Advanced Scripting   
Arithmetic Operators

Last Updated: 1/2/2024 1:11 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 PowerShell operators.

# Setup

## Requirements

* PowerShell
* The Windows Calculator app (or a similar calculator app)

# Task 1—Arithmetic Operators

## Steps

1. The **+** (addition/concatenation) operator:
   1. Defined for numbers, strings, and arrays. When both numbers and strings of differing data types are used, the right-hand operand will be converted to the left-hand type. Widening of the data type is performed if necessary. Try the following expressions, then record the returned value and type. If the expression produces an exception message, record the error.

|  |  |  |  |
| --- | --- | --- | --- |
| **Expression** | **Value** | **Type** | **Error if any** |
| 1+2 | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |
| 1 + '2' | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |
| 1 + 'two' | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |
| 1 + '2.5' | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |
| '1'+ 2 | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |
| 'blue' + 'green' | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |
| 1,2,3 + 4 | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |
| 1,2 + 2,3 | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |

1. The **\*** operator:
   1. Defined for numbers, strings, and arrays. For both numbers and strings, if differing data types are used, the right-hand operand will be converted to the left-hand type. Widening of the data type is performed if necessary. Try the following expressions, then record the returned value and type. If an exception message is produce, record the error.

|  |  |  |  |
| --- | --- | --- | --- |
| **Expression** | **Value** | **Type** | **Error if any** |
| 1\*2 | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |
| 1 \* '2' | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |
| 1 \* '2.5' | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |
| '1' \* 2 | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |
| 1 \* 'two' | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |
| 'red' \* 2 | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |
| 1,2,3 \* 2 | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |
| 1,2 \* 2,3 | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |

1. **–** (subtraction) **/** (division) and **%** (modulus).
   1. Defined only for numbers. If no number data types are used, PowerShell will attempt to convert to numbers. Widening of the data type is performed if necessary. Try the following expressions, then record the returned value and type. Record the error if one occurs.

|  |  |  |  |
| --- | --- | --- | --- |
| **Expression** | **Value** | **Type** | **Error if any** |
| 1 - 2 | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |
| 1 / '2' | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |
| '10'-'5' | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |
| '5' % 2 | Click or tap here to enter text. | Click or tap here to enter text. | Click or tap here to enter text. |

# Task 2—Assignment Operators

The **=** (assignment) operator is used to assign a value, or the results of an expression or pipeline, to a variable. If a variable does not yet exist, the assignment operator creates it.

## Steps

1. Basic assignment. Enter:  
   $a=10
   1. Enter **$a**
   2. Enter **Get-Variable a**
   3. What value is in $a? Click or tap here to enter text.
2. Assignment as an expression result  
   $b=1gb/1mb
   1. What value is in $b? Click or tap here to enter text.
3. Assignment as the result of a pipeline expression  
   $c=gps|measure|select count
   1. What value is in $c? Click or tap here to enter text.
4. Multiple assignments in one expression  
   **$d = $d = $f = 100**
   1. What is in **$d, $e, $f**? Click or tap here to enter text.
   2. Another   
      $g = ($h = ($i = 10) + 4) \* 2
   3. What are the values of **$g, $h, $i**? Click or tap here to enter text.
5. You can combine an arithmetic operators to the assignment operator to perform a reassignment of the variable to its current contents with the operator and operand applied. Example: $v+=1 is the same as **$v = $v + 1**. As you proceed, record each subsequent assigned result:

|  |  |
| --- | --- |
| **Enter** | **What is the value of $j** |
| $j=10 | Click or tap here to enter text. |
| $j\*=2 | Click or tap here to enter text. |
| $j-=20 | Click or tap here to enter text. |
| $j+=1000 | Click or tap here to enter text. |
| $j/=10 | Click or tap here to enter text. |
| $j%=10 | Click or tap here to enter text. |

1. You can also assign multiple variables to an array of values.  
   $k,$l,$m = 10,'red',(get-date)
   1. What is the value in $k? Click or tap here to enter text.
   2. What is the value in $l? Click or tap here to enter text.
   3. What is the value in $m? Click or tap here to enter text.
2. Easily swap values  
   $n='n'  
   $o='o'  
   $n,$o=$o,$n
   1. What is the value of $n? Click or tap here to enter text.
   2. What is the value of $o? Click or tap here to enter text.

