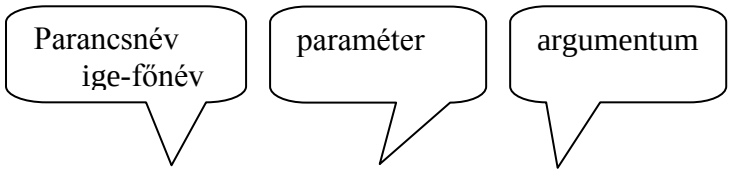
Powershell 10-11. gyakorlathoz…..
Bővebben

PowerShell (PS) = héj – Windows-os héjprogramozás eszköze
Letölthető, telepíthető –nem része az Windows XP -nek
NEM case-sensitive
Biztonsági szempontból nem lehet a ps1 kiterjesztéshez hozzárendelni a PowerShell-beli végrehajtást, el kell indítani!
Be kell állítani a futtatáshoz Set-ExecutionPolicy –ExecutionPolicy RemoteSigned
A részletesben említett feladatok, témánként könyvtárakba szervezve, script fájlokban megtalálhatóak. Futtatás módja: Lépjünk be az aktuális könyvtárba majd futtassuk a szkripteket a . .\szkriptnév.ps1 –gyel. (Ha a szkriptben több kipróbálandó rész van, az aktuálisat vegyük ki a megjegyzésből.)

3. *Help használata*

helye:./help/help.ps1 **indítása**: **. .**help.ps1

Get-Help // help használata (get-h + TAB) kiegészíti...
Get-Help about* //összefoglalók listája
Help about_escape_character
Get-Command //parancsok és más elemek (providerek, függvények)
Get-Command -? //Get-Command-ról help
Get-Help Get-Command –full (/Detailed/Examples) – paraméterek
Hívjuk fel a figyelmet a PS utasítások szerkezetére



Pl. Get-Command –commandtype function //függvények listája
Get-Command –Commandtype Cmdlet //a cmdlet-ek listája
gcm // Get-Command aliasa
Get-Alias //alias nevek
Set-Alias „ujnev” parancs //új aliast adhatunk

4. *Néhány egyszerűbb parancs*

helye: /parancsok/parancsok1.ps1 **indítása**: **. .**parancsok.ps1

Write-Host „Üdvözlés” –ForeGroundColor Blue
Clear-Host vagy Clear // képernyőtörlés
Get-Date //dátum-idő kiírása
Get-Content „fájlnév” // fájl tartalmának kilistázása
Get-Childitem //aktuális könyvtár tartalma
Get-Childitem „c:\konyvtar” // adott konyvtár tartalma
Get-ChildItem –recurse // alkönyvtár tartalmakkal együtt
Get-ChildItem –Filter *.txt // csak a txt fájlokat listázza
Copy-Item „fájlt” „fájlba” //másolás
Copy-Item „fájlt” ”fájlba” –Confirm //csak akkor hajtsa végre, ha
Copy-Item „fájlt” „fájlba” –Whatif //nem hajtja végre, csak megadja mit tenne...
Hasonlóan: Rename-Item, Remove-Item,

Common parameter: - minden cmdlet mellett ugyanúgy viselkednek

Confirm, whatif, debug erroraction, errorvariable,

New-Item –Path utvonal –ItemType Directory //új konyvtar létrehozása
New-Item –Path utvonal –ItemType File //új fájl létrehozása
Get-Location //path kiolvasása
Set-Location –path „útvonal” //könyvtárra váltás - cd
Push-Location útvonal //verembe teszi az aktuális könyvtárat és rááll az „útvonal”-ra
Pop-Location //a veremből kiolvassa az útvonalat és visszaállítja...
Get-Process //gps
Invoke-Item c:\Windows\Notepad.exe // elindítja az alkalmazást -ii
Stop-Process ID // process lelövése .- kill

