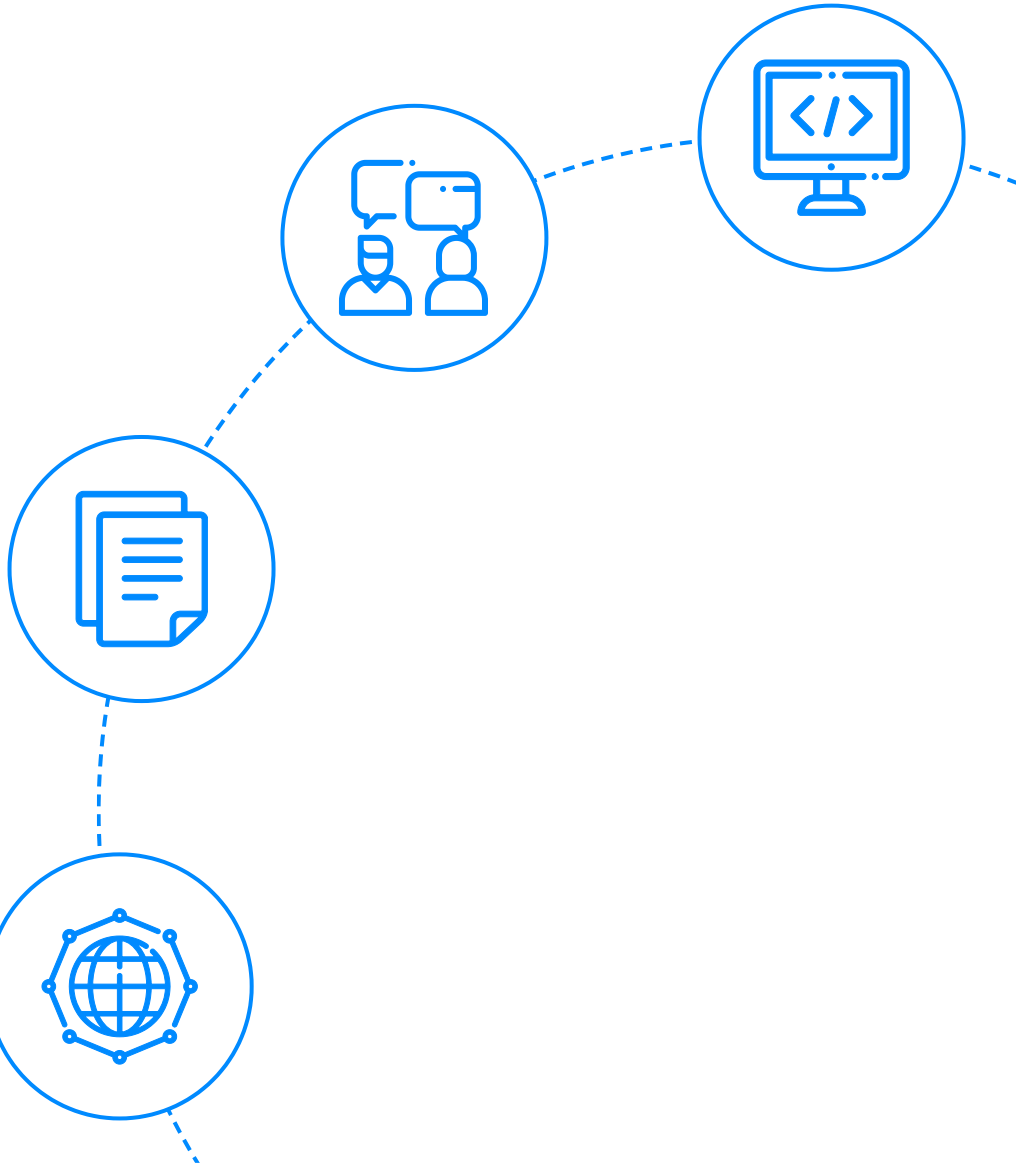
InterviewBit

PowerShell Interview Questions



To view the live version of the page, click here.

Contents

PowerShell Questions for Freshers

PowerShell Questions for Experienced

PowerShell Questions for Experienced

(.....Continued)

19. Explain the importance of PowerShell brackets?
20. Describe the various types of execution policies in PowerShell?
21. Describe what is Powershell Pipeline used for?
22. Why is scripting debugging important?
23. Do you make PowerShell scripts to deploy components in SharePoint?
24. Describe what is Powershell Get-command?
25. What is your take on Variable Interpolation?
26. What is the benefit of the hashtable in PowerShell?
27. Describe what is the benefit of Array in PowerShell?
28. Tell about PowerShell's Get-ServiceStatus function?
29. What is pipeline in PowerShell?
30. Explain PowerShell's comparison operators?

