PowerShell for Beginners

Basics and Complex Exercises

Table of Contents

Cmdlets in PowerShell

Structure and Functionality

Cmdlets vs Function

- In programming languages (PL) there are functions.
  - Was a funktion actually is, may differ severely
  - Imperative PL (e.g. Java) vs. functional PL (e.g. Haskell)
- In PowerShell there are functions and Cmdlets.
- Cmdlets are identical with functions from a user‘s point of view
- From a developer‘s point of view there are differences:

Cmdlet vs Function

Cmdlet(s)

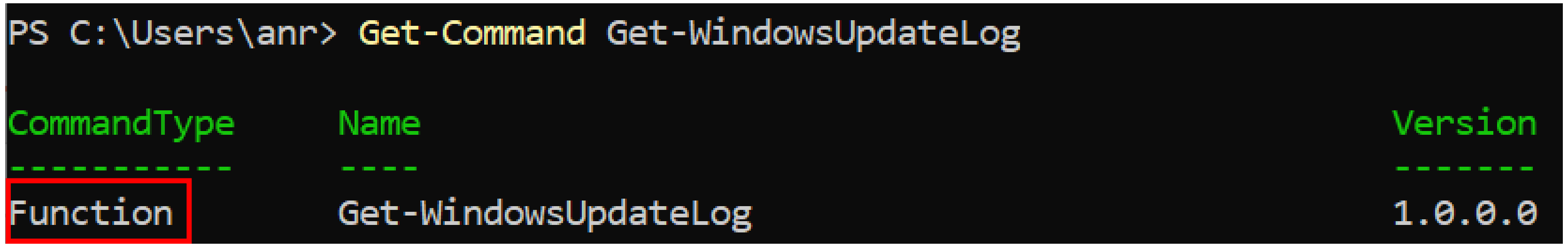
- is a .NET class
- written in C# (or another .NET language)
- available in binary format (encapsuled in a DLL)
- bundled in modules

Function(s)

- written in PowerShell Language
- can be defined directly at the prompt
- available in a text file (non binary)
- bundled in scripten
- Source code viewable through PSDrive *Function*

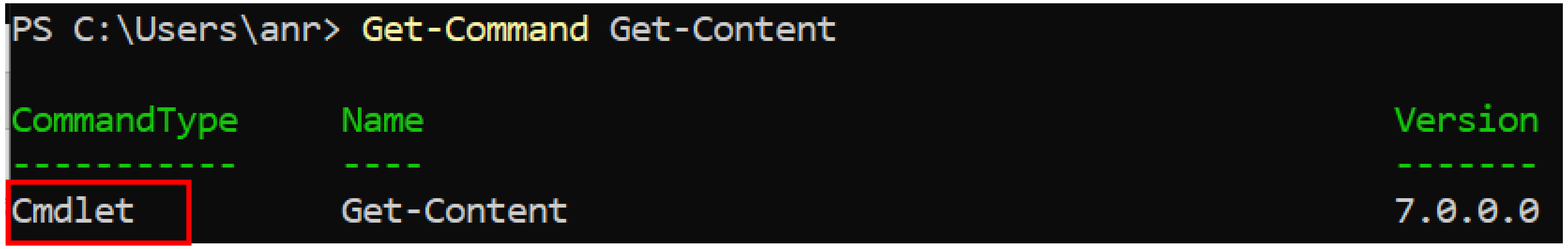
Cmdlet vs Function

- To check whether a command is a Cmdlet or a function, use the command `Get-Command` :



```
PS C:\Users\anr> Get-Command Get-WindowsUpdateLog

CommandType     Name                                               Version
-----------     ----                                               -------
Function        Get-WindowsUpdateLog                               1.0.0.0
```