5. *Csővezeték – objektumok*

helye: /csovezetek/csovezetek.ps1 **indítása:** **. .**csovezetek.ps1

Get-ChildItem | Out-File fájlnév //csővezeték
Get-Date | Get-Member //objektumok feltérképezése, pl. Dátum objektumra – NEM SZTRING-et ad át!!
pl. (Get-Date).year // objektum tagadatának elérése
Get-ChildItem „fájlnév” | Get-Member // egy fájl adatait tartalmazó objektum tagjai
(Get-ChildItem „fájlnév”).FullName //fájl neve útvonallal
Get-ChildItem | Where {$_.Length –gt 100} //100-nál hosszabb fájlokat listázza ki $_ a csővezetékből érkező objektum
Get-ChildItem | Where {$_.Extension –eq „.txt”} //txt fájlokat listázza
Get-Childitem| Sort-Object –Property LastTimeWrite –Descending //a könyvtárat csökkenő sorrendben utolsó írás ideje szerint
Get-Process // futó processzeket listázza
Get-Process | Format-List –Property Name, Id // a processzeket listázza, csak az ID-t és nevet
Get-Process |Format-table –AutoSize //táblázatos formában írja ki,
Get-Process| Where-Object {$_.ProcessName –lt „Nev”} //szűrés, $_ csővezetéken érkező objektum, operátorok!!
Get-Process| Sort-Object –Property CPU| select-Object –Property Name, Cpu –Last 5 | Format-Table –AutoSize //a processzeket állítsuk sorba a CPU szerint, szűrjük le a name és cpu tulajdonságra és tábla formába írjuk ki.

6. *Változók – bármit tartalmazhatnak, objektumot is*

helye: /valtozok/valtozok.ps1 **indítása:** **. .**valtozok.ps1
Változók azonosítója $-ralkezdődik

$HOME // user home könyvtára c:\document and setting\ user,
$PSHOME //Powershell home könyvtára c:\Windows\system32\WindowsPowerShell\v1.0
Set-Location $HOME //
Saját változók:
Get-Variable // változók listája
Set-Variable –Name x –Value 2 // x változó, 2-s értékkel
$x=3 // x változó 3-s értékkel
$x //kiírja az értéket
$sz=”hello”+”vilag” //” a string, + konkatenáció
„hello$x” // „ „között a változó tartalma kerül be a szövegbe – hello3
’hello$x’ // ’’-között a szöveg, a változó azonosítóját tárolja --- hello$x
$ma=Get-Date // objektumokat is tárolhat!

$ma.Year // az objektum Year adattagja
Remove-Variable –Name azonosító // azonosító eltávolítása
$tombbe=Get-Content „szoveg.txt”
$tombe // a teljes fájl tartalmát beolvassa
$tombbe |get-member // a tömbbel kapcsolatos elérhető adatok, metódusok
Get-psprovider // beépített szolgáltatók listája – drive, alias, registry....

---

7. *If, for, foreach* (C#-hoz hasonló szintaxis)

helye: /if_for/

-Fájl meglétének ellenőrzése **fájl:** if.ps1

```
if (Test-Path „fájlnév”) {„Van”;} else {„Nincs”;}
```

-Faktoriális számítás **fájl:** fakt.ps1

```
$er=1
for($i=1;$i –lt 6;$i++){$er=$er*$i;} //-lt logikai operátor, for és ( közötti space hibás
$er
```

-tömbben keresünk , reguláris kifejezés

```
$szavak="alma", "korte", "banan", "szilva", "szolo","barack"      fájl: while_if.ps1
$i=0;
while ($i -lt $szavak.length) {
        if ($szavak[$i] -match "^B"){
                write-host ($szavak[$i], " B-vel kezdodik")
        }
        $i++
 }
```

-könyvtárban lévő fájlok kiterjesztésének kiiratása **fájl:** konyvtar.ps1

```
foreach($elem in Get-ChildItem){$elem.Extension;} //$elem egy fájl-objektum
```

- asszociatív tömb használata **fájl:**foreach.ps1

```
write-host "Szotar kezelese"
$szotar=@{kutya="dog";macska="cat";eger="mouse"}
write-host "Szavak:"
foreach ($magyar in $szotar.Keys){
      write-host ($magyar," ",$szotar[$magyar])
}
```

- -többszörös elágazás **fájl:**foreach.ps1

```
$szam=2
switch ($szam){
1 {write "megerett a meggy"; break}
2 {write "csipkebokor vesszo";break }
3 {write "Te legy az en parom";break }
default {write "Sok";}
}
```

8. *Függvények* (csak a PS futási ideje alatt él)

helye: /fuggvenyek/

listaz1 függvény kilistázza az aktuális könyvtár tartalmát **fájl:** listaz1.ps1

```
function listaz1 {
```

```
 get-childitem;
} //függvény deklaráció
```
listaz1 // függvény hívása

---

listaz2 függvény kilistázza a paraméterben megadott mappa tartalmát

fájl: listaz2.ps1

```
function listaz2 {
  param($mappa);
  get-childitem –path $mappa;
} //paraméter használata
```
listaz2 c:\windows

