PowerShell - Intro

CHRIS THOMAS

DESKTOP ENGINEER - INGHAM ISD

CTHOMAS@INGHAMISD.ORG

@AUTOMATEMYSTUFF

Ingham Intermediate School District
A Regional Educational Service Agency

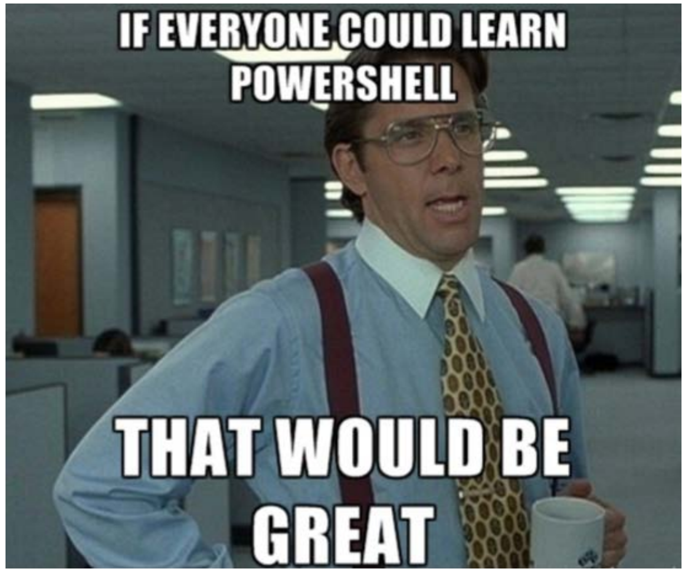
Session Description

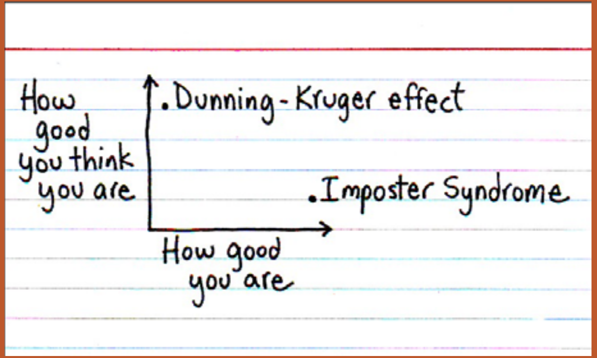
An introductory walkthrough of Windows PowerShell, the ISE and a small portion of the many cmdlets at your disposal. Learn how to use simple commands to perform powerful administration. See why you'll want to transform commands you're likely familiar with into PowerShell cmdlets that'll give you more control and data for your efforts.

My hope is that everyone can leave here feeling more comfortable around the language within PowerShell and start to think of ways that PowerShell can help them with their day-to-day work.

I challenge everyone to replace command prompt with PowerShell and not to rely on aliases in the beginning.

- Right-click taskbar
- Select Properties
- Select Navigation tab
- Check "*Replace Command Prompt with Windows PowerShell...*"
- Click OK





Who Am I?

- 20 Years In K12 Technology
  - Intern, Tech, Coordinator, Engineer
- Lifelong Learner
  - /r/sysadmin, /r/k12sysadmin, RSS feeds
- Relentlessly Inquisitive
  - Let's Ask The WinAdmins Slack
- Problem Solver
  - Professional Googler
- Voracious Reader
  - docs.microsoft.com
- Community Minded
  - MAEDS, MISCUG, WMISMUG
- #ImpostorSyndromeBeDamned

Past Presentations

2013 MAEDS Fall Conference
Attended 'Marketing Yourself' by Kris Young and Kevin Galbraith
"Our Name, Reputation and Skill Sets Need to be KNOWN"

2014 MAEDS Spring PD Day
First time presenting and it's a 'ConfigMgr panel' for a half day session.

2014 MAEDS Fall Conference
Presented 'PADT and SCCM'

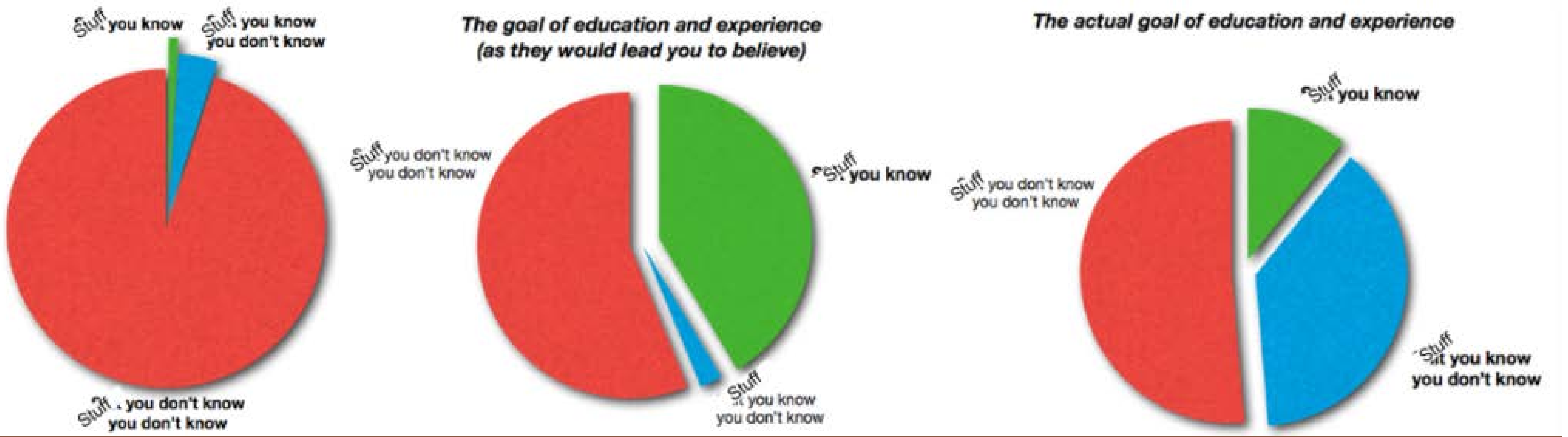
2015 MAEDS Spring PD Day
Presented 'ConfigMgr panel' for a half day session.

2015 MAEDS Fall Conference
Presented 'Automate ALL THE THINGS with PowerShell App Deployment Toolkit'

2017 MAEDS Fall Conference
Presented 'PowerShell – Intro session for those that have been too afraid to take the plunge'

2018 MAEDS Fall Conference
Presented 'PowerShell - Intermediate Session for Those That Overcame Their Fears and Took the Plunge'

2019 MACUL Conference
Presented 'Office 365 Administration'



No One Know What the F*** They're Doing (or "The 3 Types of Knowledge")

Steve Schwartz – February 9, 2010

http://jangosteve.com/post/380926251/no-one-knows-what-theyre-doing

Be The Master

"Teaching does not always feel rewarding. It doesn't need to be. It is a repayment of something that was done for you. It is not a good thing that you do; it is an obligation that you have."

https://donjones.com/2017/10/19/become-the-master-or-go-away/

Why PowerShell?