Let's get Started

Most of us are familiar with the term PowerShell, it is a configuration management framework and task automation software developed by the renowned software company, **Microsoft**. It comprises an associated scripting language and a command-line shell also.

Previously, PowerShell was regarded as a Windows component but later came to be known as Windows PowerShell, and it was made open-source. Post which, it also brought PowerShell Core, and currently, PowerShell is a cross-platform framework. It was designed on the .NET Framework and the most recent version is created on .NET Core. PowerShell is equipped with elements including full access to WMI (Windows Management Instrumentation), which allows administrators to accomplish many administrative tasks on both remote and local Windows systems. It also offers a hosting API that can be operated by the PowerShell runtime to be implanted within other applications. These applications then utilize the ultimate benefits of PowerShell functionality to enforce specific operations which may have graphical interfaces etc.

Powershell is gaining more popularity these days. PowerShell Scripting is considered one of the benchmarks in the job description by every **DevOps professional** and System Admin. Now, if you are seeking a job that is related to PowerShell, then you ought to prepare for the PowerShell Interview Questions. Every interview is indeed different as per the varied job roles.

In this article, we have mentioned the necessary PowerShell Interview Questions and Answers which will allow you to ace your interview.

PowerShell Questions for Freshers

1. Briefly explain what is PowerShell

PowerShell is a cross-platform task automation solution composed of a scripting language, command-line shell, and a configuration management framework. PowerShell operates on Windows, Linux, and macOS. In 2006, when PowerShell was released, this effective tool effectively substituted Command Prompt as the default way to ensue automatic batch processes and develop personalized system management tools. Multiple system administrators, such as MSPs (managed services providers) depend on Powershell's 130 plus command-line tools to simplify and scale tasks in remote and systems. Its abilities have been integrated into several other interfaces. It has always been crucial for MSPs to comprehend how PowerShell functions, what it is employed for, and how to make management tasks automatic, which will optimise time and effort.

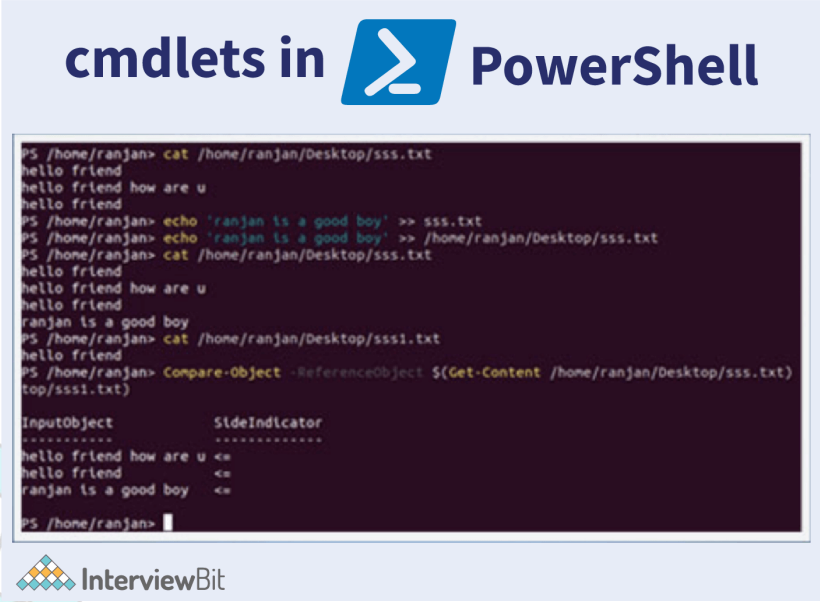
2. What does PowerShell do?

Microsoft created PowerShell to make system tasks automatic, like batch processing, to build system management mechanisms for generally implemented processes. The PowerShell language is quite similar to Perl and delivers different ways to automate tasks:

- One way is with cmdlets, which are miniature in nature .NET classes emerge as system commands;
- Another way is with scripts, these are an amalgamation of cmdlets and associated logic;
- It can be done with executables, these are stand-alone mechanisms
- Also with the embodiment of standard .NET classes.

Admins can employ PowerShell to manage a broad range of activities like it can pull information on Operating Systems, as the particular version and service pack levels. "PowerShell providers" are programs that enable data included in specialized data stocks available at the command line. Those data stores contain Windows file system drives. PowerShell also acts as the substitute for Microsoft's Command Prompt, which goes way back to DOS. For example, Microsoft turned PowerShell the default command-line interface (CLI) for Windows 10 as of build 14791. PowerShell's function as a command-line shell is how most consumers become familiarized with the technology.