---

fakt függvény kiszámítja a paraméterben megadott szám faktoriálisát

fájl: fakt.ps1

```
function fakt{
  param($szam);
  $eredmeny=1;
  for($i=2;$i –lt $szam+1;$i++){
    $eredemeny*=$i;
  }
 $eredemeny
}
```
fakt 5

$eredmeny //nincs, hatóköre a függvényre vonatkozik

---

fakt2 függvény kiszámítja a paraméterben megadott szám faktoriálisát

fájl: fakt2.ps1

```
function fakt2{
  param($szam);
  $Global:eredmeny=1;
  for($i=2;$i –lt $szam+1;$i++){
    $Global:eredmeny*=$i;
  }
 $Global:eredmeny
}
```
fakt 5

$eredmeny //értéke 720, most globális

---

letrehoz függvény az első paramétert, mint fájlt, a másodikat mint útvonalat tekinti. A fájlban lévő azonosítókkal létrehoz az útvonalon mappákat.

fájl: letrehoz.ps1

```
function letrehoz {
 param($fajl,$utvonal);
 $mappak=Get-Content $fajl;
 foreach ($mappa in $mappak){
   New-Item –path „$utvonal\$mappa” –ItemType Directory
 }
 ;} // a mappa és almappái törlése
```
letrehoz mappak.txt c:\torles

Batch file – ps1 kiterjesztéssel

helye: /batch/

Figyeljünk arra, hogy engedélyezve legyen a futtatás
Set-ExecutionPolicy –ExecutionPolicy RemoteSigned //ezzel állítsuk be
Hozzunk létre ps1 kiterjesztésű fájlokat pl. az edit szerkesztő segítségével.

Kiir egy szöveget a default kimenetre **fájl:** kiir.ps1

"Ezt PS batch segítségével írtam ki"
Futtassuk azt: .\fájlnév.ps1 // kötelező kiírni a „ .\"-t

---

meghívás után kiirja a dátumat **fájl:** datum.ps1

```
$datum="Datum: "+(date.Year)+". "+(date.Month)+". „+(date.Day)+".";
//héjhatókör
$datum;
```

Futtasuk .\szkript.ps1
$datum //üres!! hatóköre csak a szkriptre vonatkozik
Futtassuk . .\fájlnév.ps1 //dot sourcing – héjhatókör beemelése
$datum //tartalmazza a dátumot

---

könyvtár és tartalmának másolása paraméterrel megadva a forrás és cél könyvtárat **fájl:** masol.ps1

```
param($honnan,$hova);
if (!test-path –path $hova)) {new-item –path $hova –type directory;};
copy-item –path $honnan –destination $hova -recurse;
```

Microsoft gyártású alkalmazások processzeinek listája **fájl:** processz.ps1

```
get-process|where {$_.Company –eq „Microsoft Corporation"}|format-table –
property name,company
```

Prím számot olvastunk-e be (keresés tömbben) **fájl:** kereses.ps1

```
write-host "Kereses tombben"
$x=@(2, 3, 5, 7, 9, 11, 13, 17, 19, 23, 29)
write-host "A tomb elemei:"
foreach ($elem in $x)
{Write-host "$elem " -nonewline}
Write-host ""
[int]$a=read-host "Kerem az egesz szamot"
$i=0
while (($i -lt $x.length) -and ($x[$i] -ne $a))
{$i++}
if ($i -lt $x.length)
{"Benne van."}
else{"Nincs benne."}
```

Text fájl mérete **fájl:** megszamol.ps1

```
param ($fajl)
$sorok=0
$sorok=get-content $fajl -filter "*.txt"|measure-object -line
write-host ($sorok.Lines," sorbol all")
$szavak=get-content $fajl -filter "*.txt"|measure-object -word
write-host ($szavak.Words," szobol all")
```

```
$betuk=get-content $fajl -filter "*.txt"|measure-object -char
write-host ($betuk.Characters," betubol all")
```

Könyvtár mérete **fájl:** megszamol2.ps1

```
$hossz=(get-childitem|measure-object -property length).Count
write-host ("konyvtarban  ",$hossz," darab bejegyzes van")
$szumma=0
$szamolandok= Get-childitem|where-object {!$_.PsIsContainer}
foreach($egy in $szamolandok){
$szumma+=($egy|measure-object -property length -minimum).Minimum
}
write-host ("konyvtarban  ",$szumma," byte van")
```