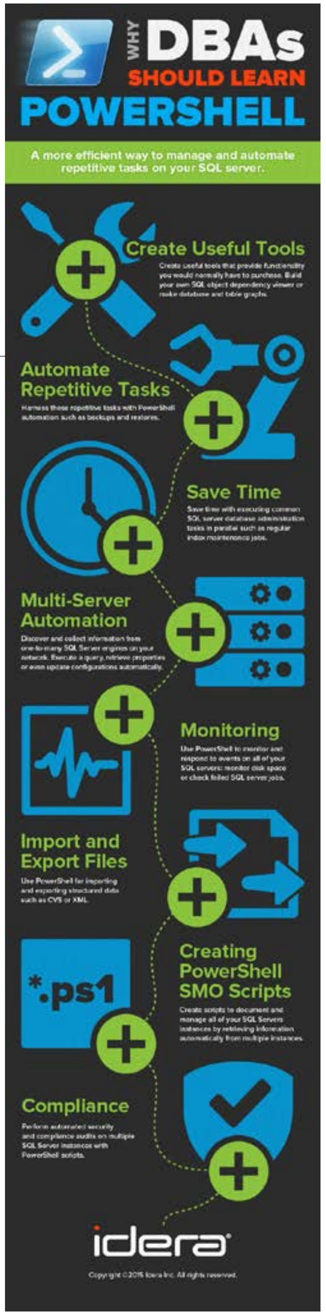
jsnover
@jsnover
Following

GUI-only, Click-Next, no-value-add Admins will be replaced with a new type of Admin - the kind that greet you in the lobby #LearnPowerShell

Images are linked to appropriate articles that are worth a read

```
Go away or I will replace you
with a
very small script.
```

Are you too busy to improve?
No thanks!
We are too busy
Håkan Forss @hakanforss http://hakanforss.wordpress.com
This illustration is inspired by and in part derived from the work by Scott Simmerman, "The Square Wheels Guy" http://www.performancemanagementcompany.com/


WHY DBAs SHOULD LEARN POWERSHELL

A more efficient way to manage and automate repetitive tasks on your SQL server.

Create Useful Tools
Create useful tools that provide functionality you would normally have to purchase. Build your own SQL object dependency viewer or make database and table graphs.

Automate Repetitive Tasks
Harness those repetitive tasks with PowerShell automation such as backups and restores.

Save Time
Save time with executing common SQL server database administration tasks in parallel such as regular index maintenance jobs.

Multi-Server Automation
Discover and collect information from one-to-many SQL Server engines on your network. Execute a query, retrieve properties or even update configurations automatically.

Monitoring
Use PowerShell to monitor and respond to events on all of your SQL servers: monitor disk space or check failed SQL server jobs.

Import and Export Files
Use PowerShell for importing and exporting structured data such as CVS or XML.

Creating PowerShell SMO Scripts
Create scripts to document and manage all of your SQL Servers instances by retrieving information automatically from multiple instances.

Compliance
Perform automated security and compliance audits on multiple SQL Server instances with PowerShell scripts.

idera

Why PowerShell?

- Discoverability
- Consistency
- Verb-Noun Syntax
- Update-Help
- Object Orientation
- PSDrive
- Parameters
- Functions

- Modules
  - PowerShell App Deployment Toolkit
  - PowerCLI (VMWare)
  - AutoBrowse (Internet Explorer Automation)
  - WASP (GUI Automation)
  - ShowUI (GUI Building)
  - PSCX (PowerShell Community Extensions)
  - Posh-SSH (SSH Automation)

NOT FREE, but awesome

- ISESteroids (ISE GUI enhancement, version control, script refactoring)

What does PowerShell work with?

- Active Directory (AD)
- Active Directory Rights Management Services (AD RMS)
- Group Policy (GPO/GPP)
- Office 365
- SharePoint Server
- Skype for Business
- SQL Server
- Best Practice Analyzer (BPA)
- RESTful/SOAP API Calls
- Internet Information Services (IIS)
- Remote Desktop Services (RDS)
- System Center Suite
  - Configuration Manager (SCCM)
  - Operations Manager (SCOM)
  - Orchestrator (SCORCH)
  - Service Manager (SCSM)
  - Virtual Machine Manager (SCVMM)
  - Data Protection Manager (SCDPM)
- File System
- Registry
- WMI/CIM
- Event Viewer
- AppLocker

Vocabulary – 01/02

Shell: the command interpreter that is used to pass commands to the operating system

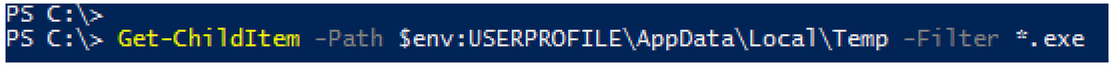
ISE: the Integrated Scripting Environment is an application where you can run, test, and debug scripts in a single GUI with tab completion, syntax coloring, selective execution, and context-sensitive help

Cmdlet: a task-oriented command that is typically used to return a .NET Framework object to the next command in the pipeline

Variable: a name given to stored information, e.g. $users, $iWishYouHadAStrongPassword, $x

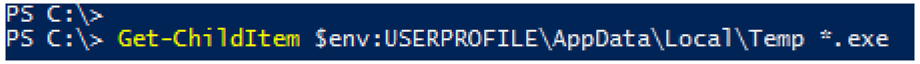
Parameter: input values or arguments used by the cmdlet or script to make it more dynamic

- **Named**:



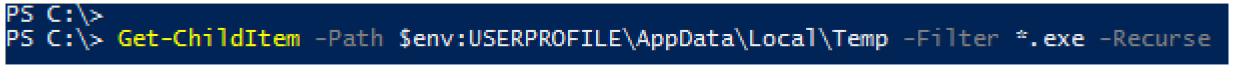
```
PS C:\>
PS C:\> Get-ChildItem -Path $env:USERPROFILE\AppData\Local\Temp -Filter *.exe
```

- **Positional**:



```
PS C:\>
PS C:\> Get-ChildItem $env:USERPROFILE\AppData\Local\Temp *.exe
```

- **Switch**:



```
PS C:\>
PS C:\> Get-ChildItem -Path $env:USERPROFILE\AppData\Local\Temp -Filter *.exe -Recurse
```

Object: a representation of something with methods to take actions against it and properties to access information stored within it

Vocabulary – 02/02

Module: a set of related Windows PowerShell functionalities, grouped together as a convenient unit (usually saved in a single directory)

- **Script Module**: a file (.psm1) that contains PowerShell code for functions, variables and more
- **Binary Module**: a .NET Framework assembly (.dll) that contains compiled code
- **Dynamic Module**: a module that only exists in memory (Import-PSSession)

Pipeline (|): a method to send the results of the preceding command as input to the next command

$_: “the current object in the pipeline", or "this", or $PSItem

PSSession: a persistent connection to a local or remote computer that is created, managed and closed by the user

PSDrive: a virtual drive that provides direct access to a data store (file system, registry, certificate store, SCCM)

Function: a command or series of commands grouped to run together

Alias: shortcut to a command, cmdlet or function

Tab Completion: the ability to complete cmdlet names, parameters, file paths, file names, etc with the use of the tab key

Don Jones: "If you're not willing to play a little bit you'll probably not be successful at PowerShell."

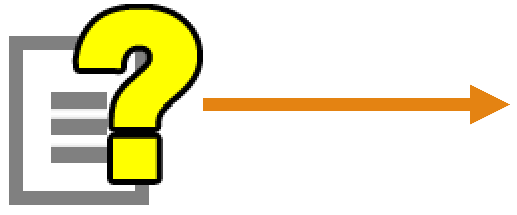
Jeffrey Snover: "I'm a Distinguished Engineer, I'm the Lead Architect with Windows Server and System Center Datacenter, and I invented the dang thing and still there's a struggle to it and that's normal.