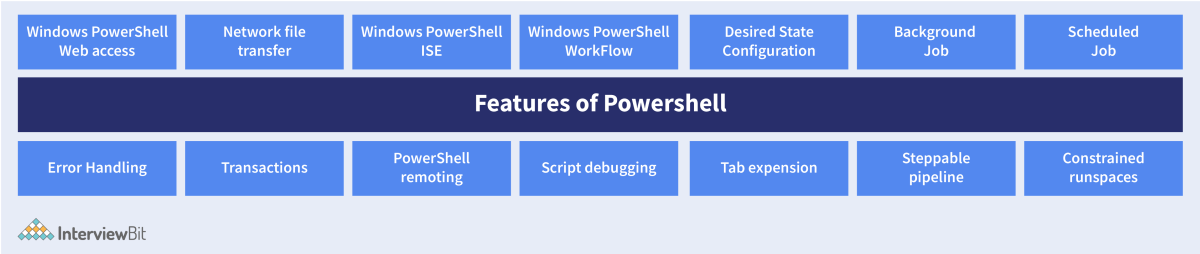
3. What are cmdlets?



A cmdlet is a lightweight command that is utilised in the PowerShell environment. The PowerShell runtime conjures these cmdlets in the context of automation scripts that are given at the command line. You can see the entire inventory of these commands by executing “Get-Command-Type Cmdlet” in PowerShell. Cmdlets can be USED individually, but they are more effective when conjoined—cmdlets can be used within scripts, then package scripts into more exhaustive modules. These are effective mechanisms largely as PowerShell is created on an underlying .NET framework, which lets PowerShell work more like a programming language rather than an easy command-line program. The program makes use of objects, which are a sort of manifestation of properties (attributes) or methods (instructions). With the help of PowerShell, you can utilize “pipes” that let you pass output of cmdlet to another cmdlet’s input as an object, letting numerous cmdlets operate jointly to configure the identical data. This is the basic change that has made PowerShell such a strong tool for Windows configuration.

4. What are the key PowerShell features?

Microsoft implements updates and the latest features separately for the PowerShell version, but let us look at the list of the primary features and characteristics of PowerShell:



- **Discoverability:** Consumers can uncover PowerShell's attributes with the help of cmdlets including Get-Command, which constructs a list of all the commands, like cmdlets and functions, accessible on a particular computer. Parameters can be utilized to limit the scope of the search.
- **Remote commands:** Admins can execute remote operations on one or numerous computers, taking the benefit of technologies like WS-Management and Windows Management Instrumentation. For example, the WS-Management protocol, allows the customer to execute PowerShell scripts and commands on remote computers.
- **Pipelining:** Owing to PowerShell, commands can be connected together via the pipe operator, represented as |. This approach permits the output from a particular command to turn into the input for the subsequent command in the pipeline sequence. The PowerShell pipeline allows objects, instead of flowing from one cmdlet to another and text strings.
- **Tab expansion:** Tab expansion is an execution of auto-completion, which executes the cmdlets, properties, and parameter names by pushing the Tab key once.

5. Define PowerShell Integrated Scripting Environment (ISE).

PowerShell ISE(Integrated Scripting Environment), released by Microsoft in PowerShell version 2.0, is a PowerShell host application utilized to document, test, and debug scripts or compose commands in a Windows graphical user interface (GUI). PowerShell ISE is packed with numerous components, including syntax colouring, multiline editing, context-sensitive assistance, and tab completion. PowerShell ISE contains refined characteristics that Windows users are familiar with. For example, a customer can emphasize and copy a part of a PowerShell command with the help of a mouse or can opt for the Shift + Arrow hotkey combination. Also, the user can paste the content at any place in the editor window.

Another beneficial attribute is the capacity to keep various versions of a command in the editor and execute commands you require in the PowerShell ISE. A command is directly launched by the F5 key from the editor. To run a certain line, choose it and press F8. The context-sensitive help exhibits matching cmdlets when the customer begins to enter a command. A command add-on displays a list of cmdlets to opt from.

PowerShell ISE enables tabs to work on numerous administrative tasks. PowerShell ISE facilitates speedy switching from the CLI to scripting mode.

6. What are PowerShell modules?

A PowerShell module is a conjoined portion of PowerShell functions or grouped code. All PowerShell cmdlets and providers are incorporated by a module or a snap-in. The receivers of these modules can add the commands included in the module to their PowerShell sessions to use them like built-in commands. The most uncomplicated method to build a PowerShell module is to save the script as a PSM1 file.

