

# **CYB631 Automating Information Security with Python and Shell Scripting**

## **Lab 4: Confidentiality**

Your Name: **VAIBHAV A MAYEKAR**

### **Exercises:**

#### **[Environment: Starting with PowerShell ISE]**

1. Windows 10 environment is needed for the lab. Launch Windows PowerShell ISE. Click Start->Run, and then type PowerShell ISE.
2. Now, open a new PowerShell ISE and run as an administrator. Make sure that you navigate to the directory where your script is. Change the executive policy to RemoteSigned.

```
cd C:\Users\cybusr\cyb631  
Set-ExecutionPolicy remotesigned
```

#### **[Exercise I: Hashing]**

3. PowerShell has a Get-FileHash cmdlet for hashing files. The default Hash function is SHA256 if it is not specified.
4. Create some files for testing file hash. Let us learn about how to do this with a loop.

```
for ($i=1; $i -le 3; $i++)  
{  
    $fname = "HashTest"+[string]$i+".txt"  
    $msg= "This is a test file"+[string]$i+" file!!!"  
    New-Item -Name $fname -ItemType "file" -Value $msg  
}
```

5. We can then generate hash for all the files that you have just created. Here, we specify MD5 as the hashing algorithm.

```
Get-FileHash -Path .\HashTest*.txt -Algorithm MD5 | Format-List
```

6. Paste screenshot of your result above.

```

PS C:\Users\mayek\OneDrive\Desktop\python shell scripting> Get-FileHash -Path .\HashTest*.txt -Algorithm MD5 | Format-List

Algorithm : MD5
Hash      : BEC541A8FC89118EBD5E8236D2DC6AE2
Path      : C:\Users\mayek\OneDrive\Desktop\python shell scripting\HashTest1.txt

Algorithm : MD5
Hash      : 0BA8F3541F41A6D31CE5C1F55A69A6C9
Path      : C:\Users\mayek\OneDrive\Desktop\python shell scripting\HashTest2.txt

Algorithm : MD5
Hash      : 5B56F8EF355C1A695E478CDBE8C646F3
Path      : C:\Users\mayek\OneDrive\Desktop\python shell scripting\HashTest3.txt

```

- Alternatively, you can search for all the files under a directory and pipe them to the function.

**Get-ChildItem -Path \*.\*.txt | Get-FileHash**

- Next, please develop some scripts yourself to format the results from Get-FileHash like the one below. (**Hints:** try Foreach loop and Write-Host). The first column lists the file name and the second column lists the corresponding hash value of the file.

HashTest1.txt HashValue

HashText2.txt HashValue

.....

- Paste a screenshot of your scripts.

```

$files = Get-ChildItem -Path "." -Recurse -Filter "*.txt"
foreach ($file in $files) {
    $hash = Get-FileHash -Path $file -Algorithm MD5
    Write-Host "$file $hash.Hash"
}

```

- Paste a screenshot of your results running the scripts.

```

HashTest1.txt @{Algorithm=MD5; Hash=BEC541A8FC89118EBD5E8236D2DC6AE2; Path=C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4\HashTest1.txt}.Hash
HashTest2.txt @{Algorithm=MD5; Hash=0BA8F3541F41A6D31CE5C1F55A69A6C9; Path=C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4\HashTest2.txt}.Hash
HashTest3.txt @{Algorithm=MD5; Hash=5B56F8EF355C1A695E478CDBE8C646F3; Path=C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4\HashTest3.txt}.Hash

```

## [Exercise II: Integrity Check for Downloading Files]

10. Further, we can use Get-FileHash to perform an integrity check for downloading a file (software) from the Internet.

11. Create a Web Client.

```
$wc = [System.Net.WebClient]::new()
```

12. We will use a download link from Microsoft and verify it with the pre-calculated file hash of the software.

```
$pkgurl =
```

```
'https://github.com/PowerShell/PowerShell/releases/download/v6.2.4/powershell_6.2.4-1.debian.9_amd64.deb'
```

```
$publishedHash =
```

```
'8E28E54D601F0751922DE24632C1E716B4684876255CF82304A9B19E89A9CCAC'
```

