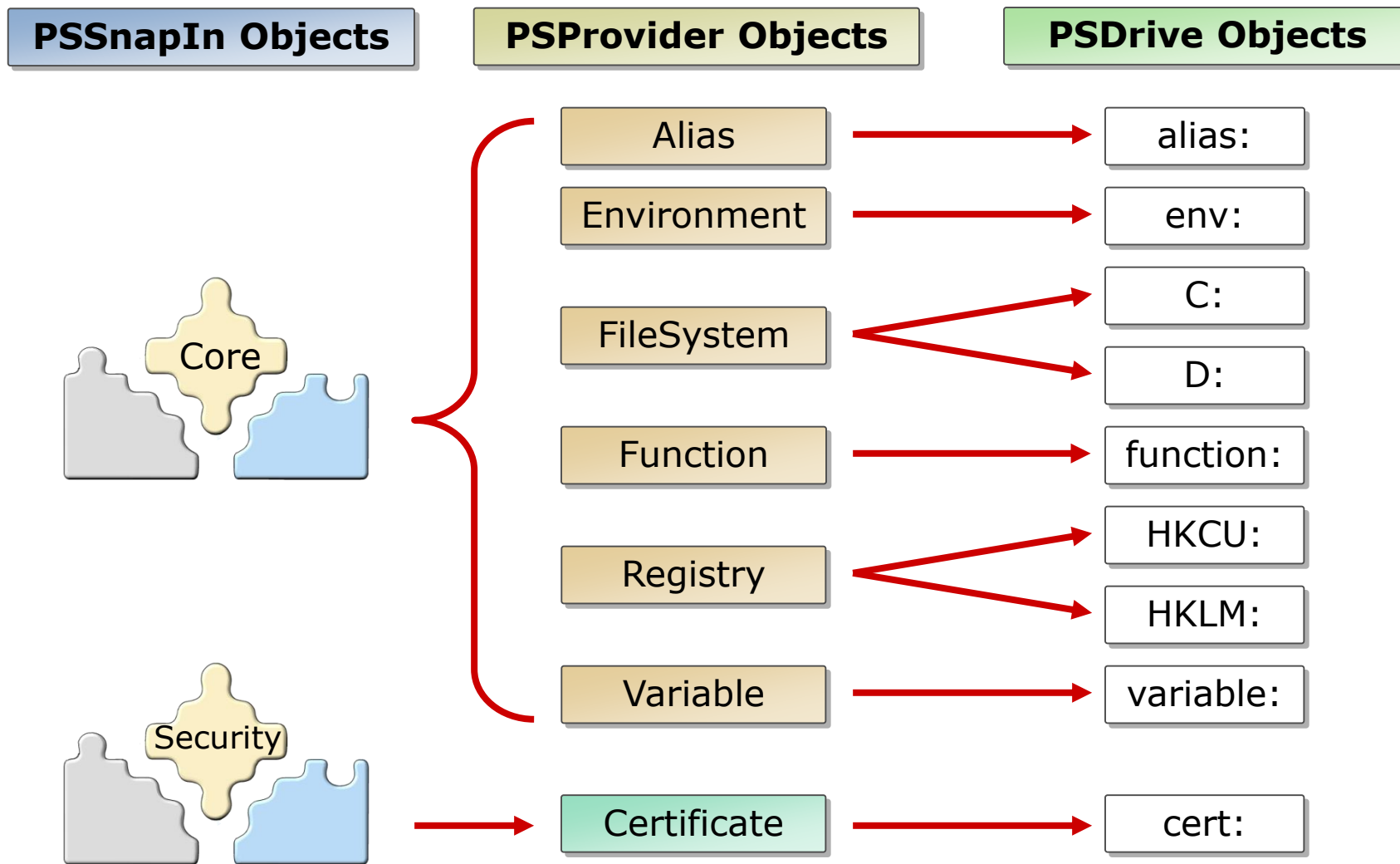
- Lehet keverni a nevesített és normál paramétereket.

```
#  
param($x,$y)  
write-output $args.length  
# write-output $args.count  
# ez ugyanaz mint az előző  
write-output $x  
write-output $y  
write-output "A 2. paramétertől kezdve:"  
foreach( $i in $args )  
    {"A script paraméterek sorban {0:D};" -f $i }
```