A PowerShell module includes 4 fundamental elements:

- A PSM file, being the module;
- Help files or scripts required by the module;
- A manifest file that defines the module
- A directory that accumulates the content.

There are four types of PowerShell modules:

- **Script module:** A PSM1 file that includes different functions to allow admins to execute management functions, import, and export.
- **Binary module:** A .NET framework assembly (DLL file) that includes saved code. Programmers generally utilize a binary module to build cmdlets with strong features not easily done with a PowerShell script.
- **Manifest module**: A module (PSM1) file with an associated PSD1 file (manifest).
- **Dynamic module:** A dynamic module is dynamically built on demand by a script. It isn't reserved or loaded to persistent storage.

7. Why should developers use PowerShell?

PowerShell is a famous instrument for many MSPs as its scalability allows to streamline management tasks and develop insights into instruments, specifically over medium or large networks.

Here’s how PowerShell's usefulness can change your workflow:

- **Automate time-consuming tasks**: With the help of Automate time-consuming tasks provided by PowerShell, you don't have to complete the identical task again and again or even take the time for manual configuration. For example, you can employ cmdlets like Get-Command to look for other cmdlets, Get-Help to find these cmdlets' syntax and benefits, and Invoke-Command to drive a common script remotely or locally even with batch control.
- **Offer network-wide workarounds:** Making use of PowerShell allows you to get around the limitations of software or program, particularly on a business-wide scale. For example, PowerShell can be employed to reconfigure the default settings of a program over an entire network. This could be helpful if a company wishes to roll out a certain protocol to all its customers—say, convincing users to employ two-factor authentication (2FA) or modify their password every other month.
- **Take your endeavours across various devices:** PowerShell serves as a lifesaver if your script requires to be run across countless systems, especially if some of them are remote devices.If you're attempting to incorporate a solution on quite a few devices or servers at once, you don't want to log in individually into devices. In minutes PowerShell can assist you to collect information about numerous devices, as compared to the endless time it would require to scan each device manually. Once you allow PowerShell remoting, you'll be able to enable your scripts to reach several machines at once, letting you install updates, settings configuration, compile information, and more importantly, cutting down hours of work and travel time.
- **The benefit of command-line interfaces:** The added advantage of command-line interfaces such as PowerShell is the access they give you to a system's file system. PowerShell constructs the Windows Registry, hard-to-find data in files, and digital signature certificates visible even though it is housed on many systems. This information can then be exported for the purpose of reporting.

Ultimately, as every Windows 10 computer should have pre-installed it, it's not challenging to understand PowerShell. As an MSP, comprehending PowerShell not only places you a step ahead of your peers but offers you a broad range of useful capabilities. If you are aware of scripting cmdlets for PowerShell, it's that much uncomplicated for you to heighten your efforts and deliver precise, adjustable, and quick service to customers.

8. Mention the two important differences between Bash and PowerShell?

This question is a great opportunity to get an understanding of the candidate's knowledge of PowerShell core concepts and his/her favourite PowerShell features.

For instance, one of the most essential distinctions between PowerShell and bash is that PowerShell is object-oriented whereas bash is text-oriented. PowerShell treats input and output as an object and Bash always considers input and output as a text structure.

PowerShell's user interface is a graphical command-line interface CLI. whereas Bash's user interface shell is a text-based command-line interface. PowerShell can run on any version of Windows ranging from Windows 97 to Windows 10. Bash is specifically designed for Linux and Unix operating systems from the first day.

9. Explain how would you place a registry value with PowerShell?

Registry values are properties of keys and, as such, cannot be directly browsed. This requires us to understand that there are no registry-specific cmdlets, so we have to utilize the registry provider and Set-ItemProperty. You can also utilize the New-ItemProperty cmdlet to build the registry entry and its value and then make use of Set-ItemProperty to modify the value.

The registry has been an essential part of Windows forever, thus this is hardly area-specific knowledge.

10. Distinguish between the notion of WMI between old and new ideas.

Old WMI

- Employs the old-style native code providers and a repository.
- Accessible only on Windows as mentioned.
- It has been more or less deprecated which means it's is not focused on further advancement or development but can be used..

New WMI

- Backs old-style native code providers and a repository, along with new-style MI providers.
- Accessible only on Windows as mentioned. It has a stateless relationship with the remote machine.
- The main attraction of the new WMI is that it employs WSMan and no more DCOM errors are possible with this.