13. Get the file hash from the file being downloaded via the web client link.

```
$FileHash = Get-FileHash -InputStream ($wc.OpenRead($pkgurl))
```

14. Compare the Hash.

```
$result=$FileHash.Hash -eq $publishedHash
```

```
Write-Host "The result of file integrity check is ", $result
```

15. Paste a screenshot of your results above.

```

PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4> $wc = [System.Net.WebClient]::new()
PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4> $pkgurl = 'https://github.com/PowerShell/PowerShell/releases/download/v6.2.4/powershell_6.2.4-1.debian.9_amd64.deb'
PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4> $publishedHash = '8E28E54D601F0751922DE24632C1E716B4684876255CF82304A9B19E89A9CCAC'

PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4> $fileHash = Get-FileHash -InputStream ($wc.OpenRead($pkgurl))

PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4> $result=$fileHash.Hash -eq $publishedHash

PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4> Write-Host "The result of file integrity check is ", $result
The result of file integrity check is True
PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4>

```

16. What does the results mean? Please explain it. **Verify that a file has not been corrupted or tampered with by comparing its hash value to a known good hash value.**

### [Exercise III: SecureString and AES Encryption]

17. PowerShell has a data type called Secure String. It stores data in the memory that cannot be viewed by users but the content is not encrypted in the memory. Read a string from command line and store it as a SecureString.

**\$secure=read-host -AsSecureString**

**\$secure**

18. Convert the SecureString into an encrypted text. The default encryption used in PowerShell is AES with a key in 16 Bytes, if not specified.

**\$encrypted = ConvertFrom-SecureString -SecureString \$secure**

**\$encrypted**

19. You can always decrypt it on the same machine. On a different machine, you will need to specify a file that stores the encryption key since it is encrypted by AES previously.

**\$decrypted = \$encrypted | ConvertTo-SecureString**

**\$decrypted**

20. To convert the Secure String to plain text, you will need to copy the SecureString in the memory to an unmanaged binary string (BSTR). Then you convert the BSTR variable to a managed string.

**\$bstr=[System.Runtime.InteropServices.Marshal]::SecureStringToBSTR(\$secure)**

**\$unsecure=[System.Runtime.InteropServices.Marshal]::ptrtostringauto(\$bstr)**

**\$unsecure**

21. What is the content of the three variables above?

**\$secure: System.Security.SecureString**

\$encrypted: **\$encrypted**

\$unsecure: **HELLOGUYS**

How are they different? **SecureString is stored in memory and not encrypted, while encrypted string is encrypted using AES and can be stored on disk or transmitted over a network.**

Does SecureString data type contain encrypted information? **NO**

22. To specify an encryption key, you can either assign a key with a fixed value or generate a random key (a better method).

```
[Byte[]]$key = (1..16)
```

```
$encryptedwithkey = $secure | ConvertFrom-SecureString -key $key
```

```
$encryptedwithkey
```

23. Alternatively, you can generate a random 16-byte (128 bits) key. Write the key to a key file which can be stored in the machine where decryption is needed.

```
$rkey= New-Object Byte[] 16
```

```
[Security.Cryptography.RNGCryptoServiceProvider]::Create().GetBytes($rkey)
```

```
$rkey | Out-File ".\keyfile.txt"
```

24. Write your own script to read the key from the key file, encrypt the **\$secure** variable from the previous step and then decryption it using the same key. (Hints: Use **Get-Content** to read from a file.)

25. Paste a screenshot of your scripts from above.

```
$key = Get-Content -Path ".\keyfile.txt"
$encrypted = ConvertFrom-SecureString -SecureString $secure -key $key
$decrypted = $encrypted | ConvertTo-SecureString -key $key
Write-Host "Decrypted string: $decrypted"
```

26. To demonstrate your script, paste a screenshot of your results from above.

```
PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4> $key = Get-Content -Path ".\keyfile.txt"
PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4> $encrypted = ConvertFrom-SecureString -SecureString $secure -Key $key
PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4> $decrypted = $encrypted | ConvertTo-SecureString -Key $key
PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4> Write-Host "Decrypted string: $decrypted"
Decrypted string: System.Security.SecureString
```