# PowerShell (fontosabb) belső változói

- `$_` - aktuális csővezeték objektum, (foreach)
- `$?` - előző parancs eredmény státusza, logikai
- `$home` - felhasználó home könyvtára
- `$$` - előző parancssor utolsó szava
- `$^` - előző parancssor első szava
- `$host` - aktuális kiszolgáló (nem egy név!!)
- `$myinvocation` – aktuális futtatási információk
- `$pshome` - PS install könyvtár
- `$profile` – felhasználó profile fájl neve

# PS forrás-Drive



# Adatforrások, Provider-ek

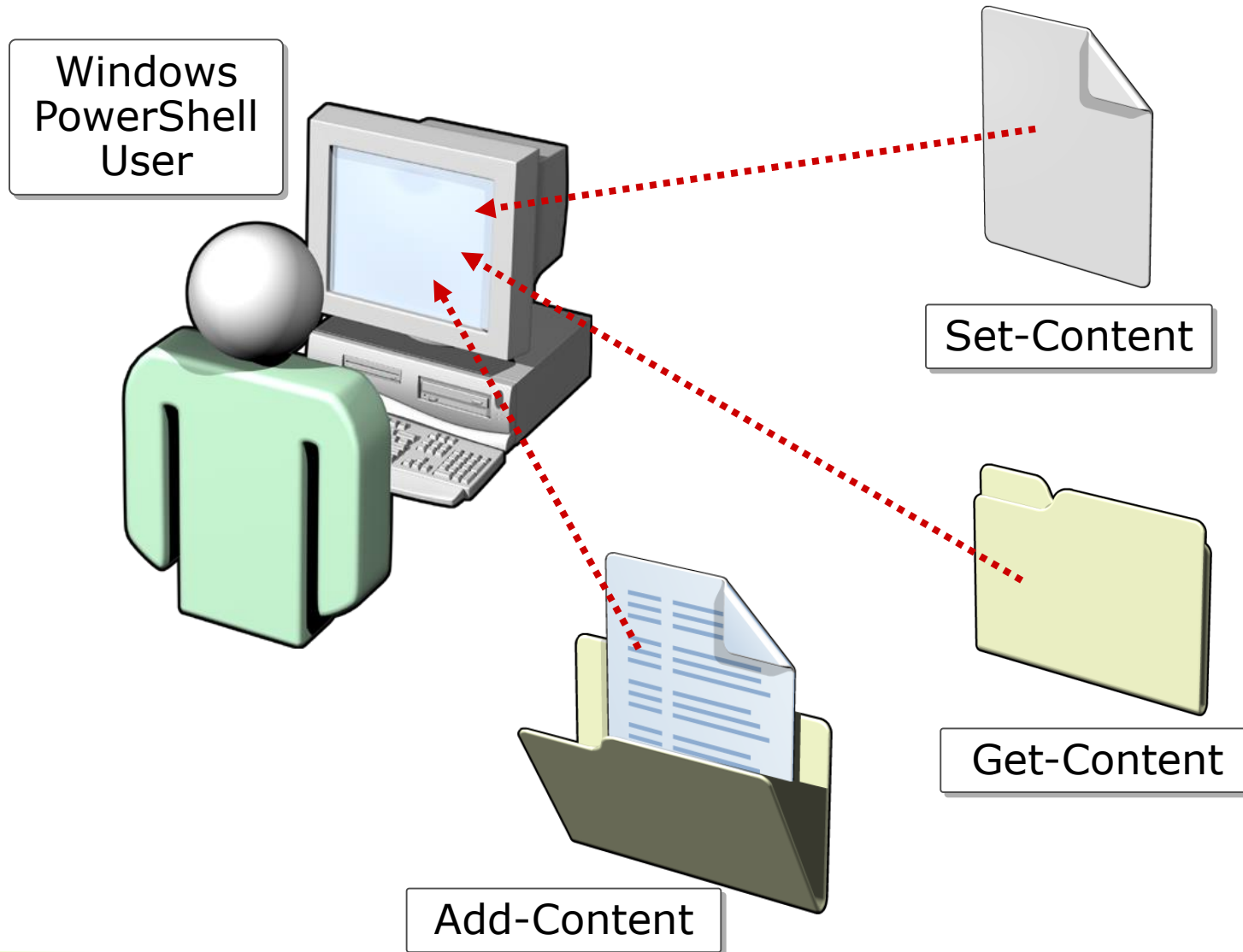
- dir parancs valójában a Get-ChildItem, az aktuális adatforrás elemeit adja meg.
- Milyen források vannak?
  - Get-PSDrive, Get-PSProvider
- Hogy tudunk váltani?
  - Set-Location, pl: set-location alias:\ul>  - Cd hklm:
- dir – Get-ChildItem mire vonatkozik?
- set-location d:\home