Windows PowerShell Unplugged with Jeffrey Snover & Don Jones
TechEd North America 2014 • (1hr 16min • quotes @ 3:15)
https://channel9.msdn.com/Events/TechEd/NorthAmerica/2014/DCIM-B318

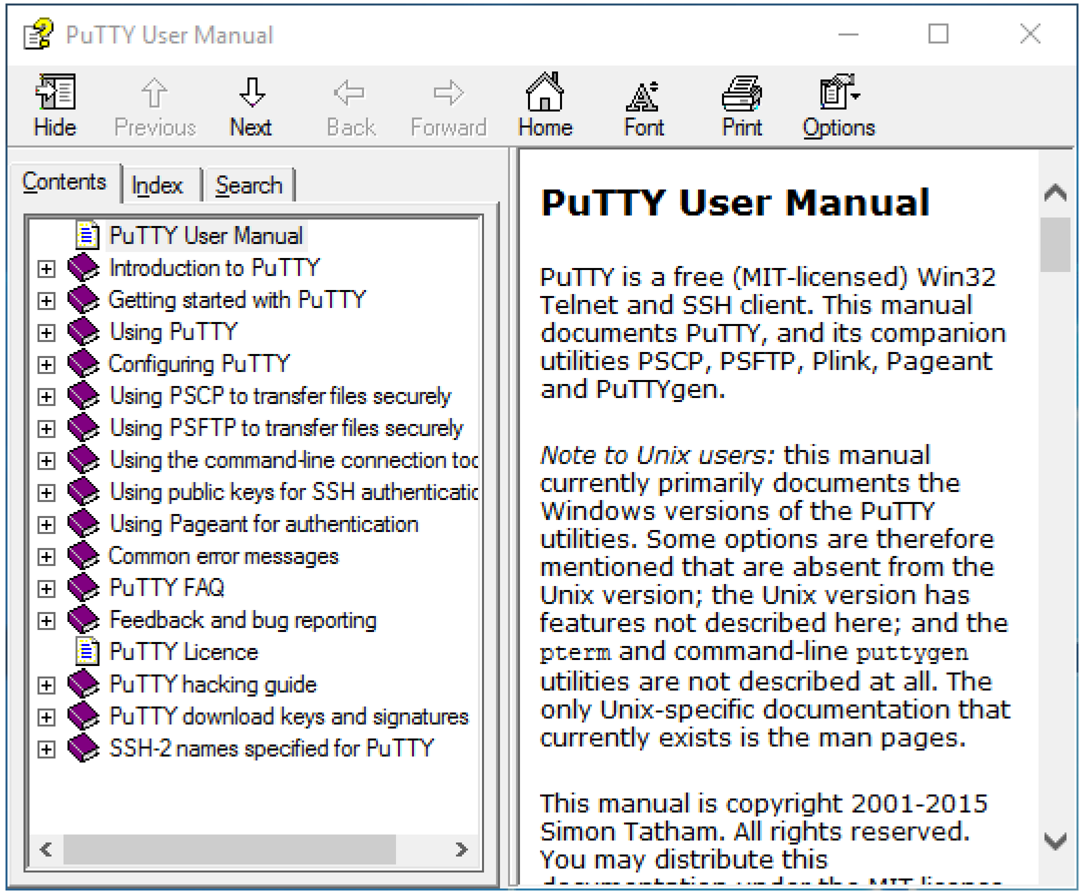
It is better to
KNOW HOW TO LEARN
than to know.

-Dr. Seuss

Remember Microsoft Compiled HTML Help Files?



putty.chm



PuTTY User Manual
Introduction to PuTTY
Getting started with PuTTY
Using PuTTY
Configuring PuTTY
Using PSCP to transfer files securely
Using PSFTP to transfer files securely
Using the command-line connection toc
Using public keys for SSH authenticatic
Using Pageant for authentication
Common error messages
PuTTY FAQ
Feedback and bug reporting
PuTTY Licence
PuTTY hacking guide
PuTTY download keys and signatures
SSH-2 names specified for PuTTY

PuTTY User Manual

PuTTY is a free (MIT-licensed) Win32 Telnet and SSH client. This manual documents PuTTY, and its companion utilities PSCP, PSFTP, Plink, Pageant and PuTTYgen.

Note to Unix users: this manual currently primarily documents the Windows versions of the PuTTY utilities. Some options are therefore mentioned that are absent from the Unix version; the Unix version has features not described here; and the `pterm` and command-line `puttygen` utilities are not described at all. The only Unix-specific documentation that currently exists is the man pages.

This manual is copyright 2001-2015 Simon Tatham. All rights reserved. You may distribute this

GET-

C	H	M	
O	E	E	O
M	L	M	D
M	P	B	U
A		E	L
N		R	E
D			

Get-Command

ALIAS: gcm

SYNOPSIS: Gets all commands.

```
PS C:\> Get-Command -Module <moduleName>
PS C:\> Get-Command -Verb Get
PS C:\> Get-Command -Verb Get -Noun P*
PS C:\> Get-Command Get-P*
PS C:\> Get-Command Get-P* -Module Microsoft.PowerShell.Utility
PS C:\> Get-Command Get-P* -Module Microsoft.PowerShell.Management
```

Cheat Mode For Slide Prep

```
PS C:\> Get-Command -Noun Object | Select-Object -ExpandProperty
Name | Clip
```

Get-Help

ALIAS: help, man

SYNOPSIS: Displays information about Windows PowerShell commands and concepts.

PS C:\> Get-Help *<cmdlet>* -Examples

PS C:\> Get-Help *<cmdlet>* -Full

PS C:\> Get-Help *<cmdlet>* -ShowWindow

PS C:\> Get-Help *<cmdlet>* -Online

PS C:\> Get-Help *<cmdlet>* | Clip

Cheat Mode For Slide Prep

PS C:\> Get-Help <cmdlet> | Select-Object -ExpandProperty Synopsis | Clip

Get-Member

ALIAS: gm

SYNOPSIS: Gets the properties and methods of objects.

```
PS C:\> <cmdlet> | Get-Member
```

Caveat

```
PS C:\> Get-ADUser cthomas | Get-Member
```

vs.

```
PS C:\> Get-ADUser cthomas -Properties * | Get-Member
```

Get-Module

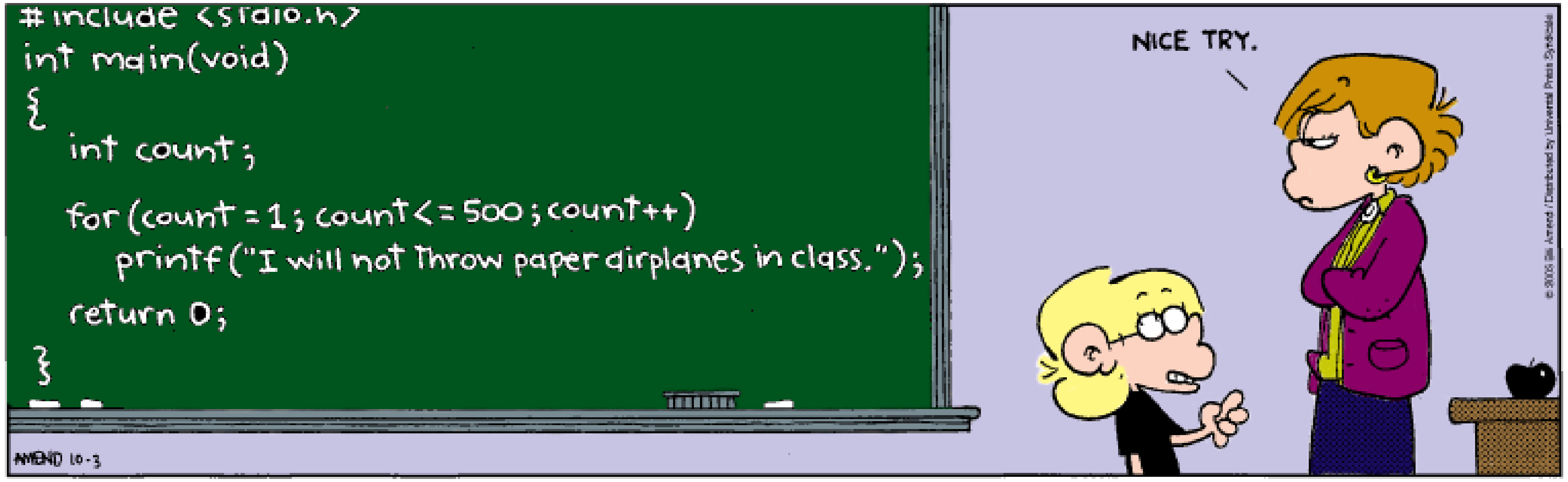
ALIAS: gmo

SYNOPSIS: Gets the modules that have been imported or that can be imported into the current session.

```
PS C:\> Get-Module
```

