|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | **Cues**  NoteGem Horizontal Line    what are cmdlets?    what do ps providers do?      what is powershell    what is its linux equivalent?  what does ISE stand for?      how to ask for help per command?  how to check version?  how to get list of updates?        what are the sections of the ISE?              Differences between cmdlets and commands?            how to output a file's contents?          how to check folder/file existence?  how to rename a file?  how to add one day to system date?      how to get system date?        how to set system time?        how to create a folder?    how to create a file?  how to copy a single file/folder?  how to recursively copy a single file/folder?  how to delete a single folder?  how to recusively delete a single file/folder?  how to delete a single file?  how to move a folder or file?  how to rename an item?  how to create a text file?  how to read a text file?  how to create an xml file?          how to read an xml or html file?  how to create a csv or html file?    how to read a csv file?  how to erase a file's contents?  how to append content to a file?  how to get object from a sorted list?  how to get properties of a passed object?      how to compare two objects?        how to format output at list of properties?          how to format output as table with one property per object?        how to select objects having certain property values from collection passed to it?        how to get items or child items in one+ locations?    how to use ForEach?    how to suspend activity in script for a particular period of time?            how to read from console?        how to select an object or its properties?        how to sot an object by its properties?        how to write warning messages?          how to write customized messages?      how to perform default action on specified item?      how to perform a command or expression on local PC?        how to measure time taken by a script or command?        how to run a command from current session which are already run?        how to add commands to current history?        how to get commands run in current session?    how to get current culture set in Windows?      what do cmdlets do?    how is powershell task oriented?            what are some features of powershell?          how to write a powershell variable?        what is $$?  what is $??        what is $^?  what is $\_?              what is $ARGS?    what is $CONSOLEFILENAME?  what is $ERROR?  what is $EVENT?  what is $EVENTARGS?    what is $EVENTSUBSCRIBER?  what is $EXECUTIONCONTEXT?    what is $FALSE?  what is $FOREACH?        what is $HOME?  what is $HOST?  what is $INPUT?  what is $LASTEXITCODE?  what is $MATCHES?  what is $MYINVOCATION?        what is $TRUE?  what is $THIS?  what is $STACKTRACE?  what is $SHELLID?  what is $SENDER?  what is $NESTEDPROMPTLEVEL?  what is $NULL?  what is $PID?  what is $PROFILE?  what is $PSCMDLET?  what is $PSCOMMANDPATH?  what is $PSCULTURE?  what is $PSDEBUGCONTEXT?  what is $PSHOME?  what is $PSITEM?        what is $PSSCRIPTROOT?  what is $PSSENDERINFO?    what is $PSUICULTURE?  what is $PSVERSIONTABLE?        what are the different powershell operation groups?        what are the different powershell comparison operators?                            how to loop in powershell?                                how to write conditions in powershell?                        how to write arrays in powershell?          how to write a hash table in powershell?                    what does a backtick do?              what do the different brackets do () {} []?                                how to use an alias?    how to get all aliases present in current session of Powershell?      how to match begin of line?  how to match end of line?  how to match any single char?  how to match [single char]?  how to match single char un- [ ]d?  how to match begin of string  how to match end of string?  how to match end of string -terminator?  how to match 0+ occurrences of preceding expression?  how to match 1+ occurrences of preceding expression?  how to match 0 or 1 occurrences of preceding expression?  how to match n occurrences of preceding expression?  how to match x+ occurrences of preceding expression?  how to match n-m occurrences of preceding expression?    