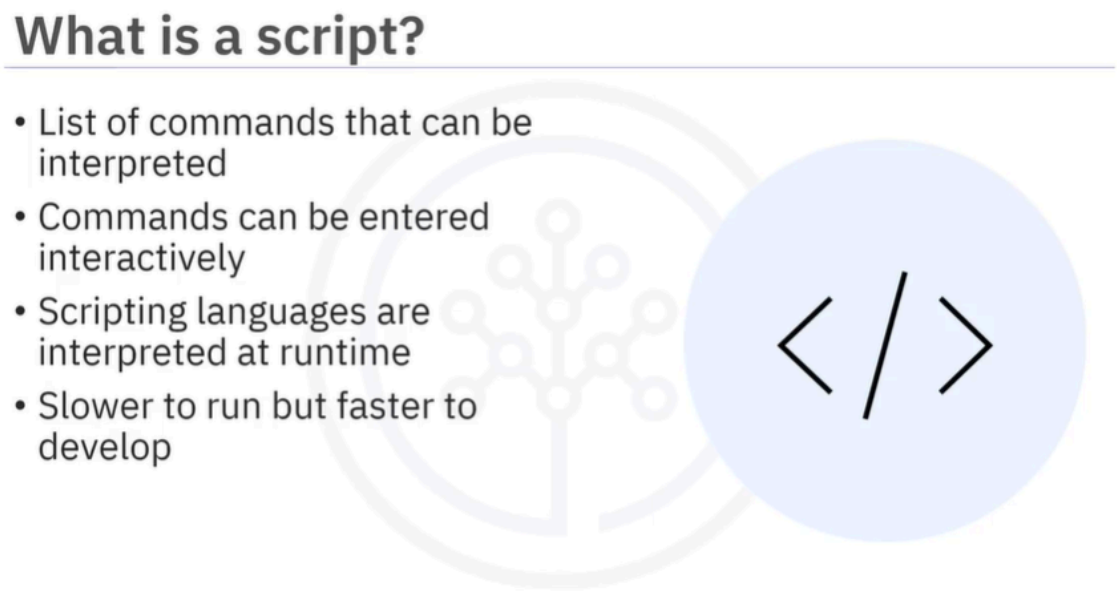
Shell Scripting Basics

Sure! Let's talk about the concept of a "shebang" in shell scripting.

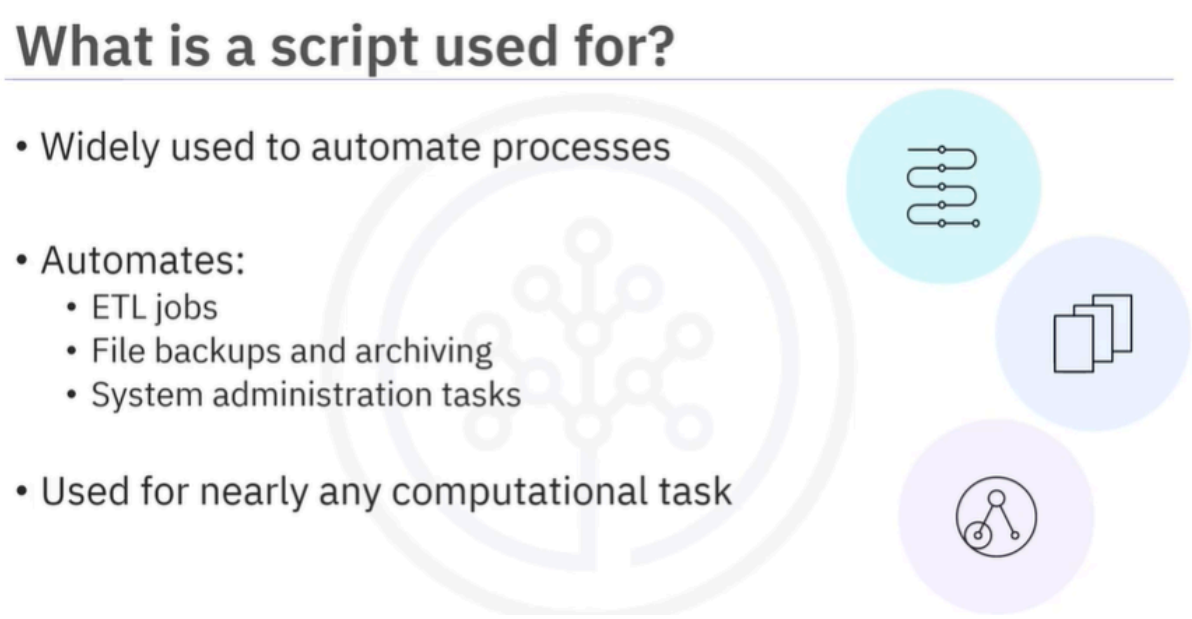
The shebang is the first line in a shell script that tells the computer which interpreter to use to run the script. Think of it like a signpost that directs the computer to the right tool for the job. For example, if you want to use the Bash shell, you would start your script with `#!/bin/bash`. This line ensures that when you run your script, the computer knows to use the Bash interpreter to understand and execute the commands written in the script.