# Task 3— Increment ++ and Decrement -- Unary Operators

The increment and decrement operators are used to add 1 or subtract 1 from a variable and store the results back in the original variable. They can be placed before or after the variable. If placed before the variable the adjustment is done prior to evaluating the expression. If placed after, it happens after evaluating the expression.

## Steps

1. Start with numeric variables  
   $a,$b=100,200  
   $a++
   1. What was output?Click or tap here to enter text.
   2. What is the current value of **$a**? Click or tap here to enter text.
2. Pre-increment (increment before)  
   Write-Host (++$a)
   1. What was output?Click or tap here to enter text.
   2. What is the current value of **$a**? Click or tap here to enter text.
3. Post-increment (increment after)  
   Write-Host ($b--)
   1. What was output?Click or tap here to enter text.
   2. What is the current value of **$b**? Click or tap here to enter text.

# Task 4—+ and - Unary Operators

The **+** and **-** unary operators convert the operand to a number, if it is not a numeric type already. The **-** operator then multiplies the number by -1

## Steps

1. The unary + operator is useful to convert a string to a number when you are not sure of the type you need.
   1. Try:  
      $a=+'123'
      1. What datatype is in **$a**?Click or tap here to enter text. *Hint*: **$a.GetType().Name**
      2. What is the current value of **$a**? Click or tap here to enter text.
   2. Try:  
      $b=+'123.5'
      1. What datatype is in **$b**?Click or tap here to enter text.
      2. What is the current value of **$b**? Click or tap here to enter text.
   3. Try:  
      $a=+'2345678901'
      1. What datatype is in $a?Click or tap here to enter text.
      2. What is the current value of $a? Click or tap here to enter text.
2. The **-** unary operator converts the operand to a number if necessary, then changes its sign.by multiplying it by -1
   1. Try:  
      $a=-'-123.5'
      1. What datatype is in $a?Click or tap here to enter text.
      2. What is the current value of $a? Click or tap here to enter text.

# Task 5—[void] operator

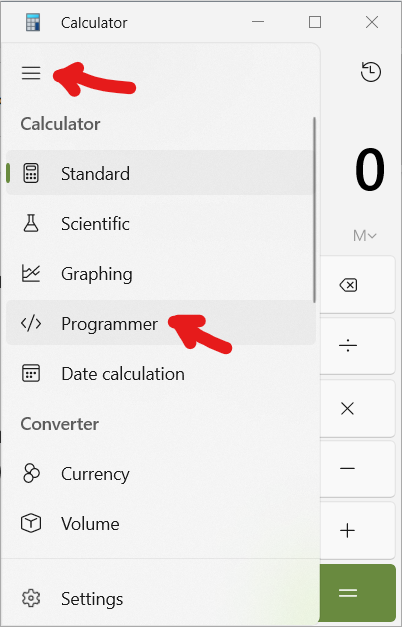
Because of the square brackets, it looks like an object class or data type, but isn’t. The **[void]** operator can be placed in front of any expression to have the results discarded.

## Steps

1. Enter  
   Write-Output 'hi'
   1. What was the output? Click or tap here to enter text.
2. Enter  
   [void](Write-Output 'hi')
   1. What was the output? Click or tap here to enter text.
3. Enter  
   $a = Write-Output 'hi'
   1. What is the current value of **$a**? Click or tap here to enter text.
4. Enter  
   $a = [void](Write-Output 'hi')
   1. What is the current value of **$a** Click or tap here to enter text. (Hint: evaluate the expression **$null -eq $a**, then interpret your result.)

