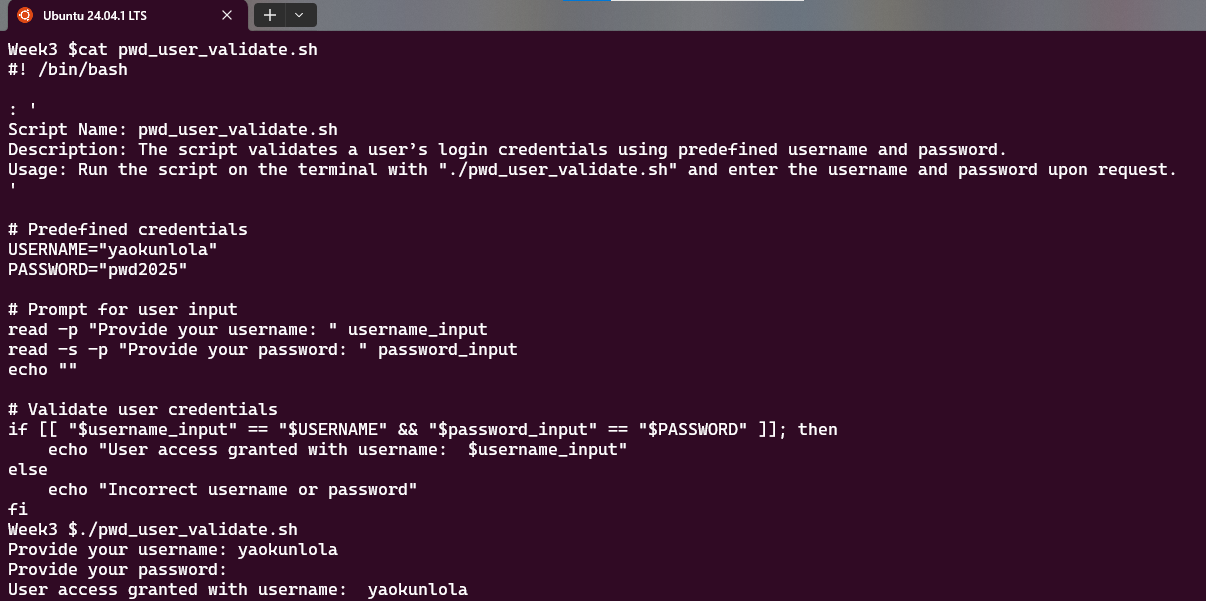
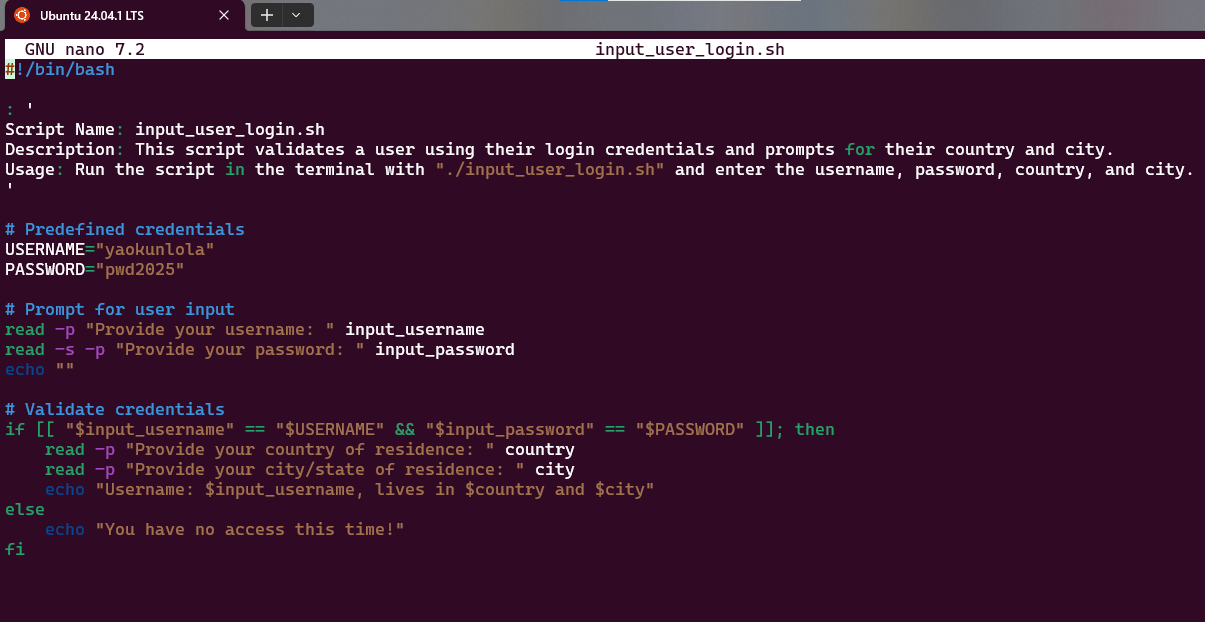
Bash Scripting programming Solutio

