Advanced Scripting   
Operators - Unary and Type

Last Updated: 9/30/2021 11:19 AM Version 1  
Document Prepared for: CIT361 Student

# Name Ryan Skidmore ID 615907798

# Instructions

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

# Requirements

PowerShell

# Task 1— increment ++ and decrement -- 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 place before or after the variable. If placed before the variable the increment is done prior to evaluating the expression. If placed after, it is done after the expression.

## Steps

1. Start with a numeric variable  
   $a=100  
   $a++
   1. What was output?[blank]
   2. What is the current value of $a? 101
2. Increment before  
   Write-Host (++$a)
   1. What was output?102
   2. What is the current value of $a? 102
3. Increment after  
   Write-Host ($a++)
   1. What was output?102
   2. What is the current value of $a? 103

# Task 2—+ - Unary Operators

The + and - unary operators convert the operand to a number, if it is not 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?int32
      2. What is the current value of $a? 123
   2. Try:  
      $a=+'123.5'
      1. What datatype is in $a?double
      2. What is the current value of $a? 123.5
   3. Try:  
      $a=+'123456789'
      1. What datatype is in $a?inc32
      2. What is the current value of $a? 123456789
2. The - unary operator converts the operand to a number if necessary, then it multiplies by -1
   1. Try:  
      $a=-'-123.5'
      1. What datatype is in $a?double
      2. What is the current value of $a? -123.5

# Task 3—[void] operator

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? hi
2. Enter  
   [void](Write-Output 'hi')
   1. What was the output? [blank]

# Task 4—Type operators -is -isnot and -as

The type operators are used to test a datatype or convert datatypes

## Steps

1. Test a variable’s type with code.
   1. Use the -is or -isnot operator to test a variable’s type  
      $a=100  
      $a -is [int]
      1. What was returned? True
   2. The variable type can also be a string that contains the variable’s type  
      $a=100  
      $type='string'  
      $a -is $type
      1. What was returned? false
2. Convert types with the -as operator. The advantages to using the **-as** operator is that a $null is returned on failure rather than an error. You can also specify the name as a string like with the **-is** operator.
   1. Try the **-as** operator  
      $a=100  
      $type='string'  
      $b=$a -as $type
      1. What is the value of $b? 100
      2. What datatype is $b String
   2. Modify the value in $b  
      $b+='yellow'
      1. What is the value of $b? 100yellow
      2. What datatype is $b String
   3. Change $b’s type to int  
      $b=$b -as [int]
      1. What is the value of $b? [null]
      2. What datatype is $b You cannot call a method on a null-valued expression.  
         At line:1 char:1  
         + $b.gettype()  
         + ~~~~~~~~~~~~  
          + CategoryInfo : InvalidOperation: (:) [], RuntimeException  
          + FullyQualifiedErrorId : InvokeMethodOnNull

# Deliverable

Upload this document with completed answers to i-learn.