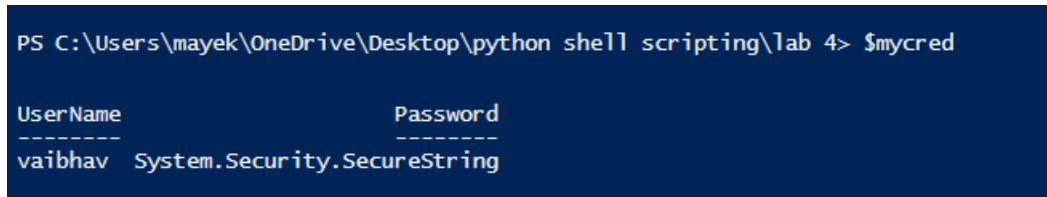
## [Exercise IV: Secure Passwords]

27. You can obtain credential information from users using Get-Credential cmdlet, which returns an object called "PSCredential".

```
$mycred=Get-Credential
```

```
$mycred
```

28. Paste a screenshot of your results above.



```
PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4> $mycred

UserName                Password
-----                -
vaibhav  System.Security.SecureString
```

29. You can obtain the key that you generated from the previous step in the key file to encrypt the password in the credential. The encrypted password can then be stored in a file.

```
$kfile=".\keyfile.txt"
```

```
$mycred.Password | ConvertFrom-SecureString -Key (Get-Content $kfile) | Out-File ".\password.txt"
```

30. What would happen if the key file in the above step is missing?

**If you don't have the key file mentioned earlier, you won't be able to unlock the encrypted password stored in the .\password.txt file. The key file is necessary for decrypting the password.**

31. Is the key file encrypted at this point? **No the key file is not encrypted at this point**

**What would happen if an adversary obtained the key file? If someone gets hold of the key file, they can unlock the password in the .\password.txt file and potentially access the account it belongs to. To keep the key file safe, make sure to store it in a secure place and use strong encryption to protect it.**

## [Exercise V: Self-Signed Certificate and Script Signing]

32. PowerShell requires scripts to be signed as a security precaution so that others cannot run malicious scripts on a protected host. Create a self-signed certificate for the computer.

```
$selfcert = New-SelfSignedCertificate -Subject "CYB631 Authentication"
-CertStoreLocation Cert:\LocalMachine\My -Type CodeSigningCert
```

33. Store the self-signed certificate, **selfcert**, in the machine's Root Store and Trusted Publisher's Store.

```

$rootstore =
[System.Security.Cryptography.X509Certificates.X509Store]::new("Root","LocalMachine")

$rootstore.Open("ReadWrite")

$rootstore.Add($selfcert)

$rootstore.Close()

```

```

$publisherstore =
[System.Security.Cryptography.X509Certificates.X509Store]::new("TrustedPublisher","LocalMachine")

$publisherstore.Open("ReadWrite")

$publisherstore.Add($selfcert)

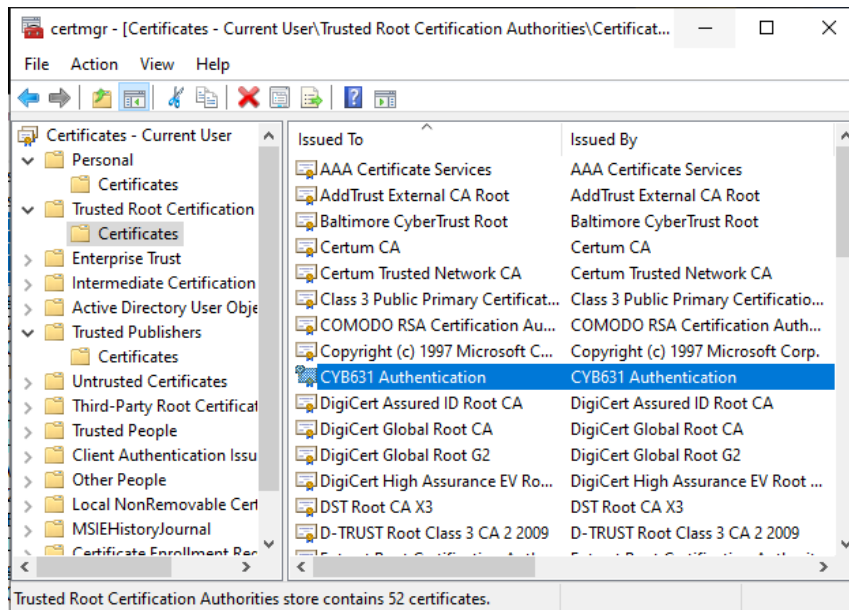
$publisherstore.Close()

