RELAZIONE POWERSHELL 

**Rossi Alessandro**

**Cosa è la powershell:**

PowerShell, è una [shell](https://it.wikipedia.org/wiki/Shell_(informatica)) caratterizzata dall'[interfaccia a riga di comando](https://it.wikipedia.org/wiki/Interfaccia_a_riga_di_comando) (CLI) e da un [linguaggio di scripting](https://it.wikipedia.org/wiki/Linguaggio_di_scripting), sviluppata da [Microsoft](https://it.wikipedia.org/wiki/Microsoft).

Si basa sulla piattaforma standard .NET framework dei sistemi Microsoft e sui principi della programmazione ad oggetti e utilizza parole chiave del linguaggio C#; risulta particolaremente adatta all'amministratore del sistema nella gestione dei processi e delle risorse.

**Spiegare gli elementi della PowerShell usati per risolvere gli esercizi:**

L’esercizio 18 pagina 161 chiedeva di:

Salvare sul file di nome Escape le informazioni ottenute dalla Guida in linea riguardante l'uso delle sequenze di escape.

Per risolvere questo esercizio ho scritto 2 righe di codice:

new-item "Escape.txt" -itemtype file

get-help about\_escape\_character | tee "Escape.txt"

Nella prima riga di codice attraverso il cmd new-item si può creare un nuovo file, in questo caso chiamato Escape.txt.

Nella seconda riga di codice possiamo trovare un | (pipe) questa ridirige l’output del primo comando all’imput del secondo.

Mentre Il comando tee realizza delle giunioni a T, il comando cattura, copiandolo su file, il flusso dei dati della pipeline senza compromettere il flusso stesso.

In questo caso le informazioni ottenute dalla Guida in linea riguardante l'uso delle sequenze escape ricavate dal primo comando vengono salvate sul file "Escape.txt" prima creato.

L’esercizio 19 pagina 161 chiedeva di:

Inviare alla stampante predefinita la lista dei nomi dei file che hanno come ultimi due caratteri le lettere at.

Per risolvere questo esercizio ho scritto 1 riga di codice:

get-childitem \*at.\*| out-printer

Con il cmd get-childitem si ottiene i file e le cartelle di un'unità di file system, in questo caso tutti i file che finivano con at. Poi l’output del primo comando veniva mandato in input alla stampante, e doveva stampare tutti i nomi di questi file.

Non siamo riusciti a testare questo script perché la stampante era fuori uso.

L’esercizio 24 pagina 161 chiedeva di:

Ridirigere su un file l'elenco formattato dei comandi ottenuti con il cmd-let Get-Command inviando in output solo la seconda e la terza colonna.

Per risolvere questo esercizio ho scritto 1 riga di codice:

get-command | format-table -property A, B | out-file z:\\Esercizi-Rossi-Alessandro\Esercizio24.txt

In questo esercizio abbiamo utilizzato due comandi importanti, **format**-table e **out**-file; il primo gestisce l'output, del comando precedente (get-command), verso una tabella con seconda e terza colonna solamente, l'ultimo salva la tabella su un nuovo file (*Esercizio24.txt*); nel comando è immesso anche il percorso su cui verrà creato il nuovo file(Esercizi-Rossi-Alessandro).

L’esercizio 27 pagina 161 chiedeva di:

Costruire lo script per controllare il valore fornito come parametro: se è maggiore di 10, scrivere "maggiore di 10", altrimenti scrivere "minore o uguale a 10". All'inizio dello script controllare anche che l'utente fornisca un parametro al momento dell'esecuzione dello script.

Per risolvere questo esercizio ho scritto 4 righe di codice:

if ($args -gt 10 && $args != null)

echo > di 10

if($args -le 10)

echo < o = a 10

La prima riga contiene i due cmd-let che controllano se il primo parametro inserito dall'utente è maggiore di 10, il cmd-let != null indica la presenza del parametro.

I parametri sono salvati di base sulla variabile $args già predefinita, il carattere "$" indica alla shell di sostituire il nome della variabile con il suo valore, in questo caso viene indicato solo il primo carattere ricevuto.

Se il parametro è maggiore di 10 stampa a video "maggiore di 10", echo è un alias del cmd-let write-host, altrimenti se minore o uguale a 10 stampa a video "minore o uguale a 10".

L’esercizio 33 pagina 161 chiedeva di:

Costruire uno script che consente all’utente di fornire una lettera e un numero: le lettere possono essere una delle quattro seguenti, e si deve fornire come output il risultato del calcolo richiesto: S = successivo, P = precedente, D = doppio, M = metà, del numero fornito dall’utente.

Per risolvere questo esercizio ho scritto 12 righe di codice:

$Numero = $args[1]

switch($args[0])

{

"S"{$Numero ++ write-host "$Numero"}

"P"{$Numero -- write-host "$Numero"}

"D"{$Numero = $Numero \* 2 write-host "$Numero"}

"M"{$Numero = $Numero /2 write-host "$Numero"}

default

{Write-host "S = successivo"

Write-host "P = precedente"

Write-host "D = Doppio"

Write-host "M = Metà"}

}

Nella prima stringa viene assegnato alla variabile numero il secondo carattere preso da tastiera che è salvato nella variabile predefinita dal sistema (args) e invece allo switch gli viene assegnato il primo carattere digitato nella tastiera.

Poi a seconda della lettera che clicchiamo il numero che inseriamo verrà o raddoppiato, dimezzato, sommato a 1 oppure sottratto a 1.

Osservazioni e Conclusioni:

Questo argomento non mi è piaciuto perché sembrava molto simile alla bash di linux, mi è sembrato di rifare le stesse cose.

Anche se è sempre piacevole vedere come funzionano altre shell.

In conclusione sono riuscito a svolgere quasi tutti gli esercizi, per questo so di saper fare script anche in powershell.

Spero che i prossimi argomenti saranno meno ripetitivi.