Advanced Scripting   
Comparison and Logical Operators

Last Updated: 5/11/2024 6:44 PM Version 1  
Document Prepared for: CYBER360 Student

# Name Click here to enter name ID Click here to enter id

# Instructions

Answer all questions directly in this document. You will save and upload this completed document as your homework submission.

# Overview

This exercise presents a few practice drills involving various PowerShell comparison operators. Each of these return a **[bool]** (Boolean) object, either **True** or **False**.

# Setup

## Requirements

* PowerShell

# Task 1—Ordinal Comparison Operators

The “ordinal” or “lattice” comparison operators are:

|  |  |  |  |
| --- | --- | --- | --- |
| Operator (Default) | Case  Sensitive version | Case  Insensitive version | Meaning |
| -eq | -ceq | -ieq | Equal |
| -ne | -cne | -ine | Not equal |
| -gt | -cgt | -igt | Greater than |
| -ge | -cge | -ige | Greater than or equal to |
| -lt | -clt | -ilt | Less than |
| -le | -cle | -ile | Less than or equal to |

## Steps

1. For each of the expressions below, first predict what you think the value will be ($true or $false). Then type the expression in PowerShell. Record the result. Describe the result, and if it’s different than what you predicted, explain *how* the operator behaved differently.

|  |  |  |  |
| --- | --- | --- | --- |
| Prediction | Expression | Result | Description |
| Click or tap here to enter text. | **'' -eq $null** | Click or tap here to enter text. | Click or tap here to enter text. |
| Click or tap here to enter text. | **'red' -eq 'red'** | Click or tap here to enter text. | Click or tap here to enter text. |
| Click or tap here to enter text. | **'red' -eq 'Red'** | Click or tap here to enter text. | Click or tap here to enter text. |
| Click or tap here to enter text. | **'red' -ceq 'Red'** | Click or tap here to enter text. | Click or tap here to enter text. |
| Click or tap here to enter text. | **1 -gt 1** | Click or tap here to enter text. | Click or tap here to enter text. |
| Click or tap here to enter text. | **12 -gt '2'** | Click or tap here to enter text. | Click or tap here to enter text. |
| Click or tap here to enter text. | **'12' -gt 2** | Click or tap here to enter text. | Click or tap here to enter text. |
| Click or tap here to enter text. | **3 -gt '11'** | Click or tap here to enter text. | Click or tap here to enter text. |
| Click or tap here to enter text. | **'3' -gt 11** | Click or tap here to enter text. | Click or tap here to enter text. |

1. When the left-hand operator is a collection (e.g. array), each item is compared to the right-hand operand, and if the comparison is true the item is passed along the pipeline.
   1. Try the following  
      1,2,3,4 -ge 2
      1. What is returned?Click or tap here to enter text.
   2. Try  
      'a','b','c','a' -lt 'c'
      1. What is returned? Click or tap here to enter text.

# Task 2—-Like and -NotLike, Basic Wildcard Pattern Matching

-like and -notlike use simple wildcard pattern matching. The wildcard characters are:

|  |  |  |
| --- | --- | --- |
| **Wildcard** | **Matches** | **Examples** |
| \* | Match any sequence of characters | \*.log |
| ? | Match any single character | 1225????.log |
| [<char>-<char>] | Match one of a range of characters | SiO2[1-6][a-b] |
| [<char><char>] | Match one of a specified set characters | [02468][ace]xyz |

## Steps

1. Many cmdlets allow basic wildcard matching for parameters. **Get-Command** is a good example.
   1. To get a list of all the commands that start with New use the argument **New-\***  
      get-command New-\*
   2. To get all commands that contain the word *process*, wrap **process** between **\*** splat wildcards:  
      get-command \*process\*
   3. Get all the commands that start with *Get-* and the noun starts with the letters a through p:  
      Get-Command Get-[a-p]\*
2. The **-like** and **-notlike** operators can be used wherever you need a comparison. Given the following, predict, then verify and record the expression result with PowerShell

|  |  |  |
| --- | --- | --- |
| Expression | Prediction | Result |
| **'red' -like 'r\*'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **'red' -like 'R\*'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **'red' -clike 'R\*'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **'red' -notlike 'red'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **8675309 -like '[5-9]??53\*'** | Click or tap here to enter text. | Click or tap here to enter text. |

# Task 3— -Match and -NotMatch, Regular Expression Matching

The **-match** operator uses regular expression (“regex”) matching. Regular expressions are a more complex pattern matching system than the wildcards used by **-like** and **-notlike**. To learn about regular expressions there are numerous tutorials and references online.