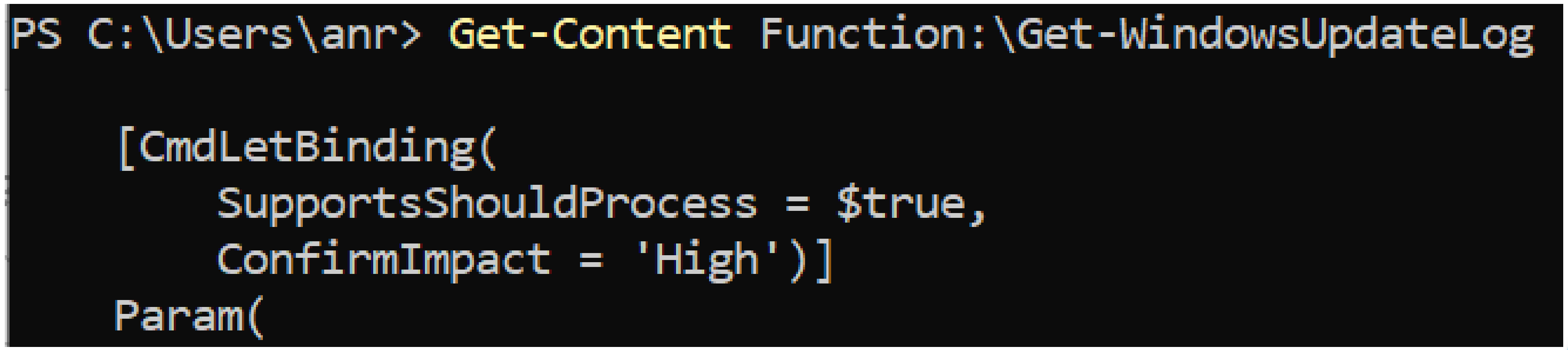
```
PS C:\Users\anr> Get-Command Get-Content

CommandType     Name                                               Version
-----------     ----                                               -------
Cmdlet          Get-Content                                        7.0.0.0
```

- Hands-on: What is the CommandType of `Get-Command` itself ?

Functions

- Source code of a function can be shown through the PSDrive *Functions*.
- Example: Function `Get-WindowsUpdateLog`



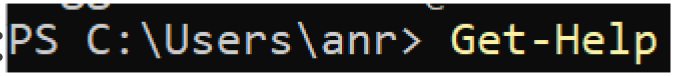
```
PS C:\Users\anr> Get-Content Function:\Get-WindowsUpdateLog

    [CmdLetBinding(
        SupportsShouldProcess = $true,
        ConfirmImpact = 'High')]
    Param(
```

- [Output omitted]

Cmdlets

- Non-function commands in der PowerShell are called Cmdlets (pronounced: Commandlets)
- Cmdlets usually return objects (not string streams)
- Cmdlets strictly follow a so-called Verb-Noun-Syntax:
  - Verb describes what action is perfomed on an object
  - Noun describes which object is affected by the action.

- Cmdlets are conventionally noted in a mixture of Pascal case and kebab case (not mandatory though)
- Example: 

```
PS C:\Users\anr> Get-Help
```

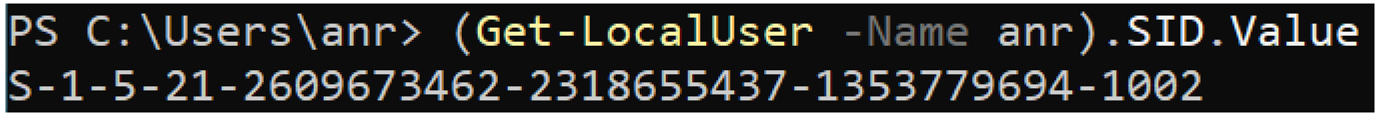
Cmdlets Access

- The Cmdlet return objects can be regarded as classical OOP objects
- They have got…
  - Properties, Sing. Property
  - Methods, Sing. Method

- Properties often have *getter* and sometimes *setter*
- Access to a property with dot notation:
  - `Objekt.Property`
  - syntactically like C# Property
  - no `getProperty()` like in Java

Cmdlet Access in Detail

- Example:
  - Print SID (Security Identifier) of the user *anr*



```
PS C:\Users\anr> (Get-LocalUser -Name anr).SID.Value
S-1-5-21-2609673462-2318655437-1353779694-1002
```

- `Get-LocalUser –Name anr` returns an object of type `Microsoft.PowerShell.Commands.LocalUser`
- `(Get-LocalUser –Name anr).SID` returns an object of type `System.Security.Principal.SecurityIdentifier`
- `(Get-LocalUser –Name anr).SID.Value` returns an object of type `String`

☞ Demo

Parameter of Cmdlets

Types, Structure and Functionality

Cmdlets Parameter

- Cmdlets can have parameters (most of them do)
- Parameter are preceded by – (Minus, Hyphen, Dash)
- Example:

```
Out-Host -Paging
```

