Assignment 3:

Create a function that takes a filename as an argument and prints the number of lines in the file. Call this function from your script with different filenames.

Sol:

```
#!/bin/bash

# Function to count the number of lines in a file
count_lines_in_file() {
  local filename="$1"

  # Check if the file exists and is a regular file
  if [ -f "$filename" ]; then
    # Count the number of lines in the file
    local line_count=$(wc -l < "$filename")
    echo "The file '$filename' has $line_count lines."
  else
    echo "The file '$filename' does not exist or is not a regular file."
  fi
}

# Call the function with different filenames
count_lines_in_file "myfile.txt"
count_lines_in_file "anotherfile.txt"
count_lines_in_file "nonexistentfile.txt"
```

Explanation:

1. **Shebang (`#!/bin/bash`)**:
    - This line specifies that the script should be executed using the Bash shell.
2. **Function Definition**:

- **`count_lines_in_file():`** A function that takes a filename as an argument and prints the number of lines in that file.
- **`local filename="$1":`** The `filename` variable is assigned the value of the first argument passed to the function.

3. **File Existence and Type Check**:

   - **`if [ -f "$filename" ];:`** Checks if the file exists and is a regular file.

4. **Counting Lines**:

   - **`local line_count=$(wc -l < "$filename"):`** Uses `wc -l` to count the number of lines in the file and assigns the result to `line_count`.
   - **`echo "The file '$filename' has $line_count lines.":`** Prints the number of lines in the file.

5. **Error Handling**:

   - If the file does not exist or is not a regular file, the script prints an appropriate error message.

6. **Calling the Function**:

   - The script calls `count_lines_in_file` with different filenames (`myfile.txt`, `anotherfile.txt`, `nonexistentfile.txt`).

Customizing the Script

- You can easily customize the script by changing the filenames passed to the `count_lines_in_file` function or by adding more filenames as needed.