LAB-3.md

Lab 3: Modifying an Existing Script

Goal: Take a simple script and supercharge it with more flexibility.

🍀 Base Script vs. ⚡ Upgraded Script

🧃 The Original: print_numbers.sh

- This script is simple-minded — it prints the numbers 1 through 5, always the same, without caring what you want. - - Think of it like a jukebox that only has one track: predictable, stable, but not very exciting.

```
for i in 1 2 3 4 5
do
  echo "Number: $i"
done
```

⚡ The Upgraded: enhanced_numbers.sh

- This version isn't just a loop anymore — it's a mini tool with personality.

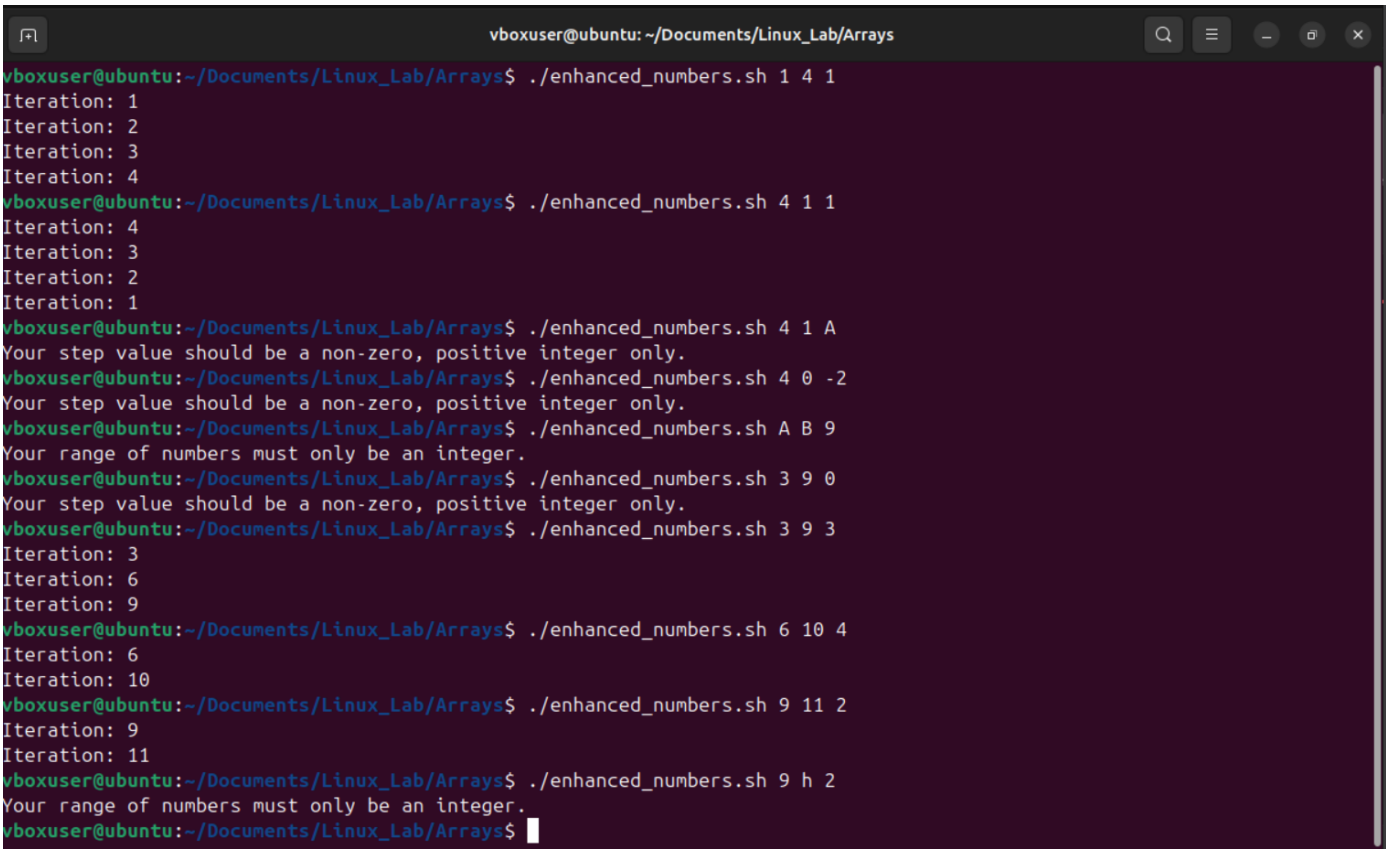
✨ What's new?

- ✅ Checks your input before running (no nonsense allowed)
- 🔄 Can count forwards or backwards
- 🚫 Prevents mistakes like a zero step
- 🧩 Takes 3 arguments: start, end, and step
- 🌊 Lets you control speed & direction with ease

🔍 Logic in Action

- You pass 3 numbers when running the script.
- It verifies if those numbers make sense.
- Based on them, it prints your custom sequence — up or down, fast or slow.

🧪 Example Run



📝 Extra Questions

Q1. What does $1 mean in Bash?

- 👉 $1 is the first argument given to the script.
- Example: ./script.sh cat dog → $1 = cat

Q2. What about $@?

- 👉 $@ expands to all the arguments, separated individually.
- Useful for loops:

```
for arg in "$@"; do
  echo "$arg"
done
```

Q3. What does $# tell us?

- 👉 $# = number of arguments supplied.
- Example: `./script.sh a b c → $# = 3`

Q4. What does exit 1 mean?

- 👉 It stops the script and returns error code 1, which signals failure.
- Often used if user input is wrong.

```
if [ $# -ne 3 ]; then
  echo "Error: Need 3 arguments."
  exit 1
fi
```

Q5. Difference between exit 0 and exit 1?

- `exit 0` → Everything went fine (success).
- `exit 1` (or other non-zero values) → Something went wrong (failure).