11. Explain the difference between CIM (Common Information Model) vs WMI (Windows Management Instrumentation).

CIM	Old WMI	New WMI
CIM is a Vendor-neutral and industry-standard designed way of manifestation of management information.	Old WMI is Microsoft's initial incorporation of CIM.	New WMI was released alongside WMF v3 in the year 2012 and it was compliant to the latest CIM standards.
It is designed by another company named the DMTF	It is designed by Microsoft	It is designed by Microsoft as well.
It not available	It is available in PowerShell v1	It is available in PowerShell v3.
It employs WSMan which is a standard developed by DMTF.	It used DCOM and RPCs Remote Procedure Calls.	It employs WSMan and no more DCOM errors are possible with this.
It can work on any platform.	Works on Windows only.	Works on Windows only.
It owns WSMan Port – 5985 (HTTP) and 5986(HTTPS) for its purpose.	It has RPC port- 135 for use	It owns WSMan Port – 5985 (HTTP) and 5986(HTTPS) for use

12. Mention the types of Powershell Scopes.

The four types of PowerShell scopes are Global, Local, Script, Private.

- **Global**:
  - The scope that is in effect while PowerShell runs or when you start a new session or run space.
  - Variables and functions that are included when PowerShell begins have been designed in the global scope. This contains preference variables and automatic variables.
  - This also comprises functions, variables, aliases that are within your PowerShell profile.
- **Local**:
  - The current scope. The local scope can be the global scope or any other scope.
- **Script:**
  - The scope is built while a script file runs. Just the commands in the script operate in the script scope. To the commands in a script, the script scope is the local scope.
- **Private**:
  - You cannot see the items in private scope outside of the current scope. You can utilize the private scope to build a private version of an item having the exact name in another scope.

13. Is Windows PowerShell similar to Command Prompt?

Though Windows Powershell 1.0 was released as a substitute for Command Prompt, it's incorrect to assume PowerShell as just a recent version of the classic command-line interpreter. As a matter of fact, both programs still live on Windows 10, although PowerShell is much more effective. Some intermediate-level consumers may select to utilize Command Prompt if they are already acquainted with the language—its interface performs simple DOS commands, and for some consumers, that is enough. But the many benefits of PowerShell make it a more engaging tool for MSPs who desire true control over a network. By offering cmdlets that can go into registry management and WMI, PowerShell provides you access to additional system administration tasks than Command Prompt, particularly as PowerShell is not only meant for Windows but it is an open-source mechanism for Linux and Mac OS X as well. SolarWinds® Remote Monitoring & Management (RMM) delivers all the benefits of PowerShell without needing MSPs to even utilize PowerShell scripts. With the help of RMM, you can employ an easy drag-and-drop object to effortlessly devise a broad range of automated functions. This type of automation is required for busy MSPs eyeing effective, scalable business practices. RMM's user-friendly interface allows you to leverage all the advantages of Windows PowerShell, faster.

14. What are language constructs?

PowerShell offers a number of language constructs that allow you to manage the flow of your script and let you make decisions about what it should do. A few of the language constructs have conditionals, switches, loops, and variables.

- **Conditionals:** This language construct is utilized to assess a conditional expression. If the conditional expression is true, a script block is accomplished:

```
if ( $i -eq 1)
{
   ## Do something
}
Else
{
   ## Do something
}
```

- **Switch:** The switch statement lets you deliver a variable and a list of potential values. If the value matches the variable, then its script block is completed..

```
switch  ($i)  {
    0
    {
       Write-Host “I is 0”
    }
    1
    {
       Write-Host “I is 0”
     }
     Default
     {
       Write-Host “I is 0”
      }
  }
```

- **Loops:** What while statement does is repeat a block of code as long as the below mentioned conditional expression is working:

```
while ($i -eq 0)  {
         ## do something
}
```

The do loop is identical to the while loop. The only distinction is PowerShell runs the do loop at the end of the loop.

```
do  {
     ## do something
  }  while  ($i -lt 0)
```

When you employ a foreach loop, PowerShell repeats the code for every item cited in the script.

```
$array = ( ‘item1’ , ‘item2’ , ‘item3’)
  foreach ($item in $array)  {
}
```

Make use of a for loop to execute statements constantly till a condition is met.

```
for ($i = 0; $i -lt 5; $i++)
  {
     $i
   }
```

15. Explain briefly what are Help and comments in PowerShell.

PowerShell allows the accumulation of help topics for modules, scripts, and individual commands. To look at all the help topics, you can use the Get-Help command. While importing a module into a session, PowerShell transfers the help topics for that module automatically. In case of no help topics for a module, the Get-Help command demonstrates autogenerated help. There are three types of help content that exist in PowerShell: external help, comment-based help, and updatable help. Comment-based help indicates comments contained with a script or command for Get-Help to read. External help allows the author to describe help content in an external XML file documented in XAML. Updatable help utilizes external help but allows users to download the new help content with the Update-Help command.