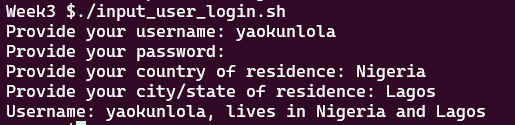
# Section A: Basic Scripting

1. Create a script “*pwd\_user\_validate.sh*” using predefined username and password variables. The script should validate the user's input with an if-else statement.
   * print: "User access granted with username [username]" (where [username] is the entered username).
   * If the input does not match, print "Incorrect username or password".

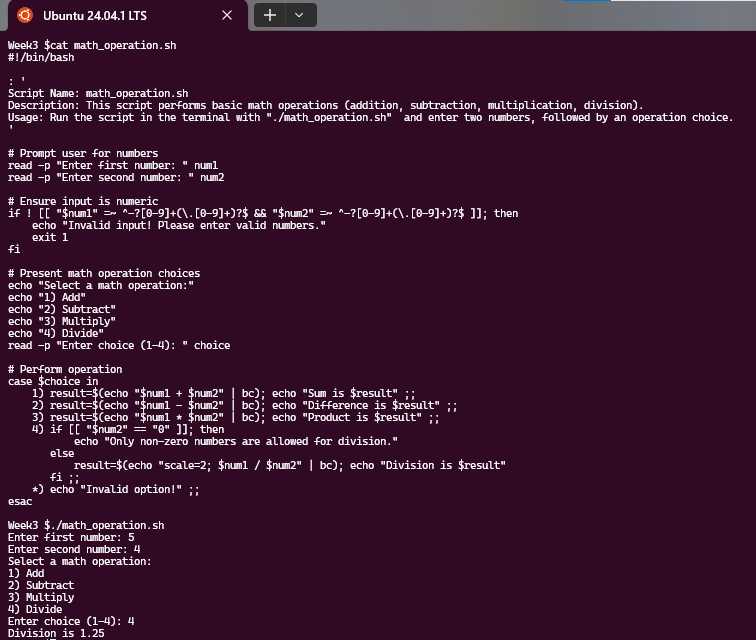


1. Create a script “*input\_user\_login.sh*” that asks a user for their login and password.
   * If the credentials are correct, prompt the user for their nation and city of residence and print: "Username: [username], lives in [country] and [city/state]"
   * If credentials are invalid, print "You have no access this time!".

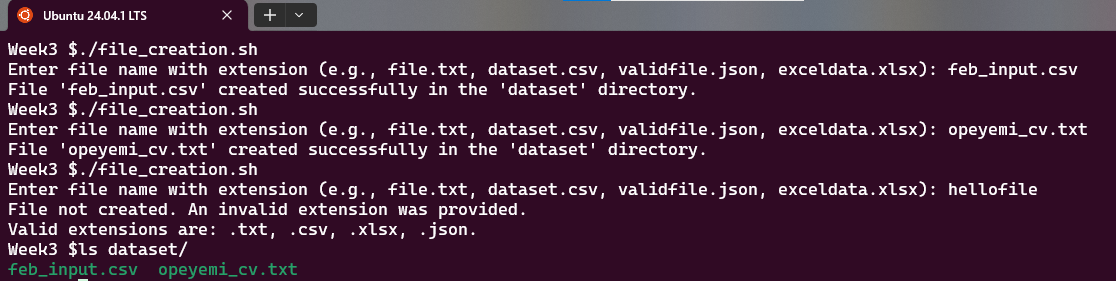




1. Create a script “*math\_operation.sh*” that allows the user to enter two numbers and then does one of four operations: addition, subtraction, multiplication, or division.
   1. Options include: 1) Add, 2) Subtract 3) Multiply and 4) Divide (Use case statement).
   2. Avoid division by zero.
   * If the second number is zero during division, print "Only non-zero numbers are allowed for division."
   * For valid operations, compute and print the result. For example, in addition: "Sum is [result]".
   * If the selection does not match, print "Invalid option!"
   * Verify that the entered numbers are valid.
   * Only two numbers are strictly allowed.