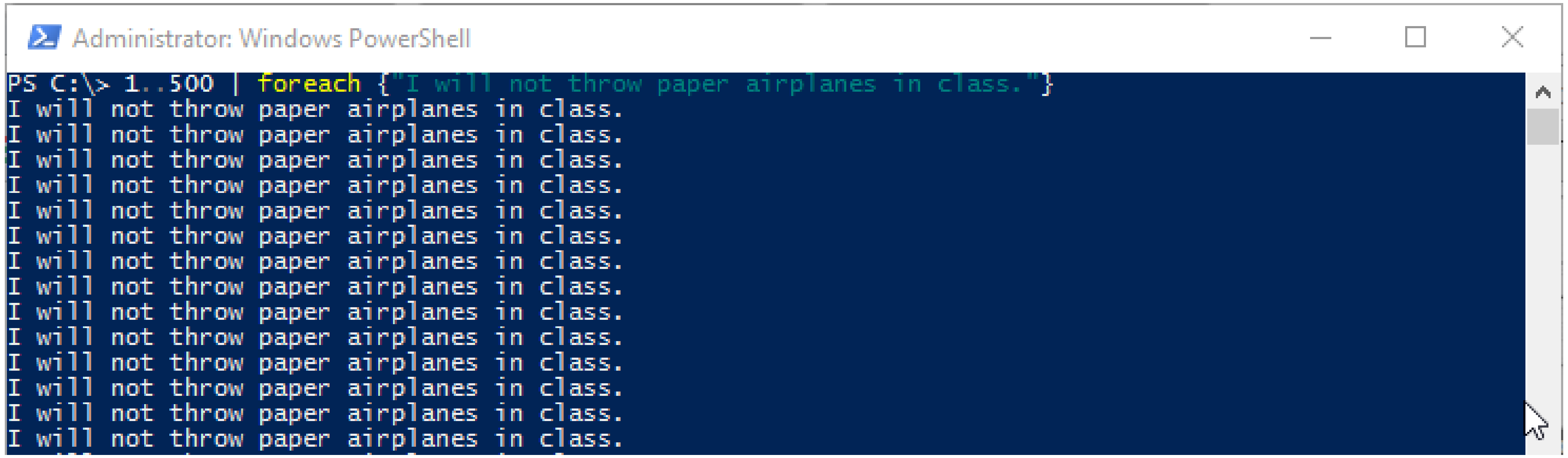
```
PS C:\> Get-Module -ListAvailable
```

```
PS C:\> $session_DC1_IISD = New-PSSession -ComputerName V-DC1
PS C:\> Get-Module -PSSession $session_DC1_IISD -ListAvailable
```



```
#include <stdio.h>
int main(void)
{
  int count;

  for(count = 1; count<=500; count++)
     printf("I will not Throw paper airplanes in class.");
  return 0;
}
```

Object Manipulation



```
PS C:\> 1..500 | foreach {"I will not throw paper airplanes in class."}
I will not throw paper airplanes in class.
I will not throw paper airplanes in class.
I will not throw paper airplanes in class.
I will not throw paper airplanes in class.
I will not throw paper airplanes in class.
I will not throw paper airplanes in class.
I will not throw paper airplanes in class.
I will not throw paper airplanes in class.
I will not throw paper airplanes in class.
I will not throw paper airplanes in class.
I will not throw paper airplanes in class.
I will not throw paper airplanes in class.
I will not throw paper airplanes in class.
I will not throw paper airplanes in class.
```

Compare-Object

ALIAS: compare, diff

SYNOPSIS: Compares two sets of objects.

```
PS C:\> Compare-Object –ReferenceObject $(Get-Content <file1path>)
-DifferenceObject $(Get-Content <file2path>)
```

```
PS C:\> Compare-Object –ReferenceObject $(Get-Content <file1path>)
-DifferenceObject $(Get-Content <file2path>) -IncludeEqual
```

```
PS C:\> $processes_before = Get-Process
PS C:\> notepad
PS C:\> $processes_after  = Get-Process
PS C:\> Compare-Object -ReferenceObject $processes_before
-DifferenceObject $processes_after
```

Full Disclosure

I don’t use this, but see how it could be useful for monitoring scripts. I always tend to lean toward if statements and Where-Object filters.
Get-Help about_If

Foreach-Object

ALIAS: %, foreach

SYNOPSIS: Performs an operation against each item in a collection of input objects.

```
PS C:\> <cmdlet> | Foreach-Object -Process { do-something }
```

```
PS C:\> Get-ADUser -Filter {UserPrincipalName -like "*cthomas*"} |
Select-Object -ExpandProperty UserPrincipalName | ForEach-Object
-Process {$_.Split(".")}
```

Group-Object

ALIAS: group

SYNOPSIS: Groups objects that contain the same value for specified properties.

```
PS C:\> <cmdlet> | Group-Object -Property <propertyName>
```

```
PS C:\> <cmdlet> | Group-Object -Descending
```

```
PS C:\> $events = Get-EventLog -LogName System -Newest 1000
PS C:\> $events | Group-Object – Property EventID
```

Measure-Object

ALIAS: measure

SYNOPSIS: Calculates the numeric properties of objects, and the characters, words, and lines in string objects, such as files of text.

```
PS C:\> <cmdlet> | Measure-Object

PS C:\> Get-ADUser –Filter * | Measure-Object

PS C:\> Get-MsolUser –All | Measure-Object
```

New-Object

ALIAS: None

SYNOPSIS: Creates an instance of a Microsoft .NET Framework or COM object.

```
PS C:\> $customObject = New-Object -TypeName PSObject
PS C:\> $customObject | Add-Member -MemberType NoteProperty -Name DistinguishedName -Value (Get-ADUser cthomas).DistinguishedName
PS C:\> $customObject | Add-Member -MemberType NoteProperty -Name GroupMemberships -Value (Get-ADPrincipalGroupMembership –Identity cthomas).Name
PS C:\> $customObject
```

```
PS C:\> $customObject2 = [PSCustomObject]@{
>> DistinguishedName = (Get-ADUser cthomas).DistinguishedName
>> GroupMemberships = (Get-ADPrincipalGroupMembership -Identity cthomas).Name
>> }
PS C:\> $customObject2
```

Select-Object

ALIAS: select

SYNOPSIS: Selects objects or object properties.

```
PS C:\> <cmdlet> | Select-Object -Property <propertyList>

PS C:\> “a”, “b”, “c”, “b”, “c” | Measure-Object
PS C:\> “a”, “b”, “c”, “b”, “c” | Select-Object -Unique | Measure-Object

PS C:\> Get-ADUser -Filter * | Select-Object -First 5

PS C:\> $users = Import-CSV -Path C:\scripts\users.csv | Select-Object -
Property UPN, SN, FN, LN, OU
```

Sort-Object

ALIAS: sort

SYNOPSIS: Sorts objects by property values.

```
PS C:\> <cmdlet> | Sort-Object
PS C:\> <cmdlet> | Sort-Object -Unique
PS C:\> Get-ChildItem | Sort-Object
PS C:\> Get-MsolUser -All | Sort-Object -Property UserPrincipalName
```

Tee-Object

ALIAS: tee

SYNOPSIS: Saves command output in a file or variable and also sends it down the pipeline.

```
PS C:\> <cmdlet> | Tee-Object -FilePath <filepath>
```

```
PS C:\> <cmdlet> | Tee-Object -FilePath
```

```
PS C:\> Get-ChildItem -Path $env:USERPROFILE -Recurse | Tee-Object
-FilePath C:\scripts\allappdatafiles.txt | Where-Object -FilterScript
{$_.Name -like "*.exe"} | Out-File -FilePath
C:\scripts\exeappdatafiles.txt
```

Full Disclosure

I don’t use this, but I can see it’s uses in data collection scripts.

Where-Object

ALIAS: ?, where

SYNOPSIS: Selects objects from a collection based on their property values.

