

Inhalt

Vorwort	15
Wie Sie dieses Buch nutzen	16
Achtung	16
Noch mehr Unterstützung	16
1 PowerShell: Erste Schritte	17
PowerShell installieren	19
Windows-Betriebssystem	19
PowerShell nachrüsten: macOS und Linux	32
Kompatibilität der PowerShell	35
PowerShell einrichten	37
Vorsicht mit Administratorrechten!	38
Interaktive Befehle eingeben	38
Autovervollständigung: Tippfehler vermeiden	39
Befehlszeilen erneut verwenden	41
Groß- und Kleinschreibung	41
Unvollständige und mehrzeilige Eingaben	41
PowerShell-Hilfe aus dem Internet nachladen	42
Skriptausführung erlauben	44
Weitere PowerShell-Einschränkungen	46
Wichtige PowerShell-Werkzeuge	47
PowerShell-ISE-Editor	47
VSCode (Visual Studio Code)	49
Windows-Terminal	58
Codebeispiele automatisch herunterladen	63
Befehl zum Herunterladen von Codebeispielen nachrüsten	63
Beispielcode in Zwischenablage kopieren	64
Beispielcode sofort ausführen	64
Profilskripte: PowerShell dauerhaft anpassen	65
Einzelne Profilskripte verwenden	65
Fragen stellen	69
Alle Skripte herunterladen	71
Zusammenfassung	72
Ausführungsrichtlinie festlegen	72
Hilfe nachrüsten	72
Windows PowerShell aktualisieren	73
Hilfsbefehle zum Download von Beispielcode	74
»Fehlende Befehle« verstehen	75
Zusätzliche Werkzeuge	75

2 Überblick: Was PowerShell leistet	77
Befehle lösen Aufgaben	79
Literale	81
Text	82
Zahlen	84
Datum und Zeit	84
Kommentare	85
Cmdlets: die PowerShell-Befehle	87
Nach Tätigkeitsbereich suchen (»Noun«)	88
Nach Tätigkeit suchen (»Verb«)	88
Nach Herkunft suchen	90
Standardisierte Verben	93
Cmdlets per Fenster suchen	97
Syntax verstehen	101
Ausführliche Hilfe und Beispiele	103
Anwendungsprogramme (Applications)	105
Applications starten	106
Applications finden	107
Systembefehle nutzen	110
Hilfe für Applications abrufen	112
.NET-Methoden	113
Methoden verwenden	114
Hilfe für Methoden	114
Noch mehr Methoden	116
Methoden: vielseitiger, aber kleinteiliger	117
Operatoren	122
Operatoren nachschlagen	123
Vergleichsoperatoren	124
Textoperatoren	124
Zuweisungen und Pipeline	124
Zahlenreihen	125
Befehle verbinden	126
Das Türchen-Modell	126
Normalfall: »Türchen 3«	127
Modernes Pipeline-Streaming: »Türchen 1«	128
Klassische Variablen: »Türchen 2«	133
Neue Befehle nachrüsten	134
Beispiel 1: QR-Codes generieren	135
Beispiel 2: Automatische HTML-Reports	137
Beispiel 3: Musik spielen	142
Zusammenfassung	149
 3 Skripte und Funktionen	 151
PowerShell-Skripte verstehen	152
Skriptcode eingeben	153
Skripte mit dem ISE-Editor entwickeln	153
Neues Skript anlegen	154
Skripte mit dem VSCode-Editor entwickeln	155
Neue Skripte anlegen	156