Másodfoku egyenlet megoldasa **fájl:**masodfoku.ps1

```
write-host "Masodfoku egyenlet megoldasa axx+bx+c=0\n\n"
[float]$a=read-host "K,rem a m sodfoku tag egyutthatojat:"
[float]$b=read-host  "K,rem az els‹foku tag egyutthatojat:"
[float]$c=read-host "K,rem a konstans tagot:"
$d=($b*$b)-4*$c*$a
if ($a -eq 0){Write_host "Nem m sodfoku ";}
else { if ($d -lt 0){ Write-Host "Nincs valos gyok";}
        else { if ($d -eq 0){ Write-Host ("Egy gyok ",(-$b/(2*$a)));}
                else { $x1=(-$b-[System.Math]::Sqrt($d))/(2*$a)
                        $x2=(-$b+[System.Math]::Sqrt($d))/(2*$a)
                        Write-Host ("Egyik gyok: ",$x1," masik gyok: ",$x2)}

        }
}
```

Prímtényezőkre bontás **fájl:**primtenyezore.ps1

```
[int]$szam=read-host "Kerem az egesz szamot, amit primtenyezokre bontok"
$primt=@()
$i=2
while ($szam -gt 1){
        if (($szam % $i) -eq 0){
                $szam=$szam / $i
                $primt+=$i     }
        else {$i++}
}
foreach ($elem in $primt){$elem}
```

Prímosztók megadása **fájl:**prime.ps1

```
function prime{         {
param($szam)
$gyoke=[System.Math]::sqrt($szam)
$i=2
while (($i -le $gyoke) -and (($szam % $i) -ne 0)){$i++       }
return ($i -gt $gyoke)
}
```

```
[int]$szam=read-host "Adjon meg egy egeszet"
for($j=2;$j -le $szam;$j++){
        if ((($szam % $j) -eq 0) -and (prime($j))){
        write-host $j}
}
```

---

időzítés, paraméterben megadott másodpercenként kiír **fájl**:idozit.ps1

```
param([int]$mennyi)
$kezd=Get-Date
while (1 -lt 2){

 $most=Get-Date
 $kul=$most-$kezd
 if ($kul.Seconds -gt $mennyi){
      write-host ("Eltelt...",$mennyi," masodperc"); $kezd=$most}
}
```

---

adott idő múlva elindít egy alkalmazást **fájl:**idozit2.ps1

```
param([int]$mennyi)
$kezd=Get-Date
do
        {
        $most=Get-Date
        }
while (($most-$kezd).Seconds -lt $mennyi)
write-host ("Eltelt",$mennyi,"masodperc.")
invoke-item "c:\windows\notepad.exe"
```

---

adott idő múlva elindít egy alkalmazást **fájl:**idozit3.ps1

```
param([int]$mennyi)
while (1 -lt 2){
  sleep $mennyi
  write-host ("Eltelt...",$mennyi," masodperc");
}
```

Paraméterben kapott tömb összegzése **fájl**:valtozopar.ps1

```
$szumma=0
if (("-?" -eq $args[0]) -or ($args.length -ne 1)){
      write "Ez a szkript osszeadja a parameterul kapott szamokat pl. .\valtozo 2,4,5";}
else {  foreach($p in $args[0]){$szumma+=$p}
      write $szumma
}
```

9. *Hibakezelés*

helye: /hibakezeles/

A listaz2_hiba egy paraméterként megadott mappa tartalmát listázná, ellenőrizve a mappa meglétét (függvények közül az egyiknek a javítása)
CommonParameter segítségével kezeli a hibákat:
function listaz2_hiba{ fájl: listaz2_hiba.ps1

```
 param($mappa);
Get-ChildItem $mappa –ErrorAction SilentlyContinue –ErrorVariable Err;
if ($Err){write-Host „Nem tudom kilistázni a könyvtár tartalmat”;}
}
```
Próbáljuk ki létező és nem létező mappával
A letrehoz2_hiba a paraméterként megadott file-ban lévő azonosítók alapján létrehozza a könyvtárszerkezetet. Rossz fájl vagy mappa esetén hibát jelez.
trap segítségével kezeli a hibákat (csak megszakító jellegű hibákat fog el, ha nem az a hiba, akkor -ErrorAction-t Stop-ra kell állítani):

fájl: letrehoz_hiba.ps1

```
function letrehoz_hiba{
        param($fajl,$utvonal)
        trap [Exception]{
                write-host „Nincs ilyen fajl vagy mappa”
                break           //break, continue vagy return állhat itt
        }
        $mappak=Get-Content $fajl –ErrorAction Stop; //megszakító jellegű hiba lép fel
        Foreach($mappa in $mappak){
                New-Item –path „$utvonalak\$mappa” –ItemType Directory –errorAction Stop
}
}
letrehoz_hiba //nincs paraméter, ezért a Get-Content  kiolvasásánál megszakítás
letrehoz_hiba mappa2.txt f: //nincs ilyen fajl és mappa sem, ezért hibát jelez
letrehoz_hiba mappa.txt c:\ //ez jó, ha a gyökérbe még nem hoztuk létre ezeket a
                                mappákat ;))
trap{continue} letrehoz_hiba mappa2.txt c:\ //a külső trap megfogja a belső hibát…
```

A fakt_hiba a paraméterben megadott szám alapján faktoriálist számít.
throw segítségével kezeli a hibákat

fájl: fakt_hiba.ps1

```
        function fakt_hiba{
param(int)$szam=$(throw write „nem adott parametert”));
if ($szam –lt 0){throw write „nem értelmezett, kisebb nullánál”}
$eredmeny=1;
for($i=2;$i –lt $szam+1;$i++){
 $eredmeny*=$i
}
$eredmeny
}
fakt_hiba  //nem adunk paramétert
fakt_hiba -1 //nem jó paraméter
fakt_hiba 5 //rendben számol
```

10. *Registry használata*
helye: /registry/

A szkript megjegyzi az aktuális könyvtárt az ide azonosítóval
set-location ide: //visszaváltás

fájl: registry.ps1

Get-PsDrive //megmutatja a drive-okat – hkcu(registry-current-user) és hklm(local-machine)

---

set-location hklm: //HKEY_CURRENT_USER – re áll
Get-ChildItem // kiolvassa a kulcsokat
Nézzük meg a **Start/futtatas/regedit-ben ugyanezt!**
set-location-nel mozogjunk a registry-ben
---
set-location hklm:/software/Microsoft/Windows/currentversion
//REGEDIT-ben ellenőrizzük
get-childitem
get-itemproperty Run //ezek indulnak a Windows indulásával
Set-Location hkcu:/software //current-user software bejegyzés
Get-Childitem //
New-item –path hkcu:\software\toroldki //új registry érték beírása
Get-Childitem ////REGEDIT-ben ellenőrizzük
new-ItemProperty –name torol –value 3 –path hkcu:/software/toroldki
// új kulcsot és értéket jegyez be a registry-be
---
remove-itemproperty –name torol –path hkcu:/software/toroldki //törlése
---
new-ItemProperty –name notepad –value c:\windows\notepad.exe –path hklm:/software/microsoft/windows/currentversion/Run
VIGYÁZAT – minden Windows indítás után elindul a notepad!!!

11. WMI használata (Windows Management Instrumentation)

helye: \wmi\

Get-WMIObject –list //kilistázza a WMI obejktumokat **fájl:** wmi.ps1

Get-WmiObject win32_bios //bios információk

Get-wmiobject –class win32_OperatingSystem // OS információkat ad vissza
---
Get-WmiObject –class Win32_OperatingSystem | get-member //többi információ
(Get-WmiObject –class Win32_OperatingSystem).WindowsDirectory //hol a windows könyvtár
---
Get-wmiobject –class win32_Logicaldisk //disk-ekről információ

Get-wmiobject –class win32_Logicaldisk|select –property deviceId,Size,freespace| format-table –autosize // a létező disk-ek megnevezését,méretét és szabad területét listázza
---
Get-Wmiobject –class win32_LocalTime //rendszeridő

12. ComObjektumok

helye: \ComObj\

fájl: comobj.ps1

Asztalra ikon készítése
$wshshell=new-object –comobject wscript.shell
$lnk=$wshshell.createshortcut(„$home\asztal\Sajat.LNK”) //shortcut referencia készítése
$lnk.targetpath=$home // az útvonal beállítása

$lnk.save() //

$ie=new-object –comobject internetexplorer.application //elindít egy IE processt
Get-process //látszik az elindított explorer process
$ie|get-member //milyen propertyk, metódusok

$ie.visible=$true //megjeleníti az IE ablakot

$ie.gohome() //megjeleniti a home lapot

$ie.navigate(„szamalap.inf.elte.hu”) //navigál az url-re
$ie.quit()