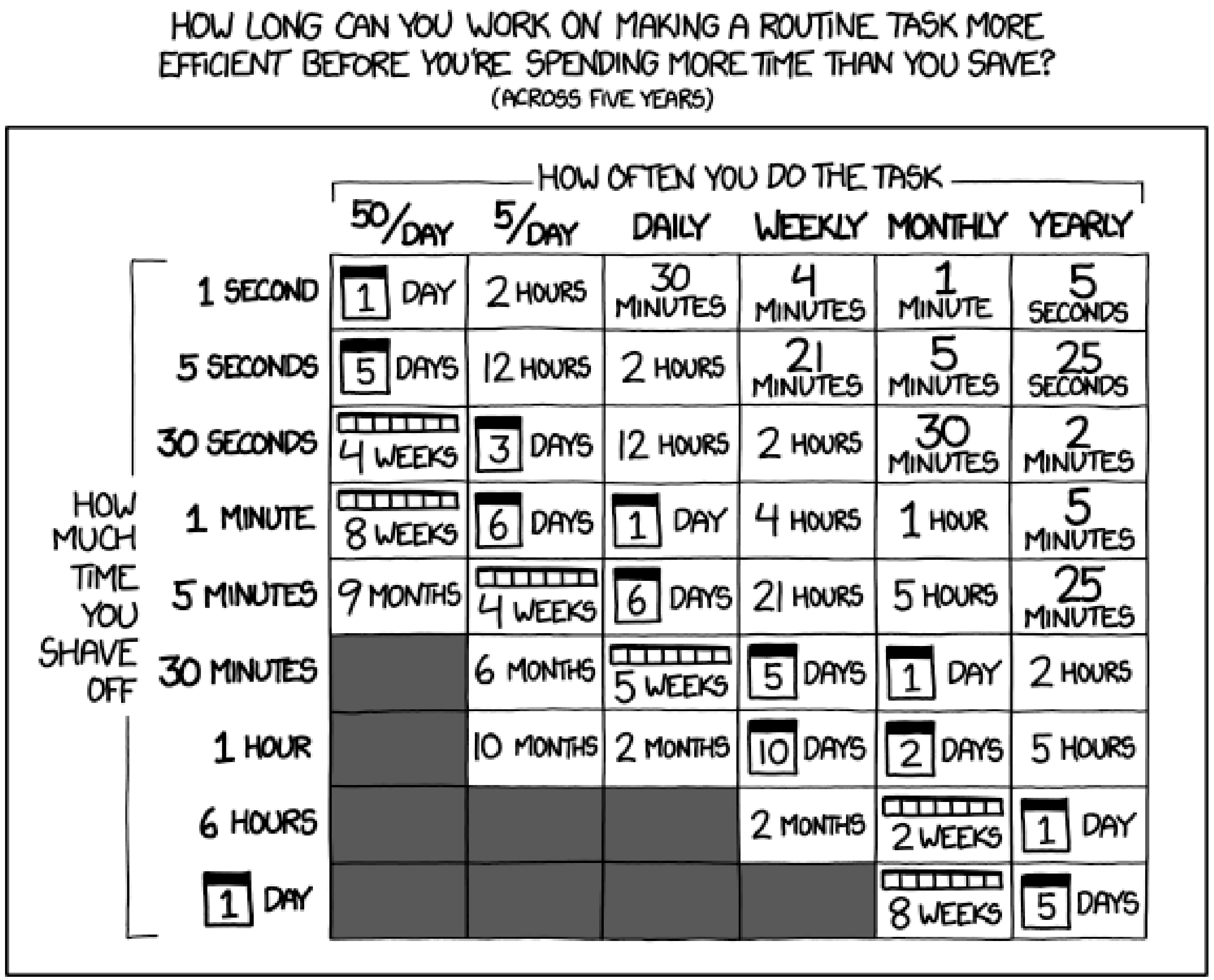
PS C:\> *<cmdlet>* | Where-Object –FilterScript { $_.*PropertyName* -like "*query*" }

PS C:\> Get-WmiObject -Class win32_product | Where-Object -FilterScript {$_.InstallDate -eq 20150922} | Select-Object -Property Name, IdentifyingNumber
PS C:\> gwmi win32_product | ?{$_.InstallDate –eq 20150922} | select name,identifyingnumber

Cheat Mode For Slide Prep

PS C:\> Get-Alias | Where-Object -FilterScript {$_.ResolvedCommand -like "*<cmdlet>*"} | Select-Object -ExpandProperty Name | Clip

Demos and Examples

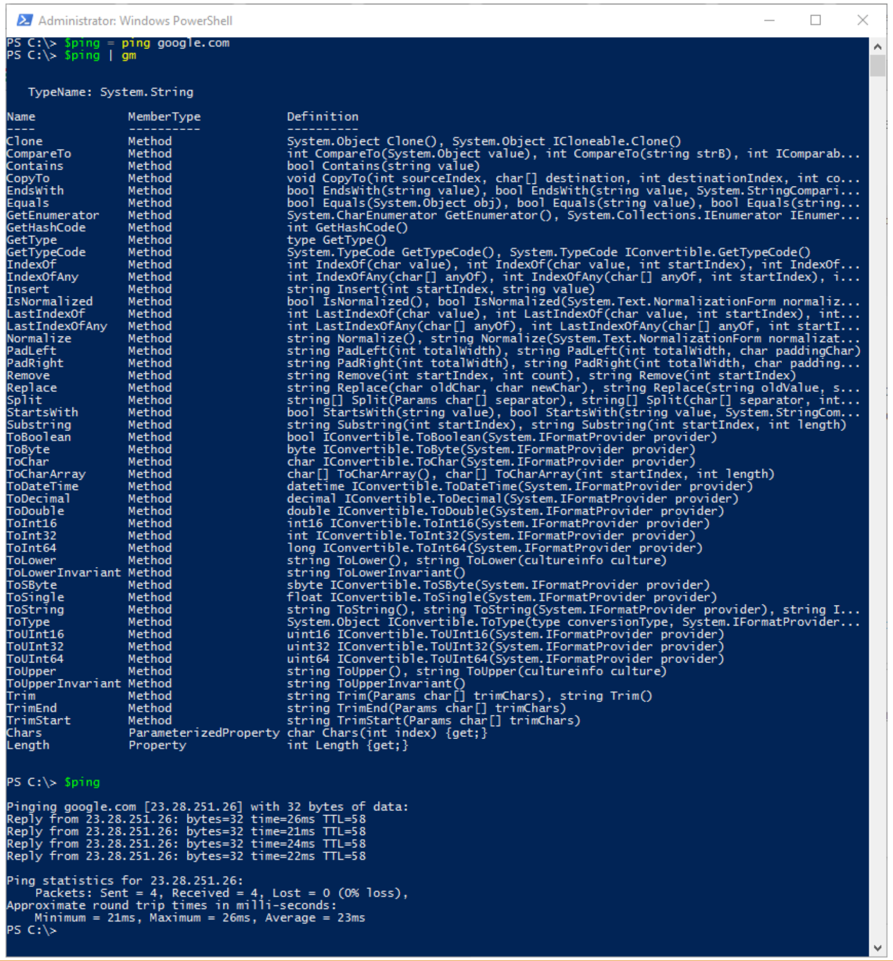


HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE? (ACROSS FIVE YEARS)

HOW OFTEN YOU DO THE TASK

HOW MUCH TIME YOU SHAVE OFF	50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
1 SECOND	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
5 SECONDS	5 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
30 SECONDS	4 WEEKS	3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
1 MINUTE	8 WEEKS	6 DAYS	1 DAY	4 HOURS	1 HOUR	5 MINUTES
5 MINUTES	9 MONTHS	4 WEEKS	6 DAYS	21 HOURS	5 HOURS	25 MINUTES
30 MINUTES	■	6 MONTHS	5 WEEKS	5 DAYS	1 DAY	2 HOURS
1 HOUR	■	10 MONTHS	2 MONTHS	10 DAYS	2 DAYS	5 HOURS
6 HOURS	■	■	■	2 MONTHS	2 WEEKS	1 DAY
1 DAY	■	■	■	■	8 WEEKS	5 DAYS