## Steps

1. The simplest and probably the most used regular expressions determine whether a string contains a specific sequence of characters. Get familiar with it:

|  |  |  |
| --- | --- | --- |
| Expression | Prediction | Result |
| **'99 red balloons' -match 'Red'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **'99 red balloons' -notmatch 'Red'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **'99 red balloons' -cmatch 'Red'** | Click or tap here to enter text. | Click or tap here to enter text. |

1. A regex “or” operation has alternative strings between parentheses and separated by | vertical bars. Use regex “or” alternatives to match one among a list words:

|  |  |  |
| --- | --- | --- |
| Expression | Prediction | Result |
| **'99 red balloons' -match '(red|blue|green)'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **'99 orange balloons' -match '(red|blue|green)'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **'99 blue balloons' -cmatch '(red|blue|green)'** | Click or tap here to enter text. | Click or tap here to enter text. |

1. Square brackets **[ ]** can be used to specify ranges or sets of single characters to be matched. These patterns are identical to the corresponding character wildcards of the **-like** operator:

|  |  |  |
| --- | --- | --- |
| Expression | Prediction | Result |
| **'99 blue balloons' -match '[0-9]'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **'99 blue balloons' -match '[0-9][0-9]'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **'99 blue balloons' -match '[0-9][0-9][0-9]'** | Click or tap here to enter text. | Click or tap here to enter text. |

1. Special regex *anchors* ^ and $ don’t match a character, but a *position* instead: ^ matches the beginning of a string, $ matches the end of a string:

|  |  |  |
| --- | --- | --- |
| Expression | Prediction | Result |
| **'99 blue balloons' -match '^[0-9]'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **'99 blue balloons' -match '^blue'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **'99 blue balloons' -match 'balloon$'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **'99 blue balloons' -match 'loons$'** | Click or tap here to enter text. | Click or tap here to enter text. |

1. Regex wildcards:
   1. The period **.** is the regex wildcard that matches any single character (analogous to the question mark **?** wildcard of the **-like** operator).
   2. The two-character sequence “dot-splat” **.\*** is the regex wildcard that matches any sequence of any characters (analogous to the splat **\*** wildcard of the **-like** operator).
   3. Any other character, followed by a splat **\***, is a regex wildcard that matches any repeated sequence of that character only. Example: the regex pattern **'=\*'** matches any of **''**, **'='**, **'=='**, **'==='**, etc.

|  |  |  |
| --- | --- | --- |
| Expression | Prediction | Result |
| **'99 red balloons' -match '99 r.d b......s'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **'99 blue balloons' -match 'blue.\*loon'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **'99 red balloons' -match '^9\*'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **'99 red balloons' -match '99 red [a-z]\*'** | Click or tap here to enter text. | Click or tap here to enter text. |

1. To match a specific character, like dot **.**, splat **\***, hat **^**, or dollar **$**, it must be “escaped” by prefixing with a backslash (**\.** or **\\*** or **\^** or **\$**) so that they are not misinterpreted as wildcards.

|  |  |  |
| --- | --- | --- |
| Expression | Prediction | Result |
| **'Price\* $9.90' -match '\$9\.90$'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **'Price\* $9.90' -match '^Price\\* '** | Click or tap here to enter text. | Click or tap here to enter text. |

# Task 4—Containment Operators

Use either **-contains** or **-notcontains** to check whether an array (or other collection object) contains a value.

Use either **-in** or **-notin** to check whether a value is present in an array (or other collection object).

## Steps

1. Make a couple of arrays:   
     
   **$evens = 1..99 | ? {$\_ % 2 -eq 0}**   
   **$quartet = 'violin', 'violin', 'viola', 'cello'**
2. Try some containment expressions.

|  |  |  |
| --- | --- | --- |
| Expression | Prediction | Result |
| **$evens -contains 44** | Click or tap here to enter text. | Click or tap here to enter text. |
| **$evens -contains 77** | Click or tap here to enter text. | Click or tap here to enter text. |
| **$evens -contains '6'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **$evens -contains 'six'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **66 -in $evens** | Click or tap here to enter text. | Click or tap here to enter text. |
| **'two' -notin $evens** | Click or tap here to enter text. | Click or tap here to enter text. |
| **$quartet -contains 'harp'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **$quartet -notcontains 'viola'** | Click or tap here to enter text. | Click or tap here to enter text. |
| **'violin' -in $quartet** | Click or tap here to enter text. | Click or tap here to enter text. |

# Task 5—Logical Operators

Fill in the following logical truth tables. For example, to fill out the table in step 1, get results of these:  
$true -and $true   
$true -and $false   
$false -and $true   
$false -and $false

## Steps

1. For the -and operator to return True both operands must be True.

|  |  |  |
| --- | --- | --- |
| -and | $true | $false |
| $true | Click or tap here to enter text. | Click or tap here to enter text. |
| $false | Click or tap here to enter text. | Click or tap here to enter text. |

1. For the -or operator, at least one of the operands must be True. (Think “left or right or both.”)

|  |  |  |
| --- | --- | --- |
| -or | $true | $false |
| $true | Click or tap here to enter text. | Click or tap here to enter text. |
| $false | Click or tap here to enter text. | Click or tap here to enter text. |

1. For the -xor operator, one operand must be True and the other operand must be False. (Think “left or right but not both.”)

|  |  |  |
| --- | --- | --- |
| -xor | $true | $false |
| $true | Click or tap here to enter text. | Click or tap here to enter text. |
| $false | Click or tap here to enter text. | Click or tap here to enter text. |

1. The -not operator is a unary prefix operator. It returns the inverse of its operand.

|  |  |
| --- | --- |
|  | -not |
| $true | Click or tap here to enter text. |
| $false | Click or tap here to enter text. |

# Deliverable

Upload this document with completed answers to I-Learn Canvas.