(equivalent to `more` in the cmd.exe)

Recycle bin full?

```
Clear-RecycleBin -Force
```

- Online reference for all Cmdlets:

https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.management/?view=powershell-7.3

Sorts (Types) of Parameters

- There are four sorts (types) of parameters:

…and some subtypes and hybrids

1) Named Parameters
2) Positional Parameters
3) Switch Parameters
4) Common Parameters

Named Parameter

- Parameter has a namen (preceded by –) and value (Key-Value-Principle)
- Name must be unambiguous (but not necessarily complete)

- Example:

```
PS> Get-ChildItem –Path C:\Users\anr\Documents
```

- Name (Key): Path
- Wert (Value): C:\Users\anr\Documents

☞ Named Parameter are very common

Named Partial Parameter

- Incomplete named parameters are called *partial parameters*.
- Prefix length is arbitrary, but name must be unambiguous

- Examples:

```
PS> Get-ChildItem -Path C:\Users\anr\Documents
```
fully named ✓

```
PS> Get-ChildItem -Pa C:\Users\anr\Documents
```
partial ✓

```
PS> Get-ChildItem -P C:\Users\anr\Documents
```
partial ✗

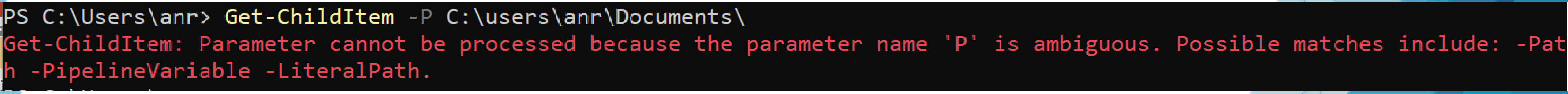
Named Partial Parameter

- Incomplete named parameters are called *partial parameters*.
- Prefix length is arbitrary, but name must be unambiguous

- Example:

```
PS> Get-ChildItem -P C:\Users\anr\Documents
```

partial (ambiguous)



```
PS C:\Users\anr> Get-ChildItem -P C:\users\anr\Documents\
Get-ChildItem: Parameter cannot be processed because the parameter name 'P' is ambiguous. Possible matches include: -Pat
h -PipelineVariable -LiteralPath.
```

Positional Parameter

- Like named parameter without the key, providing only a value
- More terse, but readability deteriorates

- Example:

```
PS> Copy-Item a b
```

  - What is source (Path) and what is destination?

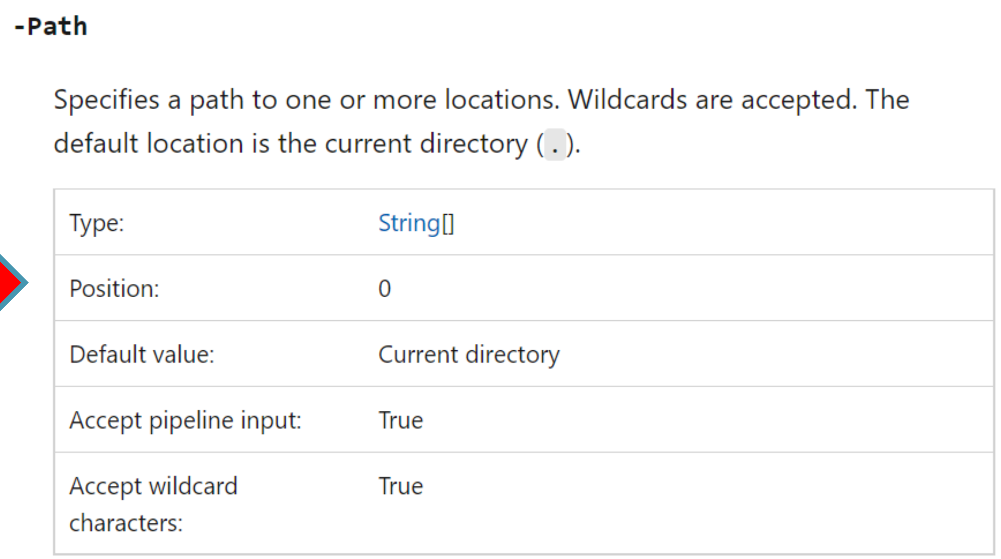
- Best Practice:
  - usually avoid positional parameters (especially when teaching beginners)
  - only to be user with easy to grasp Cmdlets, e.g. `Get-Process`
  - possible use them if a default parameter is obvious

Positional Parameter

- Example:

```
PS> Get-ChildItem C:\Users\anr\Documents
```

(`-Path` omitted)



-Path

Specifies a path to one or more locations. Wildcards are accepted. The default location is the current directory (`.`).