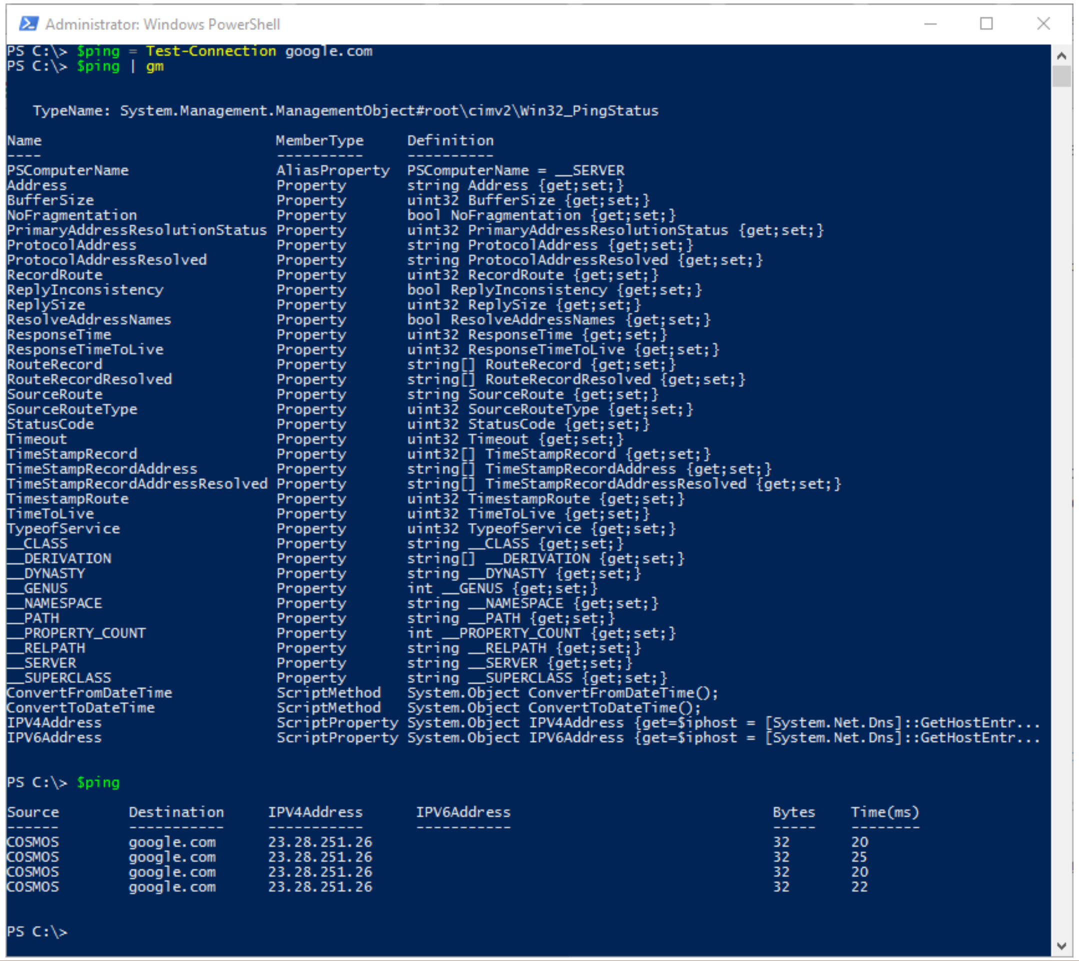
Ping



Administrator: Windows PowerShell

```
PS C:\> $ping = ping google.com
PS C:\> $ping | gm


   TypeName: System.String

Name             MemberType            Definition
----             ----------            ----------
Clone            Method                System.Object Clone(), System.Object ICloneable.Clone()
CompareTo        Method                int CompareTo(System.Object value), int CompareTo(string strB), int IComparab...
Contains         Method                bool Contains(string value)
CopyTo           Method                void CopyTo(int sourceIndex, char[] destination, int destinationIndex, int co...
EndsWith         Method                bool EndsWith(string value), bool EndsWith(string value, System.StringCompari...
Equals           Method                bool Equals(System.Object obj), bool Equals(string value), bool Equals(string...
GetEnumerator    Method                System.CharEnumerator GetEnumerator(), System.Collections.IEnumerator IEnumer...
GetHashCode      Method                int GetHashCode()
GetType          Method                type GetType()
GetTypeCode      Method                System.TypeCode GetTypeCode(), System.TypeCode IConvertible.GetTypeCode()
IndexOf          Method                int IndexOf(char value), int IndexOf(char value, int startIndex), int IndexOf...
IndexOfAny       Method                int IndexOfAny(char[] anyOf), int IndexOfAny(char[] anyOf, int startIndex), i...
Insert           Method                string Insert(int startIndex, string value)
IsNormalized     Method                bool IsNormalized(), bool IsNormalized(System.Text.NormalizationForm normaliz...
LastIndexOf      Method                int LastIndexOf(char value), int LastIndexOf(char value, int startIndex), int...
LastIndexOfAny   Method                int LastIndexOfAny(char[] anyOf), int LastIndexOfAny(char[] anyOf, int startI...
Normalize        Method                string Normalize(), string Normalize(System.Text.NormalizationForm normalizat...
PadLeft          Method                string PadLeft(int totalWidth), string PadLeft(int totalWidth, char paddingChar)
PadRight         Method                string PadRight(int totalWidth), string PadRight(int totalWidth, char padding...
Remove           Method                string Remove(int startIndex, int count), string Remove(int startIndex)
Replace          Method                string Replace(char oldChar, char newChar), string Replace(string oldValue, s...
Split            Method                string[] Split(Params char[] separator), string[] Split(char[] separator, int...
StartsWith       Method                bool StartsWith(string value), bool StartsWith(string value, System.StringCom...
Substring        Method                string Substring(int startIndex), string Substring(int startIndex, int length)
ToBoolean        Method                bool IConvertible.ToBoolean(System.IFormatProvider provider)
ToByte           Method                byte IConvertible.ToByte(System.IFormatProvider provider)
ToChar           Method                char IConvertible.ToChar(System.IFormatProvider provider)
ToCharArray      Method                char[] ToCharArray(), char[] ToCharArray(int startIndex, int length)
ToDateTime       Method                datetime IConvertible.ToDateTime(System.IFormatProvider provider)
ToDecimal        Method                decimal IConvertible.ToDecimal(System.IFormatProvider provider)
ToDouble         Method                double IConvertible.ToDouble(System.IFormatProvider provider)
ToInt16          Method                int16 IConvertible.ToInt16(System.IFormatProvider provider)
ToInt32          Method                int IConvertible.ToInt32(System.IFormatProvider provider)
ToInt64          Method                long IConvertible.ToInt64(System.IFormatProvider provider)
ToLower          Method                string ToLower(), string ToLower(cultureinfo culture)
ToLowerInvariant Method                string ToLowerInvariant()
ToSByte          Method                sbyte IConvertible.ToSByte(System.IFormatProvider provider)
ToSingle         Method                float IConvertible.ToSingle(System.IFormatProvider provider)
ToString         Method                string ToString(), string ToString(System.IFormatProvider provider), string I...
ToType           Method                System.Object IConvertible.ToType(type conversionType, System.IFormatProvider...
ToUInt16         Method                uint16 IConvertible.ToUInt16(System.IFormatProvider provider)
ToUInt32         Method                uint32 IConvertible.ToUInt32(System.IFormatProvider provider)
ToUInt64         Method                uint64 IConvertible.ToUInt64(System.IFormatProvider provider)
ToUpper          Method                string ToUpper(), string ToUpper(cultureinfo culture)
ToUpperInvariant Method                string ToUpperInvariant()
Trim             Method                string Trim(Params char[] trimChars), string Trim()
TrimEnd          Method                string TrimEnd(Params char[] trimChars)
TrimStart        Method                string TrimStart(Params char[] trimChars)
Chars            ParameterizedProperty char Chars(int index) {get;}
Length           Property              int Length {get;}


PS C:\> $ping

Pinging google.com [23.28.251.26] with 32 bytes of data:
Reply from 23.28.251.26: bytes=32 time=26ms TTL=58
Reply from 23.28.251.26: bytes=32 time=21ms TTL=58
Reply from 23.28.251.26: bytes=32 time=24ms TTL=58
Reply from 23.28.251.26: bytes=32 time=22ms TTL=58

Ping statistics for 23.28.251.26:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 21ms, Maximum = 26ms, Average = 23ms
PS C:\>
```