# Output átirányítás (fájl létrehozás)

- „Hajrá Fradi!” >fradi.txt # felülírás, új fájl
  - „Kukába” >\$null
  - Del fradi1.txt 2> \$null # hibakimenet máshova
    - 1> ez nincs, helyette simán >
- Get-Content fradi.txt # PS típus
  - Cat fradi.txt # unix típus
  - Type fradi.txt # dos típus
- Append: >>
  - „Pápa-FTC 0:5” >>fradi.txt # Ha nincs fájl, létrehozza!
- Nincs < vagy << átirányítás!



# Fájlok elérése



# Példa fájlok használatára

- `dir | set-content dir.txt` # csak a fájl nevek kerülnek bele, miért? (a `dir` elemei objektumok)
- `dir | out-string | set-content dir1.txt` # teljes tábla kiírásra kerül
  - `dir | out-file dir2.txt` # a teljes táblaszerű kiírás
- `dir | out-printer` # az alapértelmezett nyomtatóra nyomtatunk
- `dir | export-csv dir.csv; import-csv dir.csv`
- `dir | export-clicxml dir.xml; import-clicxml dir.xml`
- `Out-null` # mint a `/dev/null`

# Adatok szűrése- Where parancs

- Adatok megadása:
  - Pipeline
  - Paraméter segítséggel: -inputobject
- Where-Object { szűrőblokk }
  - A szűrőblokk logikai igaz érték esetén átengedi a szűrőn az adott objektum elemet. (logikai operátorok: -gt, -lt, -eq, stb)
  - PL: dir | where-object { !\$\_.PSisContainer}
    - Nem könyvtárak listázása.
    - \$\_ Pipe elem aktuális értéke (objektuma).

# Where-Object – foreach példa

- A foreach két változata where-object –tel
  - A where a unix grep-hez hasonlít

```
$list = Get-ChildItem -Recurse | where-object  
{!$_.PSisContainer} #könyvtárakat is megnézzük  
foreach ( $file in $list )  
{  
    $név = $file.name; $size = $file.length  
    write-output "A $név fájl mérete: $size byte."  
}
```

```
Get-ChildItem -Recurse | where-object  
{!$_.PSisContainer } |ForEach-Object {  
    $név = $_.name; $size = $_.length  
    write-output "A $név fájl mérete: $size  
byte." }
```

# Reguláris kifejezések PowerShellben

Karakter	Jelentés	Példa
.	Tetszőleges karakter	<b>.o.th</b>
<b>[xyz]</b>	Egy a felsoroltak közül	<b>[CMRS]andy</b>
<b>[x-z]</b>	Egy az intervallumból	<b>[A-Z]eramy</b>
<b>^</b>	Szöveg kezdet	<b>^Subject:</b>
<b>\$</b>	Szöveg vége	<b>meeting\$</b>
<b>*</b>	0 vagy több ismétlése az előzőnek	<b>W.*s</b>
<b>+</b>	1 vagy több az előzőből	<b>[MZ]+any</b>
<b>?</b>	0 vagy 1 előző	<b>[MZ]?any</b>
<b>\</b>	Speciális karakter a következő	<b>Try\\$</b>

# –like és –match operátor példák

```
PS> gci -r | where-object { $_.name -match "\.x[m1][1s]" }  
... # reg. kif, az XML, XLS, XMS, XLL fájlok.
```

```
PS> get-process | where-object { $_.name -match "ss$" }  
... # összes system service processz.
```

```
PS> $p = Get-Content planes.txt; `  
    $p -match "a[ie]ro?plane"  
... # sorok az airplane, aeroplane, szavakból.
```

```
PS> $p -like "*plane*"  
... # sorokban a plane bent van.
```