Skripte ausführen	158
Skripte innerhalb von PowerShell starten	158
Skripte außerhalb von PowerShell starten	160
Skripte automatisch starten	164
Skriptstart durch Task Scheduler	164
Profilskripte – die Autostartskripte	175
Profilskript anlegen und öffnen	179
Typische Profilskriptaufgaben durchführen	180
Neue Befehle: Funktionen	183
Schritt 1: Nützlicher Code	184
Schritt 2: Als Funktion verpacken	184
Schritt 3: Parameter definieren	185
Schritt 4: Funktionen dauerhaft verfügbar machen	190
Funktionen im Modul permanent verfügbar	190
Die gemeinsame Grundlage: der Skriptblock	192
Skripte sind gespeicherte Skriptblöcke	192
Funktionen sind vorgeladene Skriptblöcke	193
Module laden Funktionen bei Bedarf in den Speicher	193
Skript oder Funktion?	193
 4 Cmdlets – PowerShell-Befehle	 195
Parameter steuern Cmdlets	197
Argumente an Parameter übergeben	197
Parameter machen Cmdlets vielseitig	198
Politisch inkorrekt: positionale Argumente	199
Gratiszugabe: »Common Parameters«	203
Auf Fehler reagieren	205
Vielseitigkeit durch Parametersätze	206
»Schmal, aber tief« – Cmdlets sind Spezialisten	206
Mehrere Parametersätze: noch mehr Vielseitigkeit	209
Praxis: Ereignisse aus dem Ereignislogbuch lesen	212
»ISA/HASA« – wie Cmdlets in der Pipeline funktionieren	219
Das »ISA/HASA«-Prinzip	219
Praxisnutzen	221
Vorteile des Pipeline-Streamings	221
 5 Die PowerShell-Pipeline	 225
Aufbau der PowerShell-Pipeline	226
Befehle reichen Ergebnisse weiter	227
Pipeline steuert Befehle	228
Prinzipieller Aufbau der Pipeline	230
Die sechs wichtigsten Pipeline-Befehle	230
Select-Object	231
Detailinformationen festlegen	231
Unsichtbare Eigenschaften sichtbar machen	232
Eine bestimmte Eigenschaft auswählen: -ExpandProperty	234
Selbst festlegen, welche Informationen wichtig sind	236
Weitere Informationen anfügen	238
-First, -Last und -Skip	239
Berechnete Eigenschaften	239

Where-Object	240
Clientseitiger Universalfilter	241
Leere Elemente aussortieren	242
Fortgeschrittene Syntax bietet mehr Möglichkeiten	242
Out-GridView: das »menschliche« Where-Object	243
Sort-Object	244
Cmdlet-Ergebnisse sortieren	245
Sortierung mit anderen Cmdlets kombinieren	245
Datentyp der Sortierung ändern	246
Mehrere Spalten in umgekehrter Sortierung	247
Group-Object	247
Häufigkeiten feststellen	247
Daten gruppieren	248
Berechnete Gruppierungskriterien	250
Measure-Object	251
Statistische Berechnungen	252
Ordnergrößen berechnen	252
Foreach-Object	252
Grundprinzip: eine Schleife	252
Format-Cmdlets	253
Gefährlich: Format-Cmdlets verändern Objekte	253
Mehrspaltige Anzeigen	254
Tabellenausgabe mit Gruppierung	254
6 Arrays und Hashtables	257
Arrays verwenden	258
Auf Array-Elemente zugreifen	259
Eigene Arrays (und einige Fallen)	260
Automatische Array-Erzeugung	260
Manuelle Array-Erzeugung	262
Hashtables – »sprechende Arrays«	265
Hashtables in Objekte umwandeln	266
Neue Objekte mit Eigenschaften initialisieren	267
7 PowerShell-Laufwerke	269
Dateisystemaufgaben meistern	271
Cmdlets für das Dateisystem finden	271
Erste Schritte	272
Ordner anlegen	273
Dateien anlegen und Informationen speichern	275
Encoding von Textdateien	280
Encodings sichtbar machen	281
Dateien finden	284
Dateien und Ordner kopieren	288
Dateien umbenennen	291
Dateien und Ordner löschen	294
Größe eines Laufwerks ermitteln	295
Größe eines Ordners ermitteln	295

Umgebungsvariablen	297
Alle Umgebungsvariablen auflisten	297
Auf einzelne Umgebungsvariablen zugreifen	298
Umgebungsvariablen ändern	299
Windows-Registrierungsdatenbank	300
Schlüssel suchen	301
Werte lesen	301
Neue Registry-Schlüssel anlegen	303
Registry-Schlüssel löschen	304
Werte hinzufügen, ändern und löschen	305
Virtuelle Laufwerke und Provider	306
Neue PowerShell-Laufwerke	307
Ohne Laufwerksbuchstaben direkt auf Provider zugreifen	309
-Path oder -LiteralPath?	310
Existenz eines Pfads prüfen	310
Pfadnamen auflösen	311
8 Operatoren und Bedingungen	313
Operatoren – Aufbau und Namensgebung	314
Wie Operatornamen aufgebaut sind	315
Unäre Operatoren	315
Zuweisungsoperatoren	316
Vergleichsoperatoren	318
Ternary-Operator	320
Unterscheidung zwischen Groß- und Kleinschreibung	321
Unterschiedliche Datentypen vergleichen	322
Vergleiche umkehren	323
Vergleiche kombinieren	324
Vergleiche auf Arrays anwenden	325
Bedingungen	327
if-Bedingung	327
switch-Bedingung	328
Where-Object	330
Null-Koaleszenz-Operatoren	330
Pipeline-Verkettungsoperatoren	331
9 Textoperationen und reguläre Ausdrücke	335
Texte zusammenfügen	336
Doppelte Anführungszeichen lösen Variablen auf	337
Der Formatierungsoperator -f	338
Array-Elemente in Text umwandeln	345
Textstellen finden und extrahieren	347
Texte splitten	348
Informationen in Texten finden	350
Reguläre Ausdrücke: Textmustererkennung	352
Erste Schritte: Textmuster basteln	353
Eigene reguläre Ausdrücke konzipieren	360
Textstellen ersetzen	363
Einfache Ersetzungen	363
Sichere Ersetzungen mit -replace	364
Mehrere Zeichen durch eins ersetzen	364