PowerShell Questions for Experienced

16. Define variables in PowerShell.

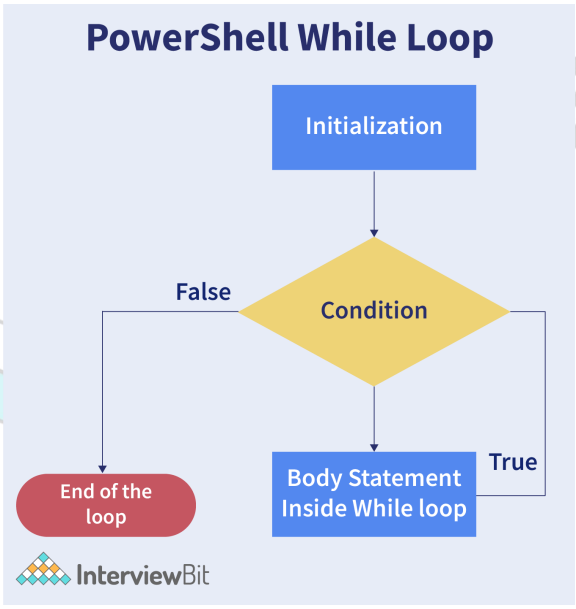
A variable is a unit of memory where values are stored. In PowerShell, variables are depicted by text strings that begin with a dollar sign ($), such as $a, $process, or $my_var. PowerShell variables are way more effective as those can be mapped to underlying classes in the framework of .NET. PowerShell considers variables as .NET objects, which implies they can save data and control data in numerous ways. Variable names in PowerShell start with a dollar sign and include a mix of numbers, letters, symbols, and spaces.

For example, $var="HELLO" saves the string HELLO in the $var variable. Variables can also possess various scopes, including global, local, script, private, and numbered scopes.

17. Explain what is a "while loop" in PowerShell?

IT professionals make use of loops when they require to complete a block of commands numerous times.

Example: An entry-controlled loop, a "while loop" runs commands consecutively as long as the provided condition is true. A bunch of IT professionals prefers to employ "while loops" rather than "for statements" as the syntax is less complex."



The following example prints the values from 1 to 5 using the while loop:

```
1. PS C:\> while($count -le 5)
2. >> {
3. >> echo $count
4. >> $count +=1
5. >> }
```

Output:

```
1
2
3
4
5
```

In this instance, the condition ($count is less than equal to 5) is true while $count = 1, 2, 3, 4, 5. Every time through the loop, the value of a variable $count is incremented by 1 utilizing the (+=) arithmetic assignment operator. When $count equals 6, the condition statement assesses to false, and the loop exits.

18. What are Automatic variables?

The automatic variables are defined as those variables that save store state information for PowerShell. These variables will include the details of a customer and the system, default and runtime variables, and PowerShell settings. These variables can be designed and handled by Windows PowerShell.

A few of the very popular Automatic Variables are mentioned below:

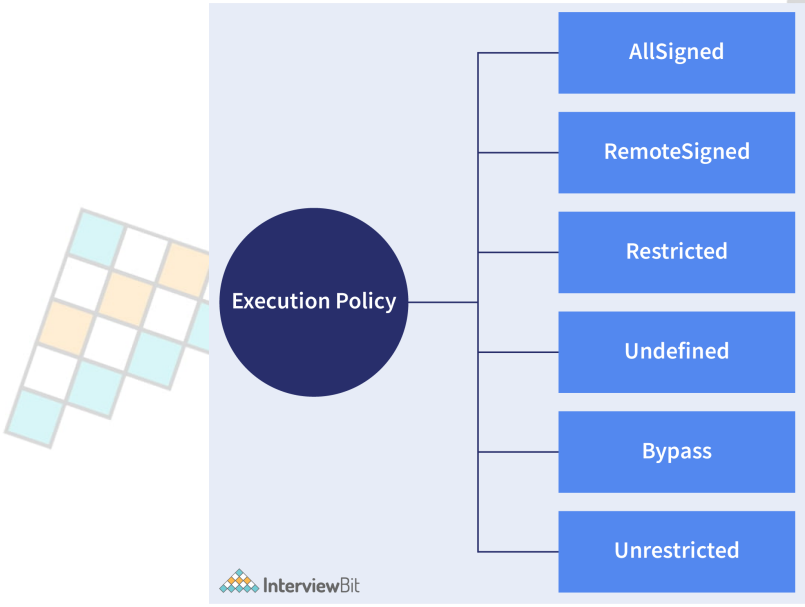
- **$**: This variable includes the last token available in the last line that is received by the session.
- **$?**: This variable may include the completion status of the last operation. If the last operation succeeded, its value is TRUE and FALSE if it has failed.
- **$^**: It may include the first token of the last line obtained by the session.
- **$Args:** Includes a collection of the undeclared parameters or parameter values. These are handed over to a script, script block, or function. When you construct a function, you can display the parameters by making use of the param keyword or by incorporating a comma-separated list of parameters in parentheses just after the function name.
- **$Error:** This variable includes an array of error objects that represent the most recent errors. The current mistake is the first error object in the array ($Error[0]).
- **$ForEach**: This variable includes the enumerator (not the resulting values) of a ForEach loop. You can make use of the properties and processes of enumerators on the value of the $ForEach variable. This variable lives only while the ForEach loop is operating; it gets deleted post the completion of the loop.
  $Home – This variable includes the full path of the customer's home directory. This variable is the counterpart of the %homedrive%%homepath% environment variables, commonly known as C:\Users<UserName>.
- **$OFS** – $OFS is a remarkable variable that saves a string (Series of characters) that you wish to utilize as an output field separator. Employ this variable when you are transforming an array into a string. By default, the value of $OFS is " ", but you can modify the value of $OFS in your session, by just typing $OFS="<value>". If you are anticipating the default value of " " in your module, script, or configuration output, be mindful that the $OFS default value has not been modified anywhere in your code.