how to match a or b?  how to group regular expressions and remember matched text?  how to group regex without remembering matched text?  how to match independent pattern without back tracing?  how to match word chars?  how to match non-word chars?  how to match whitespace?  how to match non-whitespace?  how to match digits?  how to match non-digits?  how to match begin of string  how to match point where last match finished?  how to match word boundaries when outside brackets or match backspace when inside brackets?    how to match non-wod boundaries?  how to match new lines?  how to escape all chars up to \E?  how to end quotes begun with \Q | **Notes**  NoteGem Horizontal Line    powershell commands = **cmdlets**    powershell providers let you access data stores such as registry and cert store    command-line shell & scripting language for system admin    in-linux equivalent = **bash scripting**  **windows powershell ise** = integrated scripting environment     * host app for windows powershell commands      * **Get-Help** = gives explanation of command and its parameter * **$PSVersionTable** = check version * **Get-Hotfix** = get list of updates -id kb2741530      * ISE has three sections:      * powershell console * scripting file * command module     cmdlets v. commands     * .net framework class objects, not just standalone execs * easily constructed from as few as a dozen lines of code * parsing, error presentation, output formatting not handled by them   + done by wps runtime * record based by processing single object at a time     advanced cmdlets    **retrieve item** = will output file contents     * Get-Content 'D:\temp/text.txt' * Get-Content 'D:\temp/text.txt'.length = size of file contents     **check folder/file existence** = Test-Path 'D:\temp/text.txt'  **rename file** = Rename-Item 'D:\temp/text.txt' 'D:\temp/text1.txt'  **set system date** = add one day     * Set-date -date (Get-Date).add Days(1)/     **get system date** = Get-Date     * Get-Date -DisplayHint Time (displays time only)     **set system time** = $timeToAdd = New-TimeSpan -Minutes -60     * set-date -adjust $timeToAdd     **create folder** = New-Item -Path 'D:\temp/testFolder' -ItemType Directory  **create file** = New-Item -Path 'D:\temp/text.txt' -ItemType File  **copy file/folder single** = Copy-Item 'D:\temp/text.txt' newpath  **copy file/folder recursively** = Copy-Item -filter \*.txt -Path 'D:\temp/text.txt' -Recurse -Destination 'D:\temp/testFolder1'  **delete folder single** = Remove-Item 'D:\temp/TestFolder1'  **delete folder/file recursively** = Remove-Item 'D:\temp/TestFolder1' -Recurse  **delete file single** = Remove-Item 'D:\temp/text.txt'  **move folder/file** = Move-Item oldpath newpath  **rename item** = Rename-Item oldname newname  **create text file =** New-Item path contents  **read text file** = Get-Content path  **create xml file** = New-Item path -ItemType File     * Set-Content path '<title>Welcome</title>' * Get-Content path     r**ead xml/html files** = Get-Content path  **create csv/html files** = New-Item path -ItemType File     * Set-Content path 'John, Jane, Mary'     **read csv file** = Get-Content path  **erase contents of file** = Clear-Content path  **append content to file** = Add-Content path 'content to add'  **get object from sorted list** = Get-Unique = $list | sort | Get-Unique  **get properties of passed object** = measure-object     * get-content path | measure-object -character -line -word     **compare two objects** = Compare-Object -ReferenceObject $(Get-Content path)     * Compare-Object -DifferenceObject $(Get-Content path)     **format output at list of properties** =     * $A = Get-ChildItem path * Format-List -InputObject $A * Get-Service | Format-List     **format output as table with one property per object** =     * $A = Get-ChildItem path * Format-Wide -InputObject $A * Format-Wide -InputObject $A -Property -Property Length     **select objects having certain property values from collection passed to it** =     * Get-Object | Where-Object {$\_.Status -eq 'Stopped'} * Get-Object | Where-Object {$\_.ProcessName -Match '^p.