PowerShell III.

Summary: In this lesson we will practice our PowerShell knowledge. We shall see how to start and stop processes. We shall get to know where and how can we use regular expressions. You can try out different types of error handling possibilities.

Commands: *start-process – to execute a process, stop-process (kill)*
Special parameter: *-erroraction (e.g. silentlycontinue)*
Structures: *try {} catch {} – error handling, trap (trap errors)*
.Net operator: *-match (using regular expression)*

Practices

a) There is a file with Neptun codes in it – one code per line. Create a PowerShell script which generates N length long passwords (mixed random characters and numbers) for them! Write out the result (the Neptun code, a space and the password per line) into a file! The filename and the length of the password N should have to be read! (get-random)
b) Write a PowerShell script which gets file extensions through the parameter list, creates this directories named by the elements of the parameter list and garbles the files from the actual directory into these newly created directories due to there extensions! E.g. the script gets jpg, txt and html as parameters, then creates directories jpg, txt and html and copy the files from the actual directory into them. The jpg files into jpg directory etc.
c) Write a PowerShell script which starts or stops the processes given by the parameter list! The first parameter should decide whether it has to start or stop the processes! (start-process, stop-process)
d) Implement a PowerShell script which decide those files which were modified within a given day (parameter)!
e) Create a PowerShell script which lists out the files which size is smaller then a given data (parameter) and which was modified within a given day (another parameter)!
f) Create a PowerShell script which counts the size of actual directory recursivly in full depth!
g) Write a PowerShell script which devide two numbers with each other. The numbers are given as parameters. Check whether the user would have given two parameters or not! Use some error handling to avoid division by zero! (try catch structure)
h) Implement a PowerShell script which gets a file with filenames in it and deletes them all. (One filename per line!) Be careful with missing files! (-erroraction silentlycontinue)
i) Create a PowerShell script which gets Neptun codes in a file and searches for that ones which contains a given substring – and copy them to a new file. The filenames and the substring are given by parameters.
j) Modify the script and check wheter the lines contain Neptun codes or not! If not throw an exception! (-match operator)
k) There is a file with Neptun codes in it. Write a script which generate new directories for each Neptun code and copy a readme file into them! Check whether the file is exist or not, whether the lines have Neptun code format (7 mixed characters and numbers) and whether the required directories have being existed already or not!

Financed from the financial support ELTE won from the Higher Education Restructuring Fund of the Hungarian Government