Split und Join: eine mächtige Strategie	365
Dateipfade ändern	365
X500-Pfade auslesen	366
10 Anwendungen und Konsolenbefehle	367
Programme starten	370
Optionen für den Programmstart festlegen	372
Argumente an Anwendungen übergeben	374
Hilfe für Konsolenbefehle anzeigen	374
Beispiel: Lizenzstatus von Windows überprüfen	375
Ergebnisse von Anwendungen weiterverarbeiten	376
Error Level auswerten	376
Fragen an Benutzer stellen mit choice.exe	378
Rückgabertext auswerten	379
Laufende Programme steuern	381
Feststellen, ob ein Prozess läuft	381
Auf einen Prozess warten	381
Einstellungen laufender Prozesse ändern	382
Prozesse vorzeitig abbrechen	384
11 Typen verwenden	385
Typumwandlungen	387
Geeignete Datentypen auswählen	387
Explizite Umwandlung	388
Deutsches Datumsformat mit dem Operator -as	389
Verkettete Umwandlungen	390
Typen: optimale Informationsbehälter	391
Implizite Umwandlung	393
Typisierte Variablen	393
Parameter und Argumente	394
Vergleichsoperationen	395
Statische Methoden eines Typs verwenden	397
Dateiextension ermitteln	399
Mathematische Funktionen	400
Zahlenformate konvertieren	401
DNS-Auflösung	401
Umgebungsvariablen	402
Pfade zu Systemordnern finden	404
Konsoleneinstellungen	404
Spezielle Datumsformate lesen	405
Statische Eigenschaften verwenden	407
Neue .NET-Typen finden	408
Type Accelerators untersuchen	408
Typen nachladen	410
Assembly-Namen feststellen	410
Aktuell geladene Assemblies auflisten	410
Zusätzliche Assembly nachladen	411
Assembly aus Datei nachladen	411

12 Mit Objekten arbeiten	413
Objekte kennenlernen	414
Objekte funktionieren wie beschriftete Schubladen	415
Typisierte Variablen	418
Objekteigenschaften erforschen	418
Eigenschaften lesen	420
Eigenschaften ändern	422
Methoden verwenden	424
Zusammenfassende Systematik	427
Eigenschaften	427
Methoden	431
Hilfe finden	433
Neue Objekte herstellen	435
Konstruktoren verstehen	435
Ein Credential-Object zur automatischen Anmeldung	436
Eigene Objekte erstellen	438
13 Eigene Typen und Attribute	441
Typen (Klassen) selbst herstellen	442
Einen eigenen Typ erfinden	443
Neue Objekte eines Typs generieren	443
Konstruktoren hinzufügen	444
Methoden zum Typ hinzufügen	447
Vererbung von Typen	450
Konstruktoren werden nicht geerbt	450
Philips Hue Smart Home mit Typen steuern	453
Kontakt zur Philips Hue Bridge herstellen	454
Lampen- und Schalterinventar	455
Lampen und Steckdosen ansprechen	455
Attribute erstellen	456
SecureString-Transformationsattribute	456
Auf API-Funktionen zugreifen	459
API-Funktion einsetzen	459
Wiederverwertbare PowerShell-Funktion herstellen	460
14 Parameter für Fortgeschrittene	463
Argumentvervollständigung	465
Statische Autovervollständigung	465
Autovervollständigung über Enumerationsdatentyp	466
Eigene Enumerationsdatentypen erstellen	467
Autovervollständigung über ValidateSet	470
Dynamische Argumentvervollständigung	471
Zuweisungen mit Validierern überprüfen	476
ValidateSet	476
ValidateRange	477
ValidateLength	478
ValidatePattern	479
ValidateCount	479
ValidateScript	480
Nullwerte und andere Validierer	480

Parameter in ParameterSets einteilen	480
Gegenseitig ausschließende Parameter	481
Binding über Datentyp	481
Parameter in mehreren Parametersätzen	482
Simulationsmodus (-WhatIf) und Sicherheitsabfrage (-Confirm)	483
Festlegen, welche Codeteile übersprungen werden sollen	484
Weiterleitung verhindern	485
Gefährlichkeit einer Funktion festlegen	486
Dynamische Parameter	488
Dynamische Parameter selbst definieren	489
Splatting: Argumente an Parameter binden	491
Splatting im Alltag einsetzen	491
Übergebene Parameter als Hashtable empfangen	492
Mit Splatting Parameter weiterreichen	493
15 Pipeline-fähige Funktionen	497
Anonyme Pipeline-Funktion	498
Prototyping	499
Pipeline-fähige Funktion erstellen	499
Benannte Parameter	500
Where-Object durch eine Funktion ersetzen	501
Kurzes Resümee	503
Parameter und Pipeline-Kontrakt	503
ISA-Kontrakt: Pipeline-Eingaben direkt binden	504
HASA-Kontrakt: Objekteigenschaften lesen	507
HASA und ISA kombinieren	507
CSV-Dateien direkt an Funktionen übergeben	509
Aliasnamen für Parameter	510
16 PowerShellGet und Module	513
PowerShellGet – die PowerShell-Softwareverteilung	514
Grundlage »PackageManagement«	515
PowerShellGet – Softwareverteilung per Cmdlets	515
Wo PowerShell Module lagert	516
Unterschiede zwischen Windows PowerShell und PowerShell	517
Installierte Module untersuchen	518
Module mit PowerShellGet nachinstallieren	519
Module finden und installieren	519
Modul herunterladen	520
Modul testweise ausführen	521
Modul dauerhaft installieren	522
Module aktualisieren	523
Side-by-Side-Versionierung	523
Eigene Module veröffentlichen	523
Eigene Module herstellen	524
Neue Manifestdatei anlegen	524
Neue Moduldatei anlegen	527
Modul testen	528
Nächste Schritte	529

Eigene Module verteilen	530
Netzwerkfreigabe	530
Modul in Repository übertragen	530
Modul aus privatem Repository installieren	531
17 Fehlerhandling	533
Fehlermeldungen unterdrücken	535
Bestimmen, wie Cmdlets auf Fehler reagieren	535
Fehler mitprotokollieren lassen	536
Erfolg eines Befehlsaufrufs prüfen	538
Fehlerhandler einsetzen	538
Lokaler Fehlerhandler: try...catch	538
Globaler Fehlerhandler: Trap	542
18 Windows PowerShell-Remoting	545
PowerShell-Remoting aktivieren	547
Zugriff auch auf Windows-Clients	547
Remoting-Verbindung überprüfen	549
NTLM-Authentifizierung erlauben	549
PowerShell-Remoting überprüfen	551
Erste Schritte mit PowerShell-Remoting	552
Befehle und Skripte remote ausführen	554
Kontrolle: Wer besucht »meinen« Computer?	554
Remotefähigen Code entwickeln	555
Argumente an Remote-Code übergeben	555
Ergebnisse vom Remote-Code an den Aufrufer übertragen	557
Fan-Out: integrierte Parallelverarbeitung	558
ThrottleLimit: Parallelverarbeitung begrenzen	558
Double-Hop und CredSSP: Anmeldeinfos weiterreichen	560
Eigene Sitzungen verwenden	562
Eigene Sitzungen anlegen	562
Parallelverarbeitung mit PSSessions	563
SSH-basiertes Remoting	566
SSH-Remoting aktivieren	566
Enter-PSSession über SSH	567
Invoke-Command über SSH	568
Unbeaufsichtigte Ausführung und Parallelbearbeitung	568
19 Grafische Oberflächen gestalten	569
Eigene Fenster herstellen	570
GUI mit Code definieren	571
GUI mit XAML definieren	574
Beispiel: Dienstopper-Anwendung	577
Index	581