Shell Scripting Basics:

1. What is a Shell?

- A shell is a command-line interpreter that provides a user interface for the operating system.

2. Types of Shells:

- Bash (Bourne Again SHell): Most common on Linux systems.

- sh (Bourne Shell): An older shell that is still widely used.

- csh (C Shell): C-like syntax, often used in academic environments.

- ksh (Korn Shell): Developed by David Korn at Bell Labs, an extension of the Bourne shell.

- zsh (Z Shell): A powerful and feature-rich shell.

3. Starting a Shell:

- Open a terminal (command-line interface).

- Type the name of the shell (e.g., `bash`, `sh`, `zsh`) and press Enter.

Creating Your First Script - Hello World:

1. Choose a Text Editor:

- Use a text editor to create your script. Common ones include `nano`, `vim`, `gedit`, or `code`.

2. Create a New File:

- Open your chosen text editor and create a new file. For example:

```bash

nano hello.sh

```

3. Write Your Script:

- In the text editor, enter the following script:

```bash

!/bin/bash

echo "Hello, World!"

```

- The first line `!/bin/bash` is called a shebang and tells the system to use the Bash shell to execute the script.

4. Save and Exit:

- Save the file and exit the text editor.

5. Make the Script Executable:

- In the terminal, make the script executable:

```bash

chmod +x hello.sh

```

6. Run Your Script:

- Execute the script by typing:

```bash

./hello.sh

```

- You should see the output: `Hello, World!`

Detailed Notes:

- Shebang (`!/bin/bash`):

- Specifies the path to the interpreter (Bash in this case).

- Comments (``):

- Lines starting with `` are comments and are ignored during script execution.

- Commands (`echo`):

- Commands are the actions your script takes. `echo` is used to print text to the terminal.

- File Permissions (`chmod`):

- `chmod +x` makes the script executable (`+x` adds execute permission).

- Running the Script (`./hello.sh`):

- `./` is used to run a script in the current directory.

Additional Tips:

- Variables:

- You can use variables to store and manipulate data.

- Conditionals and Loops:

- `if`, `else`, `elif` for conditional execution.

- `for` and `while` loops for repetitive tasks.

- Functions:

- Break your script into functions for better organization and reusability.