```

34. You can view the certificate using Certificate Management Console (certmgr.msc). Under Windows command search (lower-left corner), type,

**certmgr.msc**

35. You should be able to see the certificate, like the Window below.



36. Double-Click on “CYB631 Authentication” to review the content of the certificate.

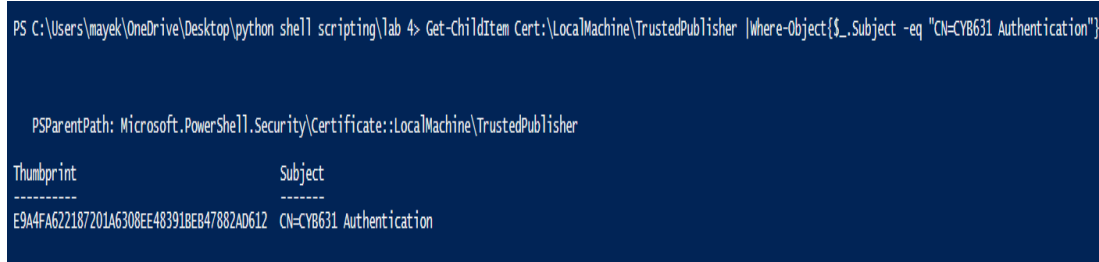
37. You can also use PowerSehll to review the certificate in your own path, in the root store and in the trusted publisher’s store. Make sure that you have done the previous step correctly.

```
Get-ChildItem Cert:\LocalMachine\My |Where-Object{$_.Subject -eq  
"CN=CYB631 Authentication"}
```

```
Get-ChildItem Cert:\LocalMachine\Root |Where-Object{$_.Subject -eq  
"CN=CYB631 Authentication"}
```

```
Get-ChildItem Cert:\LocalMachine\TrustedPublisher |Where-Object{$_.Subject -eq  
"CN=CYB631 Authentication"}
```

38. Paste the screenshot that contains the **Thumbprint** and **Subject** of the certificate that you created.



```
PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4> Get-ChildItem Cert:\LocalMachine\TrustedPublisher |Where-Object{$_.Subject -eq "CN=CYB631 Authentication"}

PSParentPath: Microsoft.PowerShell.Security\Certificate::LocalMachine\TrustedPublisher

Thumbprint                               Subject
-----
E9A4FA622187201A6308EE483918EB47882AD612  CN=CYB631 Authentication
```

39. Create a simple PowerShell script for testing certificate signing. We will sign one of them using the self-signed certificate. For example, creating a file called “**myscript.ps1**” containing the following codes:

```
Write-Host "I am on cloud nine!"
```

40. By default, any new PowerShell scripts have to be signed unless the Execution Policy is changed for that script. You should encounter an error when running myscript.ps1.

```
.\myscript.ps1
```

41. We will sign myscript.ps1 with the certificate that we just created. You will need to retrieve it from your own path and then use it to sign. We are using DigiCert’s time server for stamping the certificate. Since DigiCert’s time server is on the Internet, you will need Internet access for this script to work.

```
$codecertificate= Get-ChildItem Cert:\LocalMachine\My |Where-  
Object{$_.Subject -eq "CN=CYB631 Authentication"}
```

```
$codecertificate
```

```
Set-AuthenticodeSignature -FilePath .\myscript.ps1 -Certificate $codecertificate  
-TimestampServer http://timestamp.digicert.com
```

