

# **LAB NOTEBOOK**

**SWASTI KUMARI**

**Summer internship**

**Beginning 26 May 26, 2025**

# Monday, May 26, 2025

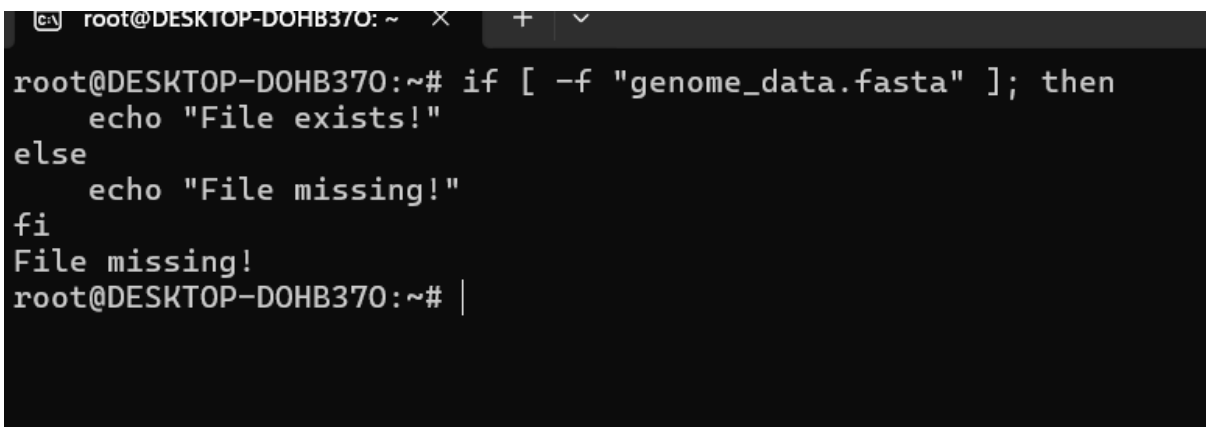
## Chapter 5. Flow Control

**Topic:** Flow Control- if-else, return, and exit

Today, I explored key flow control concepts in Bash scripting that help automate tasks and manage execution behaviour efficiently.

### 1. if-else Statements

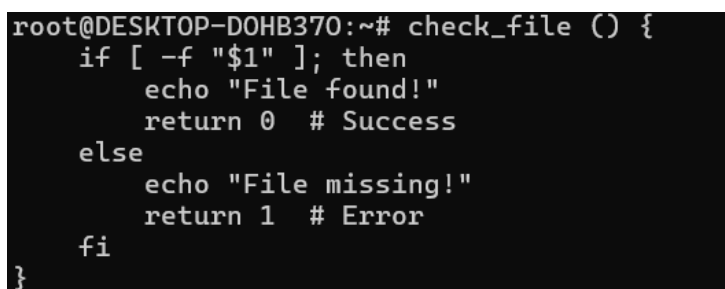
- The if statement allows conditional execution based on whether an expression is **true** (exit status 0) or **false** (exit status  $\neq 0$ ).
- **Example: Checking if a file exists**



```
root@DESKTOP-DOHB370: ~  
root@DESKTOP-DOHB370:~# if [ -f "genome_data.fasta" ]; then  
    echo "File exists!"  
else  
    echo "File missing!"  
fi  
File missing!  
root@DESKTOP-DOHB370:~# |
```

### 2. return Statement in Functions

- Used **only inside functions** to exit and pass an exit status.
- Syntax: return N (where N is an integer between 0-255).
- **Example:**



```
root@DESKTOP-DOHB370:~# check_file () {  
    if [ -f "$1" ]; then  
        echo "File found!"  
        return 0 # Success  
    else  
        echo "File missing!"  
        return 1 # Error  
    fi  
}
```

### 3. exit Statement

- Terminates **the entire script**, unlike return, which only exits a function.
- Syntax: exit N (similar to return).

The `exit` command in Bash is used with the syntax, `exit [n]`. Its function is to terminate a script and return a value to the parent script or shell. It's a way to signal the end of a script's execution and optionally return a status code to the calling process.

- `#!/bin/bash`
- `echo 'Hello, World!'`
- `exit 0`
- In this example, we've created a simple Bash script that prints 'Hello, World!' to the console and then terminates with an exit status of 0. The `exit 0` command signals successful execution of the script

## Tuesday, 27 May 2025

### 1. Retrieval and Preparation of FASTA Sequence

- Downloaded a **FASTA file** from **NCBI** for genomic analysis.
- Verified the file location using `ls` in the terminal.
- Navigated to the **Downloads** directory to ensure correct execution.

### 2. Development and Execution of Bash Script

1. Created a Bash script named **fasta\_analysis.sh** to automate sequence analysis.
2. Implemented **loops** (`while read -r line`) for efficient line-by-line processing.
3. Extracted sequence identifiers (headers) using conditional statements.
4. Computed the **total sequence length** using `${#sequence}`.
5. Calculated the **GC percentage** by counting occurrences of G and C bases:

### 3. Explored exit statuses (\$?), understand and performed how commands signal success or failure.

### 4. Practiced condition testing (`-eq`, `-ne`, `-lt`, `-gt`) for numerical comparisons.

### 5. Used string comparisons (`=`, `!=`, `-z`, `-n`) to validate file extensions and sequence headers.

## Wednesday, 28 May 2025

### 1. File Attribute Checks:

- Practiced using file test operators:
  - `-f` → check if a file exists.
  - `if [ -f "$file" ]` → conditional used to verify file presence before proceeding with analysis.

### 2. Integer Conditionals:

- Used `-gt`, `-lt`, `-eq` etc. for numeric comparisons.
- Example:

```
if [ "$length" -gt 100 ]; then
```

```
    echo "Sequence is longer than 100 bp"
```

### 3. String Conditionals:

- Learned how to compare strings:

- = for equality, != for inequality.
- Lexicographical comparisons using <, > (escaped as \< and \> in [ ] brackets).
- Example:  
if [ "\$gene" = "WNT2" ]; then  
echo "Gene of interest: WNT2"

#### 4. FASTA File Handling:

- Checked FASTA format using head -n 1 and pattern match >.
- Removed FASTA headers using: grep -v "^>"
- Used tr -d '\n' to convert multi-line sequence into a single line.

#### 5. Mini Projects & Scripts:

Created and executed the following bash scripts:

- **check\_fasta.sh** – Validates if a file is in FASTA format.
- **check\_length.sh** – Checks if sequence length exceeds 100 base pairs.
- **gc\_content.sh** – Calculates GC content of a sequence.

Each script used:

- Conditional logic (if-else)
- Commands like grep, wc, tr, echo, and head.

#### 6. Project Contribution:

- Contributed to a **WNT2 gene** bioinformatics project.

## Thursday, 29 May 2025

1. Completed **Chapter 5**, which covered control flow structures including:

- for loop
- case statements
- while loop
- until loop

2. Practiced and implemented the above loops using **FASTA sequences** for hands-on application.

3. Wrote and executed scripts to process sequence data and understand the use of different loop types in bioinformatics tasks.

## Friday, 30 May 2025

1. Revised **Chapters 1 to 5**, which included:

- **Chapter 1:** Introduction to Shell Scripting
  - **Chapter 2:** Variables and Data Types
  - **Chapter 3:** Operators
  - **Chapter 4:** Conditional Statements
  - **Chapter 5:** Loops (for, while, until, case)
2. Shot and recorded explanatory videos for:
- **Chapter 1**
  - **Chapter 2**