19. Explain the importance of PowerShell brackets?

- **Parenthesis Brackets()**: These brackets are utilized for required arguments.
- **Square Brackets[]**: These sorts of brackets are used to specify the optional items.
- **Braces Brackets{}**: These kinds of brackets are utilized in blocked statements.

20. Describe the various types of execution policies in PowerShell?

PowerShell utilizes execution policies to regulate how it loads configuration files and runs scripts. Implementation of these policies only happens on Windows platforms. The PowerShell execution policies are mentioned below:



- **AllSigned**:
  - The scripts can run.
  - Needs that all scripts and configuration files be signed by a reliable publisher, which includes scripts that you compose on the local computer.
  - Encourages you before executing scripts from publishers that are still not categorized as untrusted or trusted.
  - Risks running signed, but vicious, scripts.
- **Bypass**:
  - Nothing is stopped and there are no alerts or prompts generated.
  - The Bypass execution policy is created for configurations in which a PowerShell script is constructed into a larger application or for configurations in which PowerShell is the basis for a program that possesses its security model.
- **Default:**
  - Restricted for Windows clients.
  - RemoteSigned for Windows servers.
- **RemoteSigned:**
  - The default execution policy for the Windows system.
  - Scripts can run.
  - A digital signature from an authorized publisher is required on internet downloaded scripts and configuration files. It contains email and instant messaging programs.
  - It doesn't need digital signatures on scripts that are not downloaded from the internet and composed on the local computer.
  - It executes internet downloaded scripts that are not signed if the scripts are unblocked, like utilizing the Unblock-File cmdlet.
  - Risks executing unsigned scripts from sources except for the internet and signed scripts that could be malicious in nature.
- **Restricted:**
  - The default implementation policy for Windows client computers.
  - Allows individual commands, but does not permit scripts.
  - Stops operating all script files, including configuration files (.ps1xml), PowerShell profiles (.ps1), and module script files (.psm1) formatting
- **Undefined:**
  - There is no implementation policy specified in the current scope.
  - The execution policy is Restricted for Windows clients and RemoteSigned

21. Describe what is Powershell Pipeline used for?

PowerShell pipeline is utilized for combining two statements like when the output of a statement becomes the input of the second statement.

For example,

Command-1 | Command-2 | Command-3

In this example, the objects that Command-1 emits are sent to Command-2. Command-2 processes the objects and sends them to Command-3. Command-3 processes the objects and sends them down the pipeline. As there are no more commands in the pipeline, the results are displayed at the console.

22. Why is scripting debugging important?

Scripting debugging is an essential attribute to comprehend how to use PowerShell. Concentrate on how scripting debugging can enhance your workflow and the entire project.

Scripting debugging is necessary as it permits IT professionals to quickly scan the scripts, functions, expressions, and commands while executing. This allows us to determine possible errors, maintain model scripts and improve performance.

23. Do you make PowerShell scripts to deploy components in SharePoint?

If you construct a web part with the help of VS 2010 then, you can deploy it utilizing cntrl+f5. To activate the web part component you can document a PowerShell script (.ps1) and run it after deployment.

24. Describe what is Powershell Get-command?

The Get-Command cmdlet acquires all commands that are performed on the computer, such as functions, cmdlets, aliases, applications, filters, and scripts. Get-Command fetches the commands from PowerShell commands and modules that were imported from other sessions.

