Table of Contents

1. What is Bash?

- Bash (Bourne Again SHell) – A command-line interpreter for Unix/Linux.
- Purpose: Automate tasks, run commands, and create reusable scripts.
- Script File Extension: .sh
- Execute Script:

```
bash script.sh
./script.sh  (if executable)
```

---

2. Creating and Running a Script

1. Create the File:

   ```
   nano myscript.sh
   ```

2. Add the Shebang (Interpreter Directive):

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

3. Make the Script Executable:

   ```
   chmod +x myscript.sh
   ```

4. Run the Script:

   ```
   ./myscript.sh
   ```

---

3. Basic Script Structure

```
#!/bin/bash
# This is a comment
echo "Hello, World!"
```

---

4. Variables

- Define:

  ```
  name="John"
  ```

- Use:

  ```
  echo "Hello, $name!"
  ```

- User Input:

  ```
  read user_input
  ```

---

5. Conditional Statements

If-Else

```
if [ $age -ge 18 ]; then
  echo "Adult"
else
  echo "Minor"
fi
```

Comparison Operators:

- -eq – Equal
- -ne – Not equal
- -gt – Greater than
- -lt – Less than
- -ge – Greater than or equal
- -le – Less than or equal

---

6. Loops

For Loop:

```
for i in {1..5}; do
  echo "Iteration $i"
done
```

While Loop:

```
count=1
while [ $count -le 5 ]; do
  echo "Count: $count"
  ((count++))
done
```

---

7. Functions

```
greet() {
  echo "Hello, $1"
}
greet "Alice"
```

---

8. Command Line Arguments

- $0 – Script name
- $1, $2… – Positional arguments
- $# – Number of arguments
- $@ – All arguments

```
echo "Script name: $0"
echo "First argument: $1"
```

---

9. Error Handling

```
command || echo "Command failed"
command && echo "Command succeeded"
```

---

10. File Operations

```
if [ -f "file.txt" ]; then
  echo "File exists"
else
  echo "File not found"
fi
```

- -f – File exists
- -d – Directory exists
- -e – File or directory exists

---

11. Script Debugging

- Run with Debugging:

  ```
  bash -x script.sh
  ```

- Add Debugging Inside Script:

  ```
  set -x  # Enable
  set +x  # Disable
  ```

---

12. Scheduling with Cron

- Open Crontab:

  ```
  crontab -e
  ```

- Schedule Example (Every Day at 5 PM):

  ```
  0 17 * * * /path/to/script.sh
  ```

---

13. Useful Shortcuts

- Ctrl + C – Stop execution
- Ctrl + Z – Suspend
- Ctrl + D – End input
- !! – Repeat last command
- !n – Run nth command from history

Example: Backup Script

```
#!/bin/bash
backup_dir="/backup"
mkdir -p $backup_dir
cp *.txt $backup_dir
echo "Backup completed!"
```