# Rendezés – Sort PowerShellben

- Sort-Object [tulajdonság] –paraméterek
  - Ha tulajdonság adott, akkor a szerint rendez
    - Pl: length
  - Paraméterek: -unique, -casesensitive, -descending, -culture név, Lásd: Get-Culture, Set-Culture parancsok
  - Ha nincs semmi paraméter, tulajdonság, akkor alapértelmezésként a teljes objektumot, név szerint, növekvő sorrendben, kis-nagybetű különbséget nem figyelembe véve rendez!
  - PL: dir|sort

# Select-Object - Kiválogatás

- Objektum jellemzőket válogat ki
  - Pl: `get-process | select-object processname,Id`
  - Kiválasztjuk a processzek nevét, azonosítóját.
- Fontosabb paramétereit: `-first 4`, `-last 5`, `-unique`
- Példa: (saját hashtábla kifejezés írható)

```
$ get-process | sort-object processname | select-object -first 5
$
$ $p = get-process | select-object ProcessName,@{Name=„Kezdő idő”;
Expression = {$_.StartTime}}
$ $p
```



# Objektműveletek (Measure-Object)

- Measure-Object, átlag, összeg stb.
  - Get-content dir.txt | measure-object -line -word -char
- Objektum egy tulajdonsága alapján végzi a műveleteket.

```
PS C:\P> dir|measure-object -Property length -sum -Average -Maximum -  
Minimum  
Count : 9  
Average : 3666,66666666667  
Sum : 33000  
Maximum : 11459  
Minimum : 120  
Property : length  
PS C:\P>
```

# Processz kezelés

- Get-Process

- `ps | Where-Object {$_.handles -gt 500}`

- Processz befejezés

- `$p = Get-Process powershell`
  - `$p.kill` megadja a kill alakját!
  - `$p.kill()` # A PowerShellnek vége...
  - `ps|stop-process -whatif` #mi történne, ha ...
  - `ps|where-object {$_.name -like „s*” }` # fájlnev
  - `ps|where-object {$_.name -match „s*” }` # reguláris kifejezés illesztés

# Futtatás háttérben (PS 2.0)

- Start-Job –scriptblokk {start-sleep 10}
- Get-job # megkapjuk a futók listáját
- Remove-job –id szám #törlés
- Stop-job –id szám #megállítás
- Invoke-Command: Parancs futtatás helyi vagy távoli gépen
  - Enable-PSRemoting –force
    - Először engedélyezni kell a PS session kezelést, ha ezt használjuk!
  - Invoke parancsot vagy egy PSSession-ben, vagy direktben egy gépen (-computer) futtathatunk.

# Szerviz parancsok, indítás, megállítás...

```
PS C:\> Get-Command -Noun Service
```

CommandType	Name	Definition
-----	----	-----
Cmdlet	Get-Service	Get-Service [[-Name] <String[]>] [-Co...
Cmdlet	New-Service	New-Service [-Name] <String> [-Binary...
Cmdlet	Restart-Service	Restart-Service [-Name] <String[]> [-...
Cmdlet	Resume-Service	Resume-Service [-Name] <String[]> [-P...
Cmdlet	Set-Service	Set-Service [-Name] <String> [-Displa...
Cmdlet	Start-Service	Start-Service [-Name] <String[]> [-Pa...
Cmdlet	Stop-Service	Stop-Service [-Name] <String[]> [-For...
Cmdlet	Suspend-Service	Suspend-Service [-Name] <String[]> [-...

# Hitelesítés objektum

- Get-Credential

- \$c= Get-Credential alma

#alma felhasználónévhez

\$c=get-credential alma

write-host "Felhasználói név: "+ \$c.username

write-host "Felhasználói jelszó: "+ \$c.password

- \$c=Get-Credential

- Hitelesítést gyakran használunk a Get-WmiObject utasításnál
  - PL: Get-WmiObject Win32\_DiskDrive –computername szerver1 –credential \$c

# PowerShell bővítmények (SNAPIN)

- A PowerShell moduláris felépítésű
  - `gcm | Where-Object {$_.name -match "PSSnapin" } # Get, Add, Remove`
- `Get-PSSnapin` # megadja az aktuális modulok listáját
- `Add-PSSnapin Webadministration`
- `Remove-PSSnapin Webadministration`
- Előtte persze az IIS webadmin snapint installálni kell!