Type:	String[]
Position:	0
Default value:	Current directory
Accept pipeline input:	True
Accept wildcard characters:	True

☞ The expected position of the parameter is documented in the reference!

Switch Parameter

- Either turned on or off (just like a light switch)
- No value, activated upon naming, silent otherwise

- Example:

```
PS> Get-ChildItem -Path C:\users\anr\Documents -Name
```

- Activated: only displayes the names of the objects in the result set
- Not specified: further information is displayed as well

☞ A common switch parameter is `-Force`

Properties of Parameters

- Parameter may also…

have default values	None
	Default value as indicated in reference
Accept pipeline input	True / False
Accept wildcards	True / False

- Consult the reference or help to find the correct data type (String, UInt32, etc.) of a parameter

Common Parameter

- Available for everty Cmdlet
- Mainly for debugging or logging purposes

- Examples:

```
-ErrorAction: Break | Ignore | SilentlyContinue …

-Verbose
```

Risk Mitigation Parameters

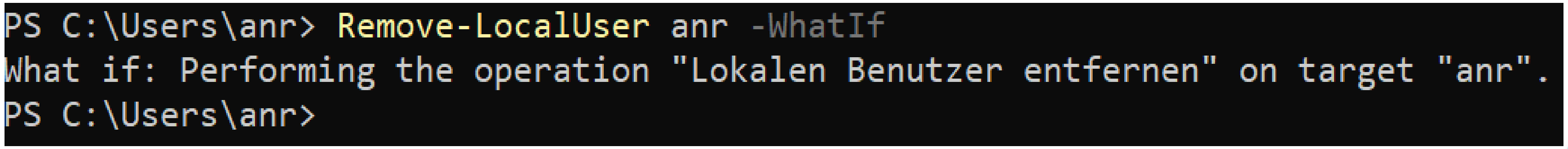
- *Risk mitigation parameters*: `-WhatIf`, `-Confirm`
  - available for many Cmdlets
  - useful for system changes, syntax checking, etc.
  - are essentially switch parameters

- `-WhatIf`: shows as text what happens if a command is executed
- `-Confirm`: requires explicit confirmation of an action via keystroke
- `-Confirm:$False` overwrites explicit confirmation requests
  - useful, if a Cmdlet does not implement `-Force` and something is to be automated, since user interaction is suppressed

Risk Mitigation Parameters

- Example:

Use risk mitigation parameter `-WhatIf`, before a user is acutally deleted.



```
PS C:\Users\anr> Remove-LocalUser anr -WhatIf
What if: Performing the operation "Lokalen Benutzer entfernen" on target "anr".
PS C:\Users\anr>
```

☞ What sort of parameter is *anr* in the above command?

(Self-)Help in PowerShell

Using built-in resources

Capacity Building

- "Give a man a fish, and you feed him for a day. Teach a man to fish, and you feed him for a lifetime." - Laotse

- If I don‘t know what to do anymore…
- …my friendly neighborhood Cmdlets will help me get along:
  - `Get-Help`
  - `Get-Command`
  - `Get-Member`

Cmdlets for Self Help

- `Get-Help` displays help for a (known) Cmdlet:
  - Structure
  - Available Parameters
- Use `–example` to display example usage of the Cmdlet
- Use `–full` to display the detailed help
- Example:

```
PS> Get-Help Get-ChildItem -example
```