\*'}     **get items or child items in one+ location** = $A = Get-ChildItem path    **ForEach** = used to perform operations on each object of a collection  1000, 2000, 3000 | ForEach-Object -Process {$\_/1000}  **Start-Sleep** = suspends activity in script or session for a particular period of time     * Start-Sleep -s 15 (-s = seconds, -m = milliseconds)     **Read-Host** = used to read from console     * $choice = Read-Host "Please put your choice"     **Select-Object** = used to select object or its properties     * Get-Process | Select-Object -Property ProcessName, Id, WS -Last 5 * "a","b","c","a","b" | Select-Object -Unique     **Sort-Object** = used to sort object by properties     * Get-Process |Sort-Object -Property WS | Select-Object -Last 5 * "d", "e", "c", "b", "a" | Sort-Object     **Write-Warning** = used to write warning messages     * Write-Warning "test warning"     **Write-Host** = used to write customized messages     * Write-Host(2, 4, 6, 8, 10, 12) -Separator ", -> " -ForegroundColor DarkGreen -BackgroundColor White     **Invoke-Item** = used to perform default action on specified item     * Invoke-Item path (opens item, open is default)     **Invoke-Expression** = performs command or expression on local PC     * $Command = 'Get-Process' * Invoke-Expression $Command     **Measure-Command** = measures time taken by script or command     * Measure-Command {Get-Event-Log "Windows Powershell"}     **Invoke-History** = runs command from current session which are already run     * Invoke-History * Measure-Command {Get-Event-Log "Windows Powershell"}     **Add-History** = adds commands in current history     * Get-History -count 5 | Add-History * Get-History     **Get-History** = gets commands run in current session     * Get-History     **Get-Culture** = get current culture set in Windows     * Get-Culture     **cmdlets** = perform common system admin tasks like managing registry, services, processes, event logs, using WMI  **task oriented** = Powershell is task based and provides support for existing scripts and command-line tools  **consistent design** = as cmdlets, system data stores use common syntax and have common naming conventions, data sharing easy  output from one cmdlet can be pipelined to another without manipulation  simple to use  **object based** = objects can be sent to other tools/databases directly  **extensible interface** = customizable, can build custom tools and utils using powershell to administer software  **powershell variables** = named objects     * start with $ and contain \_ and alphanumeric characters     create a variable by typing valid variable name     * $location = Get-Location     **$$** = represents the last token in last line received by session  **$?** = represents execution status of last operation     * contains true if last operation succeeded, false if failed     **$^** = represents first token in last line received by session  **$\_** = same as $PSItem     * contains current object in pipeline object * can use in commands that perform action on every object or selected objects in pipeline     **$ARGS** = array of undeclared parameters/values passed to function/script/block  **$CONSOLEFILENAME** = path of console file most recently used in session  **$ERROR** = array of error objects that represent most recent errors  **$EVENT** = object representing event being processed  **$EVENTARGS** = PSEventArgs object that represents event (first) argument that derives from EventArgs of event being processed  **$EVENTSUBSCRIBER** = PSEventSubscriber object that represents event subscriber of event being processed  **$EXECUTIONCONTEXT** = Engine Intrinsics Object representing execution context of Powershell host  **$FALSE** = represents false, can use instead of string "false"  **$FOREACH** = enumerator, not resulting values, of ForEach loop     * can use props/methods of enumerators on value of $ForEach variable     **$HOME** = full path of user's home directory  **$HOST** = represents current host app for Powershell  **$INPUT** = enumerator that enumerates all input passed to function  **$LASTEXITCODE** = exit code of last windows-based program run  **$MATCHES** = works with -match and -nomatch operators  **$MYINVOCATION** = populated only for scripts, blocks, functions     * PSScriptRoot and PSCommandPath props contain info about invoker or calling script, not current script     **$TRUE** = represents true, can be used instead of string "true"  **$THIS** = object being extended  **$STACKTRACE** = stack trace for most recent error  **$SHELLID** = identifier of current shell  **$SENDER** = object that generated this event  **$NESTEDPROMPTLEVEL** = represents current prompt level  **$NULL** = auto-variable containing null or empty value  **$PID** = process identifier of process hosting current Powershell session  **$PROFILE** = full path of Powershell profile for current user & current host app  **$PSCMDLET** = object representing cmdlet or advanced function being run  **$PSCOMMANDPATH** = full path & file name of script being run  **$PSCULTURE** = name of culture currently in use in OS  **$PSDEBUGCONTEXT** = while debugging it contains info about debugging environment  **$PSHOME** = full path of install directory for Powershell  **$PSITEM** = same as $\_.     * contains current object in pipeline object     **$PSSCRIPTROOT** = directory from which script is being run  **$PSSENDERINFO** = represents information about user who started powershell session  **$PSUICULTURE** = name of UI culture currently in use in ONCE  **$PSVERSIONTABLE** = read only  **Operations: Groups**     * **Arithmetic** (used like algebra, +-\*/%; modulus returns remainder) * **Assignment** = += -= * **Comparison** =      * **eq** = equals * **ne** = not equals * **gt** = greater than * **ge** = greater than or equal to * **lt** = less than * **le** = less than or equal to      * **Logical** = and or not * **redirectional** = assigns output to be printed into redirected file or output device * **split & join** * **type** * **unary**     **looping**     * $array = ("item1", "item2", "item3") * for($i=0;$i -lt $array.length; $i++){$array[$i]} * ForEach($element in $array){$element} * $counter = 0      * while($counter -lt $array.length){$array[$counter];$counter+=1;}      * $counter = 0      * Do{$array[$counter];$counter+=1;}      * While($counter -lt $array.length)     **conditions**     * if(boolean expression){//statements to execute if true} * if(boolean expression){//statements to execute if true}else{//if false} * can nest conditional statements * switch(test value) {   condition {action}  break; // optional  condition {action}  break; // optional  }    **arrays**     * $A = 1, 2, 3, 4 * $A = 1..4 * $myList = @(0..4)     **hash tables**     * $hash = @{ID=1;shape="square";color="blue"} * $hash = @{} * ordered dictionaries created using similar syntax   + maintain order entered unlike hash tables * $hash = [$ordered]@{} * $hash["ID"] * $hash.keys/values * $hash.Number, Count, Add, Remove, GetEnumerator     **multiple line commands backtick `**     * **`n** = new line * **`t** = sentences     **brackets**     * ()      * pass arguments * create array * resolve ambiguity * enclose multiple sets of instructions      * {}      * enclose statements * block commands      * []      * access to array * access to hash tables * filter using regex     **alias**     * New-Alias -Name help -Value Get-Help * help Get-WmiObject -Detailed * **Get-Alias** = get all alias present in current session of Powershell     **regex**    **^** = matches beginning of line  **$** = matches end of line  **.** = matches any single character except newline m matches newline as well  **[…]** = matches single character in brackets  **[?...]** = matches any single character NOT in brackets  **\A** = beginning of entire string  **\z** = end of entire string  **\Z** = end of entire string except allowable final line terminator  **re\*** = matches 0+ occurrences of preceding expression    **re+** = matches one+ of previous thing    **re ?** = matches 0 or 1 occurrence of preceding expression    **re{n}** = matches n number of occurrences of preceding expression  **re{x, }** = matches x+ number of occurrences of preceding expression  **re{n,m}** = matches at least n and at most m occurrences of preceding expression    **a|b** = matches a or b  **(re)** = groups regular expressions and remembers matched text    **(?:re)** = groups regex powershell without remembering matched text    **(?>re)** = matches independent pattern without back tracing    **\w** = matches word characters  **\W** = matches non-word characters  **\s** = matches whitespace, equiavlent to [\t \n \r \f]  **\S** = matches non-whitespace  **\d** = matches digits, equivalent to 0-9  **\D** = matches non-digits  **\A** = matches beginning of string  **\G** = matches point where last match finished    **\n** = back-reference to capture group # "n"    **\b** = matches word boundaries when outside brackets  matches backspace when inside brackets        **\B** = matches non-word boundaries    **\n \t** etc = matches new lines, carriage returns, tabs, etc  **\Q** = escape (quote) all characters up to \E  **\E** = ends quoting begun with \Q | |
| |  | | --- | | Summary | |