# Task 6—Bitwise Arithmetic Operators

In Microsoft Windows, launch the Calculator app. You can look for it in the start menu, or better yet, just enter **calc** at your PowerShell prompt. Then tap the **≡** (“hamburger” menu) in the top left corner, and change the calculator to “Programmer” mode. Of its four input methods HEX, DEC, OCT, and BIN, ensure that DEC (“decimal”) is activated, as illustrated here:

 A screenshot of a calculator

Description automatically generated

## Steps

1. Bitwise and
   1. At your powershell prompt, Enter  
      85 -band 170
   2. What was the output? Click or tap here to enter text.
   3. In your calculator app in Programmer mode, tap **CE** (clear entry) or **C** (clear), then enter **85**.
      1. What is 85 in hexadecimal, as shown beside HEX in the Calculator app? Click or tap here to enter text.
      2. What is 170 in binary, as shown beside BIN in the Calculator app? Click or tap here to enter text.
   4. In your calculator app in Programmer mode, tap **CE**, then enter **170**.
      1. What is 170 in hexadecimal, as shown beside HEX in the Calculator app? Click or tap here to enter text.
      2. What is 170 in binary, as shown beside BIN in the Calculator app? Click or tap here to enter text.
   5. Based on the binary numerals for 85 and 170, explain in your own words how the result you recorded in step 1.2 above was computed. Click or tap here to enter text.
   6. Verify that the Calculator app produces the same result:
      1. tap **C**,
      2. enter 85,
      3. tap **Bitwise,** **AND**,
      4. enter 170,
      5. then tap **=**. Your result: Click or tap here to enter text.
2. Bitwise or, bitwise xor
   1. At your powershell prompt, Enter  
      (204 -bor 170), (204 -bxor 170)
   2. What was the output? Click or tap here to enter text.
   3. Use the calculator app in programmer mode to find 204 in hex and binary. Based on the binary numerals for 204 and 170, explain in your own words how the results you recorded in step 2.2 above were computed. Click or tap here to enter text.
   4. Verify that the Calculator app produces the same results: tap **CE**, enter 204, tap **Bitwise,** **OR**, enter 170, then tap **=**. Repeat for Bitwise XOR.
3. Bitwise not
   1. At your powershell prompt, Enter  
      -bnot [long]3
   2. What was the output? Click or tap here to enter text.
   3. What datatype is the result? Click or tap here to enter text. *Hint: put the expression in parentheses, then call the* GetType() *method*: **(-bnot [long]3).GetType().name** . *Or you can save the result to a variable, and access the* GetType() *method on that variable.*
   4. Verify in your calculator app as follows: make sure the data type is set QWORD. (If it’s set to DWORD, WORD, or BYTE, tap that setting until it’s back to QWORD.) Then **CE**, 3, **Bitwise**, **NOT**. Examine the hex and binary numerals for your result. Explain in your own words how the result you recorded in step 3.2 above was computed. Click or tap here to enter text.
4. Bit shifting
   1. At your powershell prompt, Enter  
      (3 -shl 2, 170 -shr 1)   
      The expressions in this array are 3 shifted left twice and 170 shifted right once, respectively.
   2. Your output: Click or tap here to enter text.
   3. Verify in your calculator app:
      1. Tap the Bit Shift menu and make sure it’s set to either arithmetic or logical shift:   
         A screenshot of a computer

         Description automatically generated
      2. **CE**, **3**, **«**, **2**, **=**. Examine HEX and BIN numeral results.
      3. **CE**, **170**, **»**, **1**, **=**. Examine HEX and BIN numeral results.
   4. Based on the binary numerals for 3, 170, and the results you recorded in step 4.2 above, how were those bit-shifted results computed? Explain in your own words. Click or tap here to enter text.

# Deliverable

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