Example: Get commands in the current session

This command utilizes the ListImported parameter to obtain only the commands in the current session.

```
PowerShellCopy
Get-Command -ListImported
```

25. What is your take on Variable Interpolation?

When a variable is incorporated into double-quoted strings, then PowerShell transforms the name of that variable via its value. Generally, this feature in PowerShell is emanated as variable interpolation. It provides a more readable, easier-to-read, and convenient syntax to create a formatted string.

26. What is the benefit of the hashtable in PowerShell?

A hash table is also called the dictionary. It is a collection that lets you accumulate data in a “key-value” pair association. The “key” and “value” can be of any data and length. To display a hash table you will have to make use of @ followed by curly braces.

The syntax of a hash table is as follows:

```
PowerShellCopy
@{ <name> = <value>; [<name> = <value> ] ...}
```

27. Describe what is the benefit of Array in PowerShell?

The usage of Array in PowerShell is to execute a script against remote computers. To build an array, you have to construct a variable and assign the array. Arrays are defined by “@”symbol, they are portrayed as hashtable but are not followed by curly braces.

To utilize an array in a program, you must call a variable to reference the array, and you can define the kind of array the variable can reference.

Mentioned below is the syntax for declaring an array variable:

```
$A = 1, 2, 3, 4
Or
$A = 1..4
```

> Note – By default type of object of the array is System.Object. GetType() method returns the type of the array. Type can be passed.

Example:

The mentioned code snippets are examples of this syntax –

```
[int32[]]$intA = 1500,2230,3350,4000
$A = 1, 2, 3, 4
$A.getType()
```

28. Tell about PowerShell's Get-ServiceStatus function?

The functions of Get-ServiceStatus allow filtering of window services. PowerShell documents the services that are both 'Running', and 'Stopped' during scripting. By default, when Get-Service is operated without parameters, all the local computer's services are returned. You can control this cmdlet to obtain only certain services by selecting the service name or the display name of the services, or you can pipe service objects to this cmdlet.

- Example 1: Get all services on the computer
  This example fetches all of the services on the computer. It acts as though you typed Get-Service *. The default display displays the status, service name, and display name of each service.
  Get-Service
- Example 2: Get services that commence with a search string
  This example recovers services with service names that start with WMI (Windows Management Instrumentation).
  Get-Service "wmi*"

29. What is pipeline in PowerShell?

The concept of linking together commands with pipeline operator (|) is termed as a pipeline in PowerShell. That indicates the outcome from the first command is further sent down the pipeline as input to the second command for processing and the outcome of the second command is sent down to the pipeline as input to the third command and it goes on.

30. Explain PowerShell's comparison operators?

Comparison Operators compares the value in PowerShell. There are four kinds of comparison operators that are used, namely equality, match, containment and replace. In PowerShell, one of the essential comparison operators is –eq that is utilized inplace of "=" sign for declaring variables. Similarly, there are other operators like –ne for "not equal" , -gt (greater than) or –lt (less than).

Conclusion

PowerShell is securely incorporated into nearly all of Microsoft's products. There are specific actions in popular products like Microsoft 365 and Server 2016 that cannot be accomplished with a GUI and can solely be accomplished with PowerShell. It is 100% required for certain tasks, the capability to automate with PowerShell makes comprehending it a worthwhile skill for numerous IT professionals. Once you begin to understand, all that can be accomplished with PowerShell, it unlocks a whole new set of capabilities. From fundamental automation to advanced scripting, PowerShell can deliver a lot of opportunities for streamlining tasks and saving time. The above-mentioned question and answers will help ace the interview.

Links to More Interview Questions

C Interview Questions	Php Interview Questions	C Sharp Interview Questions
Web Api Interview Questions	Hibernate Interview Questions	Node Js Interview Questions
Cpp Interview Questions	Oops Interview Questions	Devops Interview Questions
Machine Learning Interview Questions	Docker Interview Questions	Mysql Interview Questions
Css Interview Questions	Laravel Interview Questions	Asp Net Interview Questions
Django Interview Questions	Dot Net Interview Questions	Kubernetes Interview Questions
Operating System Interview Questions	React Native Interview Questions	Aws Interview Questions
Git Interview Questions	Java 8 Interview Questions	Mongodb Interview Questions
Dbms Interview Questions	Spring Boot Interview Questions	Power Bi Interview Questions
Pl Sql Interview Questions	Tableau Interview Questions	Linux Interview Questions
Ansible Interview Questions	Java Interview Questions	Jenkins Interview Questions