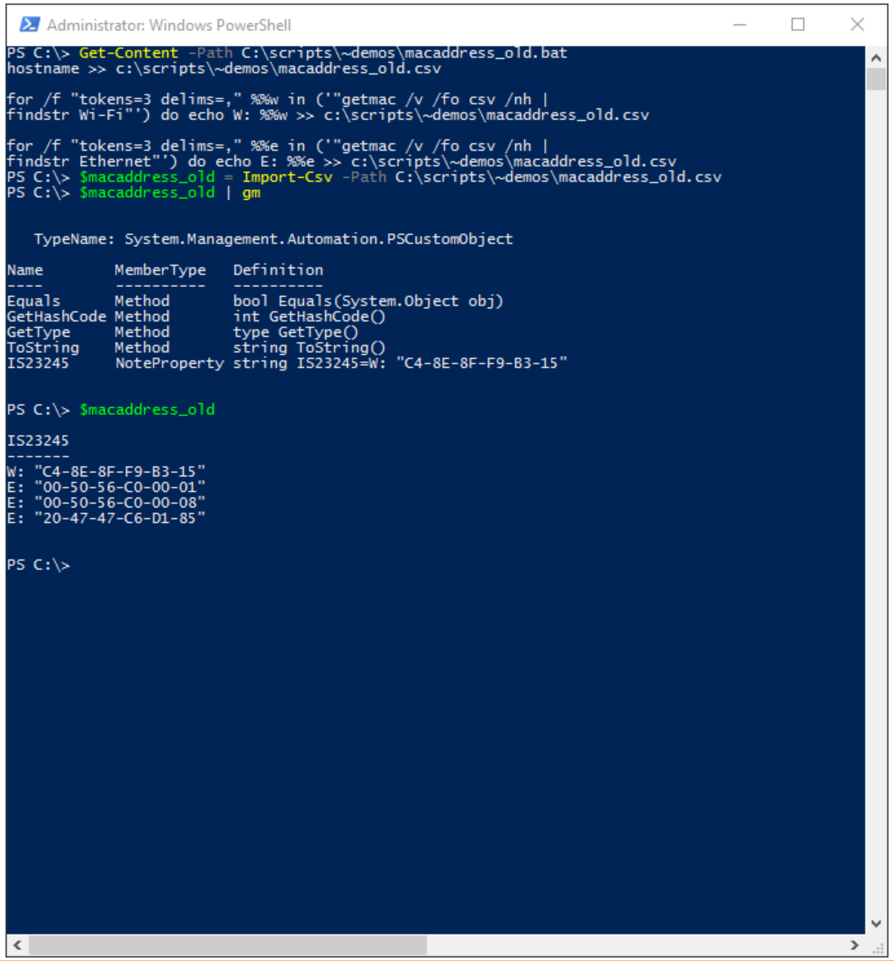
Test-Connection



Administrator: Windows PowerShell

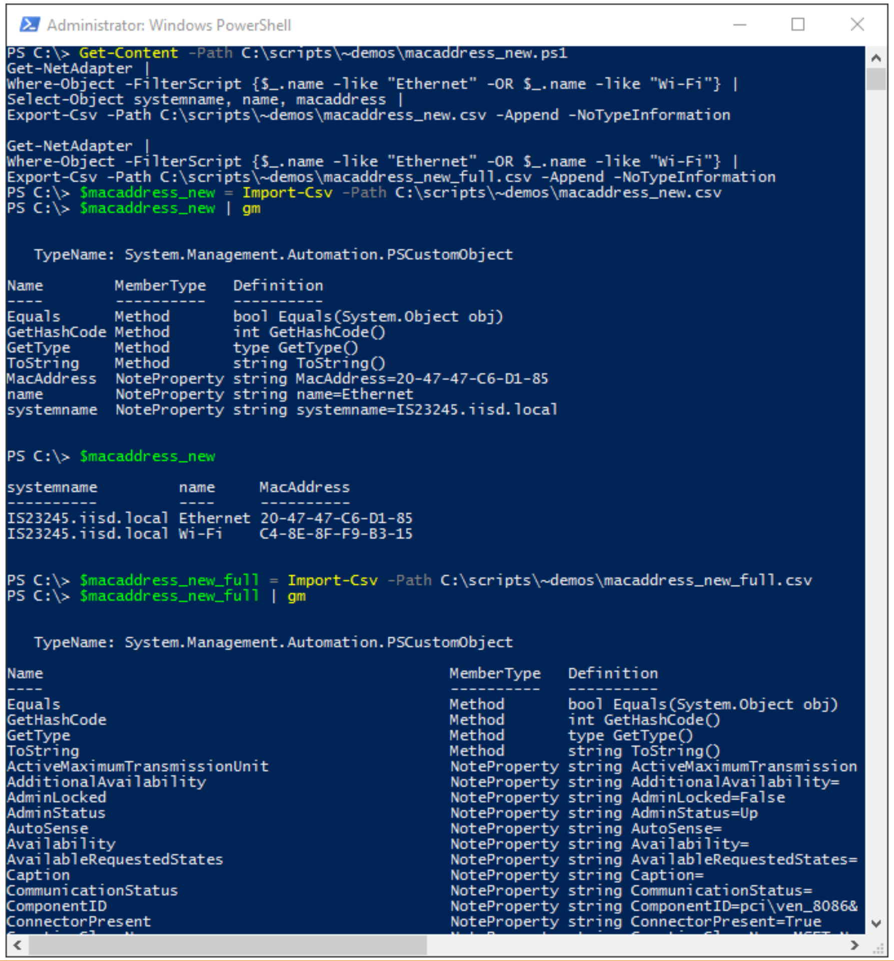
```
PS C:\> $ping = Test-Connection google.com
PS C:\> $ping | gm


   TypeName: System.Management.ManagementObject#root\cimv2\Win32_PingStatus

Name                           MemberType     Definition
----                           ----------     ----------
PSComputerName                 AliasProperty  PSComputerName = __SERVER
Address                        Property       string Address {get;set;}
BufferSize                     Property       uint32 BufferSize {get;set;}
NoFragmentation                Property       bool NoFragmentation {get;set;}
PrimaryAddressResolutionStatus Property       uint32 PrimaryAddressResolutionStatus {get;set;}
ProtocolAddress                Property       string ProtocolAddress {get;set;}
ProtocolAddressResolved        Property       string ProtocolAddressResolved {get;set;}
RecordRoute                    Property       uint32 RecordRoute {get;set;}
ReplyInconsistency             Property       bool ReplyInconsistency {get;set;}
ReplySize                      Property       uint32 ReplySize {get;set;}
ResolveAddressNames            Property       bool ResolveAddressNames {get;set;}
ResponseTime                   Property       uint32 ResponseTime {get;set;}
ResponseTimeToLive             Property       uint32 ResponseTimeToLive {get;set;}
RouteRecord                    Property       string[] RouteRecord {get;set;}
RouteRecordResolved            Property       string[] RouteRecordResolved {get;set;}
SourceRoute                    Property       string SourceRoute {get;set;}
SourceRouteType                Property       uint32 SourceRouteType {get;set;}
StatusCode                     Property       uint32 StatusCode {get;set;}
Timeout                        Property       uint32 Timeout {get;set;}
TimeStampRecord                Property       uint32[] TimeStampRecord {get;set;}
TimeStampRecordAddress         Property       string[] TimeStampRecordAddress {get;set;}
TimeStampRecordAddressResolved Property       string[] TimeStampRecordAddressResolved {get;set;}
TimestampRoute                 Property       uint32 TimestampRoute {get;set;}
TimeToLive                     Property       uint32 TimeToLive {get;set;}
TypeofService                  Property       uint32 TypeofService {get;set;}
__CLASS                        Property       string __CLASS {get;set;}
__DERIVATION                   Property       string[] __DERIVATION {get;set;}
__DYNASTY                      Property       string __DYNASTY {get;set;}
__GENUS                        Property       int __GENUS {get;set;}
__NAMESPACE                    Property       string __NAMESPACE {get;set;}
__PATH                         Property       string __PATH {get;set;}
__PROPERTY_COUNT               Property       int __PROPERTY_COUNT {get;set;}
__RELPATH                      Property       string __RELPATH {get;set;}
__SERVER                       Property       string __SERVER {get;set;}
__SUPERCLASS                   Property       string __SUPERCLASS {get;set;}
ConvertFromDateTime            ScriptMethod   System.Object ConvertFromDateTime();
ConvertToDateTime              ScriptMethod   System.Object ConvertToDateTime();
IPV4Address                    ScriptProperty System.Object IPV4Address {get=$iphost = [System.Net.Dns]::GetHostEntr...
IPV6Address                    ScriptProperty System.Object IPV6Address {get=$iphost = [System.Net.Dns]::GetHostEntr...


PS C:\> $ping

Source        Destination     IPV4Address      IPV6Address                                   Bytes    Time(ms)
------        -----------     -----------      -----------                                   -----    --------
COSMOS        google.com      23.28.251.26                                                   32       20
COSMOS        google.com      23.28.251.26                                                   32       25
COSMOS        google.com      23.28.251.26                                                   32       20
COSMOS        google.com      23.28.251.26                                                   32       22


PS C:\>
```

