Advanced Scripting   
Terminating Errors

Last Updated: 9/28/2021 8:29 AM Version 1  
Document Prepared for: CIT361 Student

# Name Zach Lucas ID 895649438

# 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.

# Overview

Terminating errors are errors that will stop script execution for the command. Terminating errors are handled with a try/catch statement. You will write a couple of try/catch blocks of code to get familiar with them.

# Requirements

PowerShell

# Task 1—Try Catch

## Steps

1. Create a file named term.ps1 in your favorite script editor.
2. Write the following code.

$dividend=Read-Host "Enter a dividend: "

$divisor=Read-Host "Enter a divisor: "

$quotient=$Dividend/$divisor

Write-Host "The answer my friend is $quotient" -f green

* 1. Run the script entering **10** for the dividend and **2** for the divisor.
     1. What is the output?5
  2. Run the script entering **10** for the dividend and **0** for the divisor.
     1. What was the output? There was an error because it didn’t like me dividing by 0
  3. Run the script entering **10** for the dividend and **foo** for the divisor.
     1. What was the output? An error because it can’t convert “foo” to an integer.

1. Modify the code to add a try catch block around the division

$dividend=Read-Host "Enter a dividend: "

$divisor=Read-Host "Enter a divisor: "

try{

    $quotient=$Dividend/$divisor

    Write-Host "The answer my friend is $quotient" -f green

}catch{

    Write-Host "You can't divide by 0." -f red

}

* 1. Run the script, entering **10** for the dividend and **2** for the divisor.
     1. What is the output? The answer my friend is 5
  2. Run the script, entering **10** for the dividend and **0** for the divisor.
     1. What was the output? You can’t divide by 0.
     2. Did you get an error? No error but the program ended
  3. Run the script, entering **10** for the dividend and **foo** for the divisor.
     1. What was the output? You can’t divide by 0.
     2. Does that make sense? It doesn’t but the program is told to output that whenever an error is caught

1. Let’s see if we can handle different kinds of errors. Modify your code as follows, adding a specific exception for the divide by 0 case. Then modify the default catch case.

$dividend=Read-Host "Enter a dividend: "

$divisor=Read-Host "Enter a divisor: "

try{

    $quotient=$Dividend/$divisor

    Write-host "The answer my friend is $quotient" -f green

}catch [System.DivideByZeroException]{

    Write-host "You can't divide by 0." -f red

}catch{

    Write-Host "Something went wrong." -f red

}

* 1. Run the script, entering **10** for the dividend and **2** for the divisor.
     1. What is the output? The answer my friend is 5
  2. Run the script, entering **10** for the dividend and **0** for the divisor.
     1. What was the output? You can’t divide by 0.
  3. Run the script, entering **10** for the dividend and **foo** for the divisor.
     1. What was the output? Something went wrong.

1. Clean things up by including a finally block and adding some more information for the user in the other error case. The $\_ intrinsic variable in the catch block will contain the exception that caused the error. You can access its properties to understand the error in more detail. Modify your code as follows:

$dividend=Read-Host "Enter a dividend: "

$divisor=Read-Host "Enter a divisor: "

try{

    $quotient=$Dividend/$divisor

    Write-host "The answer my friend is $quotient" -f green

}catch [System.DivideByZeroException]{

    Write-host "You can't divide by 0." -f red

}catch{

    Write-Host "Something went wrong." -f red

    $\_.Exception.Message

}finally{

    Write-Host "Thanks for doing math." -f Yellow

}

* 1. Run the script entering **10** for the dividend and **2** for the divisor.
     1. What is the output? The answer my friend is 5 Thanks for doing math.
  2. Run the script entering **10** for the dividend and **0** for the divisor.
     1. What was the output? You can’t divide by 0. Thanks for doing math.
  3. Run the script entering **10** for the dividend and **foo** for the divisor.
     1. What was the output? Cannot convert value “foo” to type “System.Int32”. Error: “Input string was not in a correct format.” Thanks for doing math.

# Task 2—Making your own exceptions.

Sometimes you catch an exception, but you can’t handle it properly in your code. In that case you could create and throw your own exception.

## Steps

1. Modify your code to throw an exception whenever an error other than divide by zero occurs.

$dividend=Read-Host "Enter a dividend: "

$divisor=Read-Host "Enter a divisor: "

try{

    $quotient=$Dividend/$divisor

    Write-host "The answer my friend is $quotient" -f green

}catch [System.DivideByZeroException]{

    Write-host "You can't divide by 0." -f red

}catch{

    Throw

}finally{

    Write-Host "Thanks for doing math." -f Yellow

}

* 1. Run the script, entering **10** for the dividend and **foo** for the divisor.
     1. What was the output? That there was a conversion error with foo and converting it to an integer
  2. Did the finally block run? It did but above the error

1. You can throw a custom message as well. Modify your code as follows

$dividend=Read-Host "Enter a dividend"

$divisor=Read-Host "Enter a divisor"

try{

    $quotient=$Dividend/$divisor

    Write-host "The answer my friend is $quotient" -f green

}catch [System.DivideByZeroException]{

    Write-host "You can't divide by 0." -f red

}catch{

    Throw "something went wrong"

}finally{

    Write-Host "Thanks for doing math." -f Yellow

}

* 1. Run the script entering **10** for the dividend and **foo** for the divisor.
     1. What was the output? Thanks for doing math, throw “something went wrong”

# Deliverable

Upload this document with completed answers to i-learn.