1. Create a script called "*file\_creation.sh*" that generates files and checks for valid file extensions. The files are created using prompts and saved in a subdirectory called "dataset".
   * Make a folder named "dataset" if it does not already exist.
   * Prompt users to enter file names with extensions (e.g.,.txt,.csv,.json,.xlsx).
   * Extensions supported: .txt,.csv,.xlsx, and.json.
   * If an invalid extension is entered, print: "File not created." An invalid extension.
   * Valid extensions expected include.txt,.csv,.xlsx, and.json.
   * Create a file in the "dataset" folder with the appropriate extension.

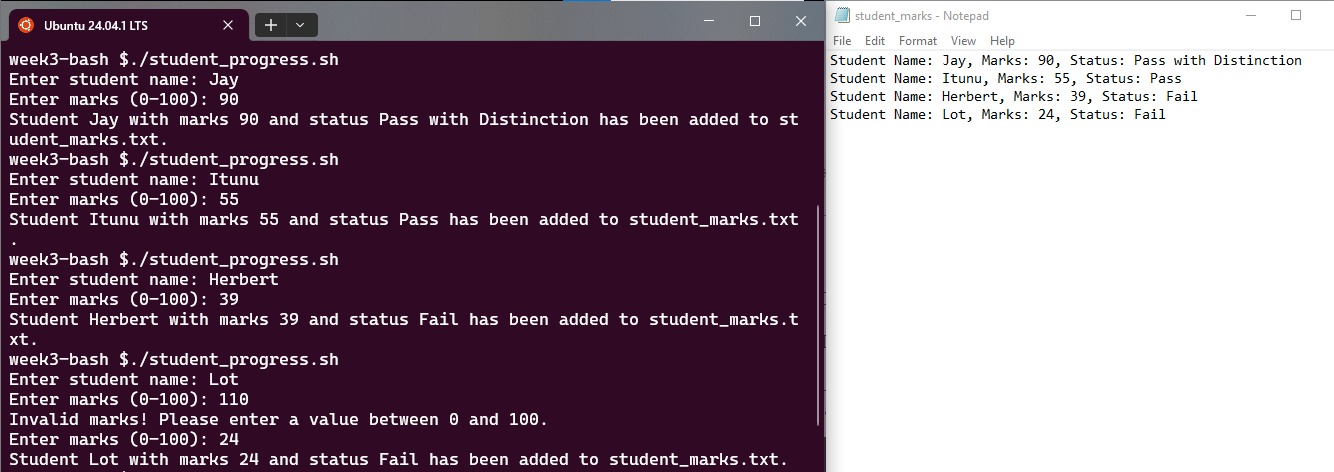


# Section B: Advanced Scripting

1. Create a script to automatically identify a student's progress status (Pass, Fail, or Pass with Distinction) depending on their grades. Every time the script is executed, the data should be appended to an existing file called "student\_marks.txt".

Requirements:

* + Make sure the file "student\_marks.txt" already exists in your preferred folder.
  + The script should prompt users to provide a student's name and marks.
  + The script should determine a student's progress based on their marks.
  + *Pass with Distinction* if marks exceed 75%.
  + *Pass* if score is 50%-75%.
  + Marks under 50%: Fail
  + Marks should not exceed 100 or fall below zero. If the input is outside this range, the script should request the user to provide proper marks again.
  + Add the following information to the file: Student Name: [name], Marks: [marks], and Status: [status].
  + Print a confirmation message after appending: "Student [name] with marks [marks] and status [status] has been added to student\_marks.txt".



1. Create a Bash script with a function that accepts arguments. The function should:
   * Output the script's name.
   * Display the number of arguments supplied to the script (*minimum 1 and maximum 6).*
   * Print each argument using a for-loop.
   * Ensure each parameter is a string with only alphabetic letters. If an argument contains numbers or special characters, display a notice that says, "You must enter characters only."
   * If the number of arguments exceeds 1-6 or contains invalid characters, use a while loop to request valid arguments.
   * Once appropriate inputs are provided, the script should generate a file named e.g., "arg1\_arg2\_arg3\_arg4.txt" (with all arguments concatenated with underscores).
   * The file should be created in the local directory, and a confirmation message should be printed once it has been successfully created.