# Powershell modulok

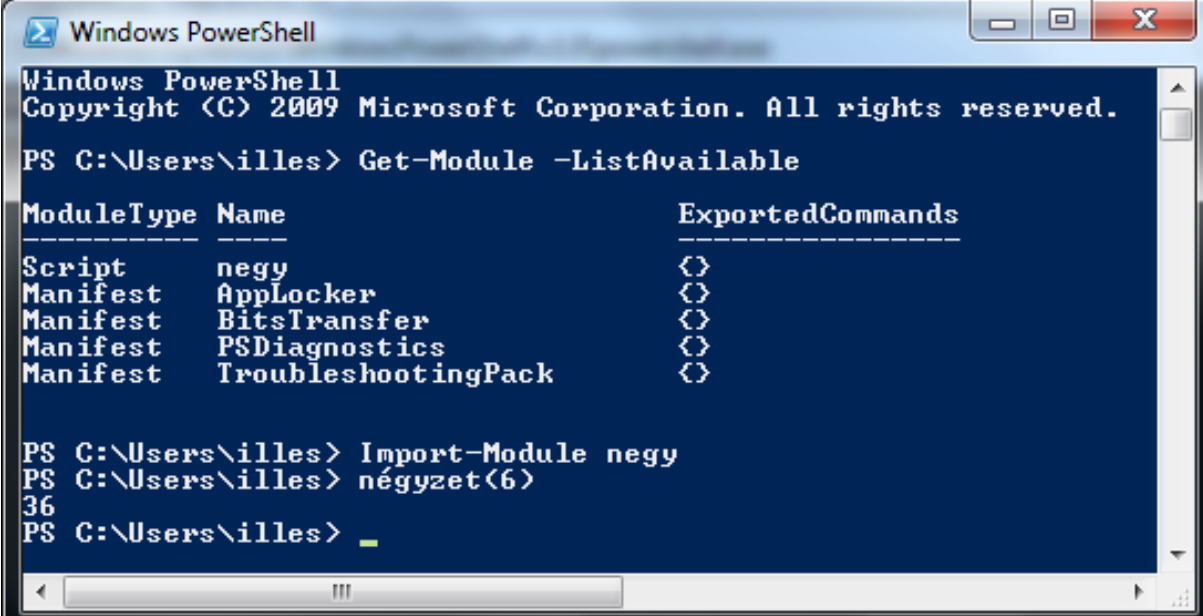
- A SNAPIN bővíthetőség egy bináris formátum, telepíteni kell először.
- A modul a PS 2.0-ban jelent meg, forráskódú
  - Get-Module – milyen modulok érhetők aktuálisan el
    - Get-Module –ListAvailable #összes elérhető modul listázás
    - Import-Module név # adott modul betöltése
  - Hasznos függvények, álnevek, változók definíciójának gyűjtőhelye.
  - \$env:PSModulePath

---

```
PS D:\home\ps> $env:PSModulePath  
C:\Users\illes\Documents\WindowsPowerShell\Modules;C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
```

# Saját modul (Script modul)

- 1. .psm1 kiterjesztés a saját modulnak.
- 2. A könyvtár neve azonos a fájl névvel, ezt helyezzük a „My Documents\WindowsPowerShell\Modules könyvtárba!
- 3. Import-Module negy.psm1



```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\illes> Get-Module -ListAvailable

ModuleType Name ExportedCommands
-----
Script negy {}
Manifest AppLocker {}
Manifest BitsTransfer {}
Manifest PSDiagnostics {}
Manifest TroubleshootingPack {}

PS C:\Users\illes> Import-Module negy
PS C:\Users\illes> négyzet<6>
36
PS C:\Users\illes> _
```



# WMI

- Windows Management Instrumentation
  - Infrastruktúra kezelés
- WMI Tools (külön kell installálni)
- WMI osztályok, névterek
  - `Get-WmiObject -Class __Namespace -Namespace root`
- `Get-WmiObject -list # wmi osztályok listája`
  - `Get-WmiObject Win32_Diskdrive`
  - `Get-WmiObject Win32_NetworkAdapter`
  - Stb...

# Active Directory (PS 1.0)

- ADSI – Active Directory Service Interface
- ADSI Providers
  - WinNT : NT4 PDC, BDC, és lokális felhasználók
  - LDAP : Win2000 óta az AD-k ezzel mennek
  - NDS : Novell Directory Services
  - Pl: \$a=[ADSI]"LDAP://dc=alma,dc=fa"
  - \$u=\$a.create(„organizationalunit”,„TesztUnit”)
  - \$u.setInfo();
- Stb....