42. Now, myscript.ps1 has a digital signature signed by the certificate. To view it, you can open myscript.ps1 and you will see the signature block at the end of the file.

```
notepad .\myscript.ps1
```

43. Paste a screenshot of the script including its original scripts and the first five lines of the signature block.



```
Write-Host "I am on cloud nine!"

# SIG # Begin signature block
# MIIbrAYJKoZIhvcNAQcCoIIbnTCCG5kCAQExCzAJBgUrDgMCGGUAMGkGCisGAQQB
# gjcCAQSGWzBZMDQGCisGAQQBgjccAR4wJgIDAQAABBAfzDtgWUsITrck0sYpfvNR
# AgEAAgEAAgEAAgEAAgEAMCEwCQYFKw4DAhoFAAQUqp8KfwLW4i740Gtbx65MnNm3
# TDugghYdMIIDEDCCAFigAwIBAgIQUh8yDJMaQq5PCMACl93bXTANBgkqhkiG9w0B
# AQsFADAgMR4wHAYDVQQDDDBVDWUI2MzEgQXV0aGVudGljYXRpb24wHhcNMjMxMDAy
# MjEwNDI1WhcNMjMxMDAyMjEwNDI1WjAgMR4wHAYDVQQDDDBVDWUI2MzEgQXV0aGVu
# dGljYXRpb24wHhcNMjMxMDAyMjEwNDI1WjAgMR4wHAYDVQQDDDBVDWUI2MzEgQXV0aGVu
```

44. Validate the signature of the file.

**Get-AuthenticodeSignature -FilePath .\myscript.ps1 |Select-Object -Property \***

45. Run myscript.again. It should work this time since it is now signed.

**.\myscript.ps1**

46. Are you able to run myscript.ps1? **Yes**

Paste a screenshot of your result from running myscript.ps1.

```
E9A4FA622167201A0506EE46591BEB47682AD01Z
TimeStamperCertificate : [Subject]
                        CN=DigiCert Timestamp 2023, O="DigiCert, Inc.", C=US
                        [Issuer]
                        CN=DigiCert Trusted G4 RSA4096 SHA256 TimeStamping CA, O="DigiCert, Inc.", C=US
                        [Serial Number]
                        0544AFF3949D0839A6BFD83F5FE56116
                        [Not Before]
                        7/13/2023 8:00:00 PM
                        [Not After]
                        10/13/2034 7:59:59 PM
                        [Thumbprint]
                        66F02B32C2C2C90F825DCEAA8AC9C64F199CCF40
Status                  : Valid
StatusMessage           : Signature verified.
Path                    : C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4\myscript.ps1
SignatureType           : Authenticode
IsOSBinary              : False

PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4> .\myscript.ps1
I am on cloud nine!
```

## [Exercise VI: Encryption and Decryption using a Certificate]

47. Digital certificates facilitate public key encryption. **Protect-CmdMessage** uses public key encryption. The public key is contained in the certificate.

**New-SelfSignedCertificate -DnsName cyberusr -CertStoreLocation "cert:\currentuser\my" -KeyUsage KeyEncipherment, DataEncipherment, KeyAgreement -Type DocumentEncryptionCert**

48. You can view the certificate by the following:

**Get-ChildItem -Path Cert:\CurrentUser\My -DocumentEncryptionCert**

49. Encrypt a message and store it in a file p1.txt. The message is encrypted using the public key stored in the certificate.