Cmdlets for Self Help

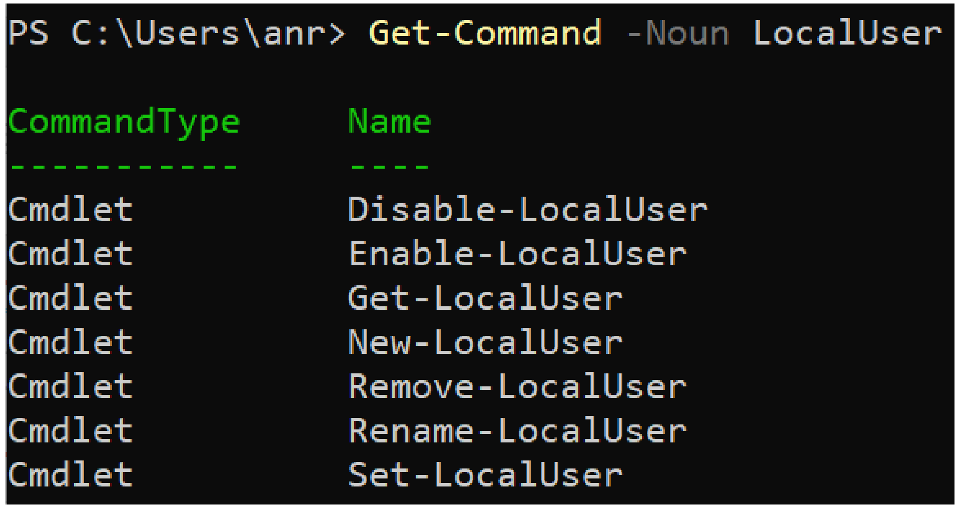
- `Get-Command` searches for a(n) (unknown) Cmdlet
  - `-Verb` specifies the action, the Cmdlet should perform
  - `-Noun` specifies the object (the family) of the Cmdlet

- Parameters may be combined
- Wildcards are accepted for the two of them

Usage of `Get-Command`

- Example:
  - Display all Cmdlets to administrate local users

```
PS> Get-Command –Noun LocalUser
```



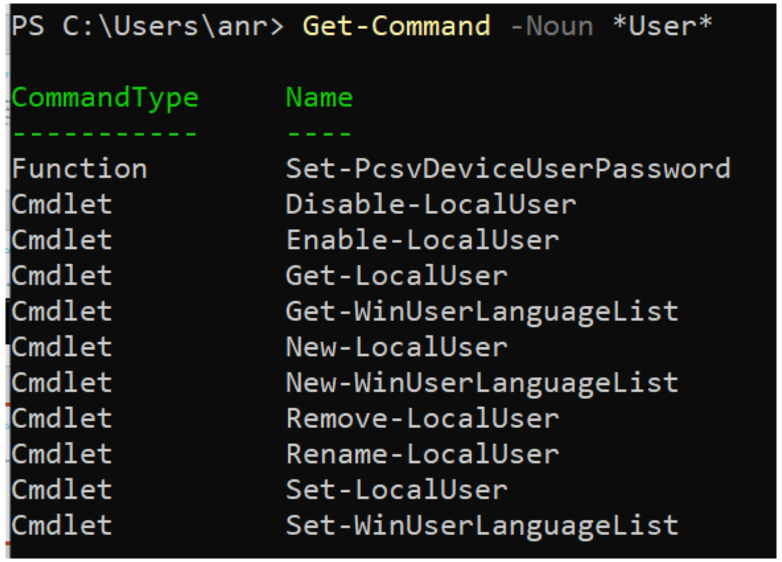
```
PS C:\Users\anr> Get-Command -Noun LocalUser

CommandType     Name
-----------     ----
Cmdlet          Disable-LocalUser
Cmdlet          Enable-LocalUser
Cmdlet          Get-LocalUser
Cmdlet          New-LocalUser
Cmdlet          Remove-LocalUser
Cmdlet          Rename-LocalUser
Cmdlet          Set-LocalUser
```

Usage of `Get-Command`

- Example:
  - Display all Cmdlets dealing with users in general

```
PS> Get-Command –Noun *User*
```



```
PS C:\Users\anr> Get-Command -Noun *User*

CommandType     Name
-----------     ----
Function        Set-PcsvDeviceUserPassword
Cmdlet          Disable-LocalUser
Cmdlet          Enable-LocalUser
Cmdlet          Get-LocalUser
Cmdlet          Get-WinUserLanguageList
Cmdlet          New-LocalUser
Cmdlet          New-WinUserLanguageList
Cmdlet          Remove-LocalUser
Cmdlet          Rename-LocalUser
Cmdlet          Set-LocalUser
Cmdlet          Set-WinUserLanguageList
```

Exercise PS21
PowerShell Self Help

- Get to know the Cmdlets `Get-Help` and `Get-Command`
- Einfache Cmdlets ohne Parameter ausführen
- Cmdlets mit Named Parameter verwenden (Syntaxgewöhnung)