# Active Directory (PS 2.0)

- Windows 2008 R2 serverhez megjelent ez a modul.
- Installálni kell először mint win2008 részt:
  - Active Directory for Windows PowerShell
- Ezután importáljuk:
  - Import-module activedirectory
- Kapunk sok új parancsot:
  - Get-command –module activedirectory

# További lehetőségek I.

- IIS szerver kezelése
  - IIS hozzáadja a kezelés parancsokat
  - [aspnet.inf.elte.hu](http://aspnet.inf.elte.hu)

The screenshot shows the Windows PowerShell ISE interface. The main editor window displays a script named `aspnet_user.ps1` with the following content:

```
139 # $acl=get-acl $útvtíval
140 #Add this access rule to the ACL
141 #SetAccessRule($rule)
142 #Write the changes to the object
143 #set-acl $útvtíval $acl
144 $útvtíval+" jogosítvány állítása!"
145 full_control($név) # így csak 1 paramétert szeret
146 # webapplication létrehozása
147 New-WebApplication -Site "Default Web Site" -Name $név -PhysicalPath $útvtíval
148 New-WebApplication -Site "Default Web Site" -Name $név+"_service" -PhysicalPath $útvtíval
149 }
150 else
151 {
152     "Adminisztrátor jog kell a futtatáshoz!"
153 }
154
```

The right-hand pane shows a list of commands available in the current context, including:

- Get-WebRequest
- Get-Website
- Get-WebsiteState
- Get-WebURL
- Get-WebVirtualDirectory
- Install-PswaWebApplication
- Invoke-WebRequest
- New-WebApplication
- New-WebAppPool
- New-WebBinding
- New-WebFtpSite
- New-WebGlobalModule
- New-WebHandler
- New-WebManagedModule
- New-WebServiceProxy
- New-Website
- New-WebVirtualDirectory
- Publish-BCWebContent
- Remove-WebApplication
- Remove-WebAppPool

The bottom console window shows the output of the command `get-help Uninstall-PswaWebApplication`:

```
PS C:\Users\illes\Documents> get-help Uninstall-PswaWebApplication

NAME
Uninstall-PswaWebApplication

SYNTAX
Uninstall-PswaWebApplication [-WebApplicationName <string>] [-WebSiteName <string>] [-WhatIf] [-Confirm] [-CommonParameters]
```

# További lehetőségek II.

- Exchange szerver kezelése
  - Exchange Management Shell-As administrator
  - Get-Excommand
  - New-ManagementRoleAssignment –Role „Mailbox Import Export” –user admin
  - New-MailboxImportRequest –mailbox usernév –FilePath [\\gépnév\share\user1.pst](#)
  - ...
- SQL szerver kezelése
  - Get-help sqlserver

# Csomagoljunk be ...

- A shell script csomagol mintájára!
- Használat, csomagolás:
  - csomagol.ps1 fájl1 fájl2 ... >csomi.ps1
- Kicsomagolás: csomi.ps1

```
# csomagoló: csomagol.ps1
# Használat: csomagol.ps1 fájl1
fájl2 ...
"#Csomagoljunk!"
foreach($i in $args)
{
    "echo $i"
    "@'"
    Get-Content $i # cat
    "'@ >$i"
    "echo '$i vége!'"
}
"#Csomagolás vége!"
```

# Kicsomagolunk ..

```
PS C:\d\home\ps> cp .\csomi.ps1 csomi
```

```
PS C:\d\home\ps> cd csomi
```

```
PS C:\d\home\ps\csomi> .\csomi.ps1
```

```
.\fradi.ps1
```

```
.\fradi.ps1 vége!
```

```
.\hajra.ps1
```

```
.\hajra.ps1 vége!
```

```
PS C:\d\home\ps\csomi> ls
```

Directory: C:\d\home\ps\csomi

Mode	LastWriteTime	Length	Name
----	-----	-----	----
-a----	2015. 11. 16. 8:52	526	csomi.ps1
-a----	2015. 11. 16. 8:53	120	fradi.ps1
-a----	2015. 11. 16. 8:53	76	hajra.ps1

# Köszönöm a figyelmet!