**"This is a secret!" | Protect-CmsMessage -To CN=cyberusr -OutFile p1.txt**

50. Exam p1.txt. Paste a screenshot of the file to show the first 5 lines.

```
-----BEGIN CMS-----
MIIBqwYJKoZIhvcNAQcDoIIBnDCCAZgCAQAxggFDMIIBPwIBADANMBMxETAPBgNVBAMMCGN5YmVY
dXNyAha03dMB4wp0h09nKQxhbkqMA0GCSqGSIb3DQEBBzAABIIABASm8aZTI08sOi99VqKBGdOh
34ZlZHC4WuJMSYAs71wbvy2+L0bUFS1xyRi6fuCJxv99o3SAjViIcLkzTs4+oJl8SudHOS6TH8B3
QhCEh7kJBhuy3VYD++087FQKymMSU6s15yg5/D8UV50eNSnpI1UkIsvscnaxzSCBcbloF8Hgl0ib
KwJQkrJuZ7KfUotl8MckPdcM51eLXEW5jsy7tBdl+O/dF2G/Vfqh2V9ry97Hsiw9lMtLDlnjmC7z
9QtI9In0eeMRiLghod3BGCBrrB8vBi2NV8QGzN3shR4DdEFH7aD0GEBYtMkp2Y8ncRZP2p9NZUwI2
Q+XbC4zoFYvnC7YwTAYJKoZIhvcNAQcBMB0GCWCGSFA1AwQBKqQe7+6zi644EzzNFau6TQyWYAg
ICSIVD0hTnrRv2jBuviv69xz34jWC0KjetoagQMv5MY=
-----END CMS-----
```

51. To decrypt the file, you will need to log in with the user credential that created the certificate and has the private key (the password for the account).

**Unprotect-CmsMessage -Path p1.txt**

52. For others (other users or other hosts) to send you an encrypted file, you will need to export your Certificate for others to import. There is a set of PKI cmdlets that can be used to manage digital certificates. Search Microsoft Learn site (<https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/powershell>) to find out how to export a certificate.

53. Paste a screenshot of the PowerShell scripts to export the certificate you created in this exercise.

```
Export-Certificate -Cert (Get-ChildItem -Path Cert:\LocalMachine\My | where-
Object {$_.Thumbprint -eq "E9A4FA622187201A6308EE48391BEB47882AD612"}) -
FilePath "C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab
4\newvam_cert.cer" -Type CERT
```

54. Paste a snapshot of the content in the certificate file exported above to show that you completed it.

```
PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4> Export-Certificate -Cert (Get-ChildItem -Path Cert:\LocalMachine\My | where-Object {$_.Thumbprint -eq "E9A4FA622187201A6308EE48391BEB47882AD612"})

Directory: C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4

Mode                LastWriteTime         Length Name
----                -
-a-----         10/5/2023   4:33 PM           788 newvam_cert.cer
```

```

PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4> Export-Certificate -Cert (Get-ChildItem -Path Cert:\LocalMachine\My | Where-Object {$_.Thumbprint -eq "E9A4FA622187201A6308EE48391BEB47882AD612"})

Directory: C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4

Mode                LastWriteTime         Length Name
----                -
-a-----         10/5/2023   4:33 PM             788 newnam_cert.cer

PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4> cat newnam_cert.cer
0,0,0 R2 "B*0A-Y0]0
+Hf+
0 10U CY8631 Authentication0
2310022104292
24100221242920 10U CY8631 Authentication0,"0
+Hf+
, 0,
, 'XA(18.pëu% 92xf11,*'ôizt0f10",of>cmw:pr|s' a6yZzI:-w'ôsv8]Y0y
UNDe-A-Uw'ITPEx20k0"0cë8ëiAA/
3Iâ/ihKR0* I0z80 77uL*([0^NAKTGâj<k^'j&:0%Z8E MÔôz!$*2-êf8e)'Zn'y,ib8.f0wÄ'"âê.'(A^zâV01,E' s""ipSjê.X6eEV517Ké')id,HA[="0-m^âK0ë+ fF000uyê0U% 0
+000y4e,;ic0^%8{Qa=XA,0
+Hf+
, :AI0"[çZ^iI7-J0W]if zê'!â 0Nÿw$000=d,*êkiAîââ f9^#*¶E80A0,+{^|\^~E <0G^v00'l891A'ÿêâ¶20Fz2zWNP^â
];vêf^50^w^mââ:ô^+^
^â:ê^ê:â0-00 -1e'p000% ...0I^w-wµzêâloiyk^H 0t/k~f^H';a1"|0âµ,Ê ]T%û:0N]_,cê-ý.{5^"K^oçkLDG
ENAS7$â8
PS C:\Users\mayek\OneDrive\Desktop\python shell scripting\lab 4>

```