hostname/getmac


Administrator: Windows PowerShell

```
PS C:\> Get-Content -Path C:\scripts\~demos\macaddress_old.bat
hostname >> c:\scripts\~demos\macaddress_old.csv

for /f "tokens=3 delims=," %%w in ('"getmac /v /fo csv /nh |
findstr Wi-Fi"') do echo W: %%w >> c:\scripts\~demos\macaddress_old.csv

for /f "tokens=3 delims=," %%e in ('"getmac /v /fo csv /nh |
findstr Ethernet"') do echo E: %%e >> c:\scripts\~demos\macaddress_old.csv
PS C:\> $macaddress_old = Import-Csv -Path C:\scripts\~demos\macaddress_old.csv
PS C:\> $macaddress_old | gm


   TypeName: System.Management.Automation.PSCustomObject

Name        MemberType   Definition
----        ----------   ----------
Equals      Method       bool Equals(System.Object obj)
GetHashCode Method       int GetHashCode()
GetType     Method       type GetType()
ToString    Method       string ToString()
IS23245     NoteProperty string IS23245=W: "C4-8E-8F-F9-B3-15"


PS C:\> $macaddress_old

IS23245
-------
W: "C4-8E-8F-F9-B3-15"
E: "00-50-56-C0-00-01"
E: "00-50-56-C0-00-08"
E: "20-47-47-C6-D1-85"


PS C:\>
```

Get-NetAdapter


Administrator: Windows PowerShell

```
PS C:\> Get-Content -Path C:\scripts\~demos\macaddress_new.ps1
Get-NetAdapter |
Where-Object -FilterScript {$_.name -like "Ethernet" -OR $_.name -like "Wi-Fi"} |
Select-Object systemname, name, macaddress |
Export-Csv -Path C:\scripts\~demos\macaddress_new.csv -Append -NoTypeInformation

Get-NetAdapter |
Where-Object -FilterScript {$_.name -like "Ethernet" -OR $_.name -like "Wi-Fi"} |
Export-Csv -Path C:\scripts\~demos\macaddress_new_full.csv -Append -NoTypeInformation
PS C:\> $macaddress_new = Import-Csv -Path C:\scripts\~demos\macaddress_new.csv
PS C:\> $macaddress_new | gm


   TypeName: System.Management.Automation.PSCustomObject

Name        MemberType   Definition
----        ----------   ----------
Equals      Method       bool Equals(System.Object obj)
GetHashCode Method       int GetHashCode()
GetType     Method       type GetType()
ToString    Method       string ToString()
MacAddress  NoteProperty string MacAddress=20-47-47-C6-D1-85
name        NoteProperty string name=Ethernet
systemname  NoteProperty string systemname=IS23245.iisd.local


PS C:\> $macaddress_new

systemname         name     MacAddress
----------         ----     ----------
IS23245.iisd.local Ethernet 20-47-47-C6-D1-85
IS23245.iisd.local Wi-Fi    C4-8E-8F-F9-B3-15


PS C:\> $macaddress_new_full = Import-Csv -Path C:\scripts\~demos\macaddress_new_full.csv
PS C:\> $macaddress_new_full | gm


   TypeName: System.Management.Automation.PSCustomObject

Name                                             MemberType   Definition
----                                             ----------   ----------
Equals                                           Method       bool Equals(System.Object obj)
GetHashCode                                      Method       int GetHashCode()
GetType                                          Method       type GetType()
ToString                                         Method       string ToString()
ActiveMaximumTransmissionUnit                    NoteProperty string ActiveMaximumTransmission
AdditionalAvailability                           NoteProperty string AdditionalAvailability=
AdminLocked                                      NoteProperty string AdminLocked=False
AdminStatus                                      NoteProperty string AdminStatus=Up
AutoSense                                        NoteProperty string AutoSense=
Availability                                     NoteProperty string Availability=
AvailableRequestedStates                         NoteProperty string AvailableRequestedStates=
Caption                                          NoteProperty string Caption=
CommunicationStatus                              NoteProperty string CommunicationStatus=
ComponentID                                      NoteProperty string ComponentID=pci\ven_8086&
ConnectorPresent                                 NoteProperty string ConnectorPresent=True
```

KEEP
CALM
AND
LEARN
POWERSHELL

Resources – 01/02

- Jeffrey Snover
  - http://https://twitter.com/jsnover
  - http://www.jsnover.com/blog
  - https://channel9.msdn.com/Events/Speakers/Jeffrey-Snover
  - http://www.jsnover.com/Docs/MonadManifesto.pdf
  - http://www.jsnover.com/blog/2011/10/01/monad-manifesto/
- Don Jones
  - https://twitter.com/concentrateddon
  - http://donjones.com/
  - https://channel9.msdn.com/Events/Speakers/Don-Jones
  - http://www.amazon.com/Learn-Windows-PowerShell-Month-Lunches/dp/1617291080
  - https://www.youtube.com/playlist?list=PL6D474E721138865A
  - https://www.cbtnuggets.com/it-training/microsoft-windows-powershell-3
  - https://www.cbtnuggets.com/it-training/microsoft-windows-powershell-2-3-4

Resources – 02/02

- http://blogs.technet.com/b/heyscriptingguy/
- http://powershell.org/wp/category/podcast/

- http://social.technet.microsoft.com/wiki/contents/articles/183.windows-powershell-survival-guide.aspx

- http://ramblingcookiemonster.github.io/How-Do-I-Learn-PowerShell/
- http://ramblingcookiemonster.github.io/Pages/PowerShellResources/index.html
- http://ramblingcookiemonster.github.io/images/Cheat-Sheets/powershell-cheat-sheet.pdf
- http://ramblingcookiemonster.github.io/images/Cheat-Sheets/powershell-basic-cheat-sheet2.pdf

- https://congn.wordpress.com/2011/11/15/10-powershell-concept/

THAT CONCLUDES MY TWO-HOUR PRESENTATION. ANY QUESTIONS?
scottadams@aol.com
www.dilbert.com
DID YOU INTEND THE PRESENTATION TO BE INCOMPREHENSIBLE, OR DO YOU HAVE SOME SORT OF RARE "POWERPOINT" DISABILITY?
© 2003 United Feature Syndicate, Inc.
8/9/03
ARE THERE ANY QUESTIONS ABOUT THE CONTENT?
THERE WAS CONTENT?

Don’t forget to log your PD hour!

Thanks for attending.
Was this helpful?
Should we have more of these?