Here's a simple analogy: imagine you're at a restaurant, and you want to order food. The menu (your script) tells the chef (the interpreter) what to prepare. If you don't specify which chef to go to, the waiter might not know who to ask, and your order could get lost! So, the shebang is like saying, "Hey, chef, I want you to make this dish using this specific recipe!"



- A script is a list of commands that can be interpreted and run by a program called scripting language. Commands can be entered interactively at the

command line, or listed line by line in a text file. Scripting languages are usually not compiled. They are interpreted at runtime. Scripts are generally slower to run than compiled languages, but they are also much easier and faster to develop.



- Scripts are widely used to automate processes, such as ETL jobs, file backups and archiving, and general system administration tasks. You can use scripts for nearly any computational task including, application integration, and plug-in and web application development

Shell scripts and the ‘shebang’

- Shell script – executable text file with an *interpreter directive*
  - a.k.a ‘shebang’ directive:

```
#!interpreter [optional-arg]
```

  - *‘interpreter’* – path to an executable program
  - *‘optional-arg’* – single argument string

- A shell script is an executable text file in which the first line usually has the form of an interpreter directive. The interpreter directive is also known as a ‘shebang’ directive, and has the following form: 'pound, bang, interpreter' plus an optional argument. Interpreter is an absolute path to an executable program, and the optional argument is a string representing a single argument.

Example – ‘shebang’ directives

Shell script directives:

```
#!/bin/sh
#!/bin/bash
```

Python script directive:

```
#!/usr/bin/env python3
```

- Shell scripts are scripts that invoke a shell program. For example: '#!/bin/sh' invokes the Bourne shell or other compatible shell program, from the bin directory. '#!/bin/bash' 'shebang' invokes the Bash shell. 'Shebang' directives aren't limited to shell programs. For example, you could create a python script with the following directive: '#!/usr/bin/env python3'. Here we demonstrate how a shell script can be used to run a program with a simple 'hello world' example.

'Hello World' example shell script

Create the shell script:

```
$ touch hello_world.sh
$ echo '#! /bin/bash' >> hello_world.sh
$ echo 'echo hello world' >> hello_world.sh
```

'Hello World' example shell script

Make it executable:

```
$ ls -l hello_world.sh
-rw-rw-r-- 1 jgrom jgrom 12 Jun 27 09:13 hello_world.sh
$ chmod +x hello_world.sh
$ ls -l hello_world.sh
-rwxrwxr-x 1 jgrom jgrom 12 Jun 27 09:13 hello_world.sh
```

‘Hello World’ example shell script

Run your bash script:

- Here we demonstrate how a shell script can be used to run a program with a simple ‘hello world’ example. From the command prompt, you can create a simple ‘hello world shell script’ as follows: Use the touch command to create an empty text file called ‘hello_world.sh’ The ‘.sh’ extension is a convention used to indicate that the file is a shell script. Now, turn your text file into a bash script by echoing the bash shebang, and appending that echoed text to your file using the ‘double greater than symbol,’ which is the Bash ‘output redirection’ operator used for appending output to file. Finally, use the echo command to print the statement ‘echo Hello World’ and again, redirect that output to your bash script. Before you can run your bash script, you need to make it executable.
- First, check the current permission settings of your script by using the ‘ls’ command with the ‘-l’ option. The R and W entries indicate that the script is readable and writable, but the lack of an X means it is not executable. The R, W, and X entries apply to three user-based permission groups: the owner, which is you, the group, and all users. You can make it executable for all users by invoking the ‘chmod’ command with the 'plus X' option. Checking the permission settings now shows that your script is executable for all users, due to the ‘X’ entries for all three permission groups.
- Now you can finally run your bash script by typing ‘./hello_world.sh’ and pressing enter, and the text ‘hello world’ appears on the command line.