# CIS 218 Lab 06: Sessions, script files, regular expressions (40 points)

Please leave the lab questions/instructions/rubrics/etc. in place. Just paste your screenshots and code below my instructions.

Purpose/knowledge/skills: .PS1 script files add capabilities to PowerShell pipelines, by providing documentation and letting users specify parameters. Regular expressions are used to match input strings for a variety of purposes. You’ll work with both of these in this lab.

PowerShell administrators may also use PSSessions to maintain connections to multiple computers, sending commands and receiving results as needed. That is theory this quarter. No lab work is involved, but please review the concepts for quizzes and the final.

## Task 1: Explore a sample script file (10 points)

A sample script is provided in the lab files for this week. Let’s run the script and explore the pieces that make it a “script” instead of a simple pipeline.

Steps to complete the task:

1. Users of SPSCC lab computers can skip this step.  
 Home users only: if you haven’t already enabled execution of script files:

1.1. Run PowerShell as an administrator.

1.2. Run Set-ExecutionPolicy RemoteSigned to enable running of local script files

2. Run PowerShell ISE.

3. Open the Count-To.ps1 file in PowerShell ISE.

4. Review the code in the script, especially:

* Comments blocks in the <# #> section
* The parameter list
* The pipeline that is the core of the command

5. In the command pane, are you in the directory where Count-To is saved? If not, use CD to go to that directory.

6. In the command pane, run .\Count-To.ps1 and observe the output. The script should display a count from 0 to 10. If the script doesn’t run, use the red error text to debug the problem.

7. In the command pane, type Count-To . Note how PowerShell does *not* run scripts in the current directory by default. This is an intentional security feature.

8. Run Get-Help .\Count-To to view the help for this command

9. In help, view the examples for this command.

10. Use .\Count-to to generate a count from -7 to 7.

11. Paste a screenshot showing output and the text of your last PowerShell command(s) below the rubric.

12. Fill in the table below the rubric with information from the command source code.

Rubric:

* Screenshot of the output created: 1 points
* PowerShell command copy-and-pasted: 1 points
* Correct program operation: 3 points
* PowerShell command documentation template: 5 points total, about 1 point per section

Please paste a screenshot of a successful program run **and** copy-and-paste the source code of your PowerShell command(s) here:

Fill in this table with information from the source code:

|  |  |
| --- | --- |
| PowerShell command documentation template | |
| Name | Count-To.ps1 |
| Synopsis | Count-To counts from . . . |
| Description | Count-To returns PowerShell objects . . . |
| Parameters | CountFrom  CountTo |
| Examples | <copy and paste from source code> |

## Task 2: Create a script file (10 points)

Build on the previous example to create a new command, Count-ToIP, which creates a list of network addresses in a given range.

The input is a addressStart and a addressEnd.

* Both values should be in the range 0-255.
* Don’t worry about validation yet.
* If no values are provided, the default is 1 for addressStart and 254 for addressEnd.

Sample input and output:

|  |  |
| --- | --- |
| Sample command | Desired output |
| .\CountTo-IP | 192.168.0.1  192.168.0.2  192.168.0.3  192.168.0.4  . . . lots of numbers . . .  192.168.0.251  192.168.0.252  192.168.0.253  192.168.0.254 |
| .\CountTo-IP -addressStart 240 | 192.168.0.240  192.168.0.241  . . . lots of numbers . . .  192.168.0.253  192.168.0.254 |
| .\CountTo-IP -addressStart 240 -addressEnd 245 | 192.168.0.240  192.168.0.241  192.168.0.242  192.168.0.243  192.168.0.244  192.168.0.245 |

Steps to complete the task:

1. Complete the design table to describe Count-ToIP. The table is included below the rubric.

2. Modify the file from task 1 to create Count-ToIP.

3. Run the provided test cases, as well as any other test cases you care to try.

4. Paste a screenshot of your results and the text of your PowerShell .ps1 script file below the rubric.

Rubric:

* Screenshot of result file: 1 points
* PowerShell command copy-and-pasted: 1 points
* Correct program operation: 3 points
* PowerShell command documentation template: 5 points total, 1 point per section

|  |  |
| --- | --- |
| PowerShell command documentation template | |
| Name |  |
| Synopsis |  |
| Description |  |
| Parameters |  |
| Examples |  |

Please paste a screenshot of a successful program run and copy-and-paste the text of your PowerShell .ps1 script file here:

## Task 3: Work with regular expressions in PowerShell (10 points)

List all the files in the Windows directory that have a one- or two-digit number.

Advanced students: you do not need to create a recursive list.

This can be done in one line using pipelines.

Steps to complete the task:

1. List all the files in the Windows directory using dir c:\windows .

2. Filter the files using Where-Object and -match "\d{2}". This gives you a list of files with two-digit numbers somewhere in the filename.

3. Adjust the filter to also accept files with a one-digit number. You must accept both 1-digit and 2-digit numbers with *one* -match expression.

4. Turn this into a documented script.

4.1. Fill in the command documentation template below the rubric.

4.2. Save the pipeline from step 3 in a .ps1 file. Give it a name using PowerShell’s Verb-Noun convention. Cranky students: please use family-friendly language.

4.3. Add a <# #> comment block to document the new command.

4.3.1. The command has no parameters.

5. When this works as expected, view the output of the command and the results of get-help for the command.

6. Please paste a copy of your .ps1 source code, and a screenshot showing correct operation, below the rubric.

Rubric:

* Screenshot of PowerShell with command output visible: 1 points
* PowerShell code copy-and-pasted: 1 points
* Correct program operation: 2 points
* One match expression handles both cases: 1 points
* Documentation template: 3 points
* PowerShell command documentation template: 3 points total, about 0.75 points per section

Please paste a screenshot of a successful program run **and** copy-and-paste the source code of your PowerShell commands here:

|  |  |
| --- | --- |
| PowerShell command documentation template | |
| Name |  |
| Synopsis |  |
| Description |  |
| Parameters |  |
| Examples |  |

## Task 4. Advanced work, review (10 points)

Let’s explore PowerShell objects.

Steps to complete the task:

1. At the command prompt, use Powershell’s .. operator to create a list of numbers in the range 1-50.

2. What methods are available for these objects? Use gm to find out. List the methods in the space provided below the rubric.

3. Use a pipeline to check if each number is equal to 25. You should get many false results and one true result.

4. Use a pipeline to pass on only the object that has the value 25. You should get just one 25 result back.

5. Paste a screenshot of your results and the text of your PowerShell command(s) below the rubric.

Rubric:

* List of methods available for system.in32 objects: 4 points
* Screenshot of results: 1 points
* PowerShell command copy-and-pasted: 1 points
* Correct program operation: 4 points

List of methods available for the objects created by the .. operator:

Please paste a screenshot of a successful program run **and** copy-and-paste the source code of your PowerShell commands here: