National University
of computer and emerging sciences

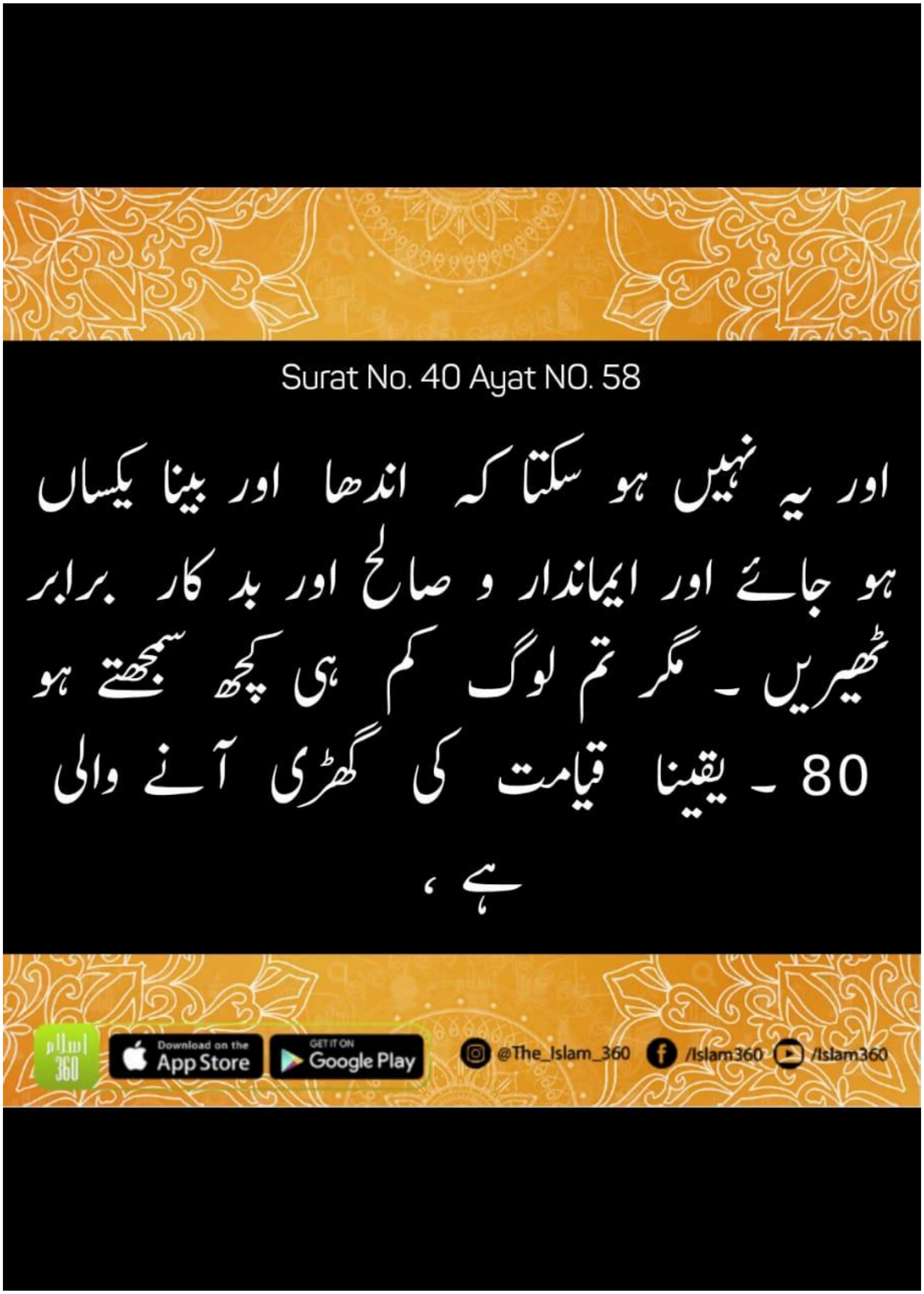
CL 2006 OPERATING SYSTEM

BS Software Engineering

Fall-2024

Course Instructors: Syed Daniyal Hussain Shah, Hifza Umar

Course Coordinator: Syed Daniyal Hussain Shah



Surat No. 40 Ayat NO. 58

اور یہ نہیں ہو سکتا کہ اندھا اور بینا یکساں ہو جائے اور ایماندار و صالح اور بد کار برابر ٹھیریں ۔ مگر تم لوگ کم ہی کچھ سمجھتے ہو 80 ۔ یقینا قیامت کی گھڑی آنے والی ہے ،

اسلام 360
Download on the App Store
GET IT ON Google Play
@The_Islam_360
/Islam360
/Islam360

Lab Policy and Rules:

1. **100% attendance is mandatory.** In case of an emergency, you can avail yourself of up to 3 absences. So, don't waste your absences , save them for emergencies. If you are debarred from the course due to low attendance, do not come to me to correct it.

2**. Disturbing the class environment during lab sessions.** such as by talking, using mobile phones, eating, etc. will result in penalties (e.g., deduction of marks).

3. **Lab tasks will not be accepted if you are absent.** If you miss a lab, you cannot submit that lab task.

4**. Lab demos for each lab task will be conducted at the end of each session.** If you miss the demo for any reason, no retake will be allowed, and that lab will be marked as 0.

5. **All quizzes will be unannounced**. Be prepared for surprise quizzes. Quizzes will cover content from previous labs as well as theory lectures from the corresponding theory course.

6. **You can take help from the internet for lab tasks,** but simply copying and pasting without understanding will be marked as 0. You should be able to explain the syntax or material used in your tasks.

7**. Do not ask for personal favors.** If you have concerns, such as short attendance, please speak with the relevant authority (e.g., academics).

8**. Students on warning:** Now is the time to study and earn a good grade. Do not ask for extra marks at the end if you are unable to clear your warning.

Lab 3: Shell Scripting

Shell Scripting:

A shell script is a computer program designed to be run by the shell. A shell script is a file containing a series of commands. The shell reads a file and carries out the commands as though they have been entered directly on the command line.

Types of shell:

There are several types of shell listed below:

1: bash (/bin/bash)

2: sh (/bin/sh)

3: ksh (/bin/ksh)

4: csh (/bin/csh)

5: zsh (/bin/zsh)

6:tcsh (/bin/tcsh)

7:fish (/bin/fish)

8:dash (/bin/dash)

How to check default shell in terminal ?

Command: echo $SHELL

Step 1: Create a New Shell Script File:

Open a terminal.

Use a text editor to create a new script file.

```
nano myscript.sh
```

Step 2: Write the Script:

```
#!/bin/bash
ls
pwd
cal
```

#!/bin/bash tells the system that this script should be run using the Bash shell.

ls lists the files in the current directory.

pwd prints the current working directory.

cal displays the calendar for the current month.

Step 3: Save and Exit:

press CTRL + O to save the file, then press Enter.

Press CTRL + X to exit the editor.

Step 4: Make the Script Executable

Before you can run the script, you need to give it execute permissions.

Step 5: Execute the Script

```
./myscript.sh
```

Echo command:

You can use the `echo` command to display statements:

```
#!/bin/bash
echo "Hello, World!"
```

bc: Best suited for floating-point arithmetic and complex calculations.

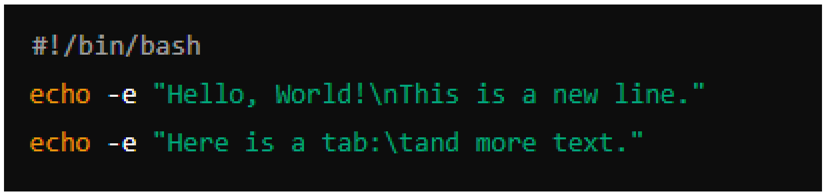
awk: Best suited for text manipulation, pattern matching, and working with structured data, while also being capable of performing basic arithmetic.

-e option:

The -e option allows interpretation of backslash escapes. For example:

\n: Newline

\t: Tab

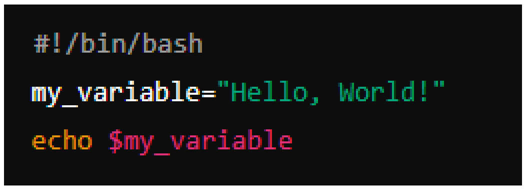


```bash
#!/bin/bash
echo -e "Hello, World!\nThis is a new line."
echo -e "Here is a tab:\tand more text."
```

Variables:

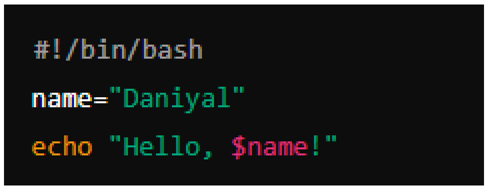
Variables in shell scripting do not require a data type declaration (e.g., `int`, `string`).

Variables are case sensitive and we cannot use spaces around = sign while assiging value to variables. You simply assign values like this:



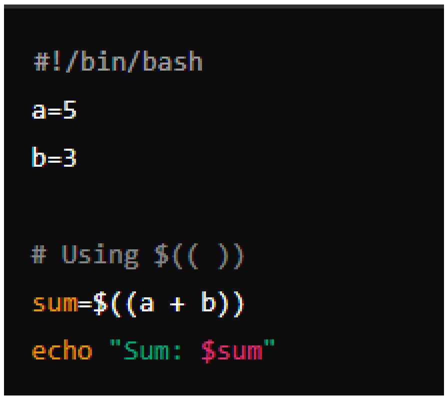
```bash
#!/bin/bash
my_variable="Hello, World!"
echo $my_variable
```

You can embed variables within strings:



```bash
#!/bin/bash
name="Daniyal"
echo "Hello, $name!"
```

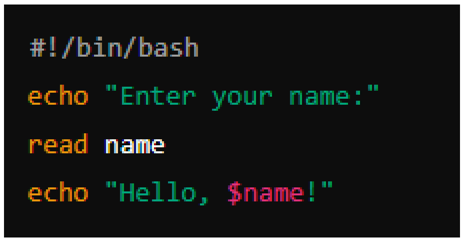
For arithmetic operations, you can use the $(()) syntax



```bash
#!/bin/bash
a=5
b=3

# Using $(( ))
sum=$((a + b))
echo "Sum: $sum"
```

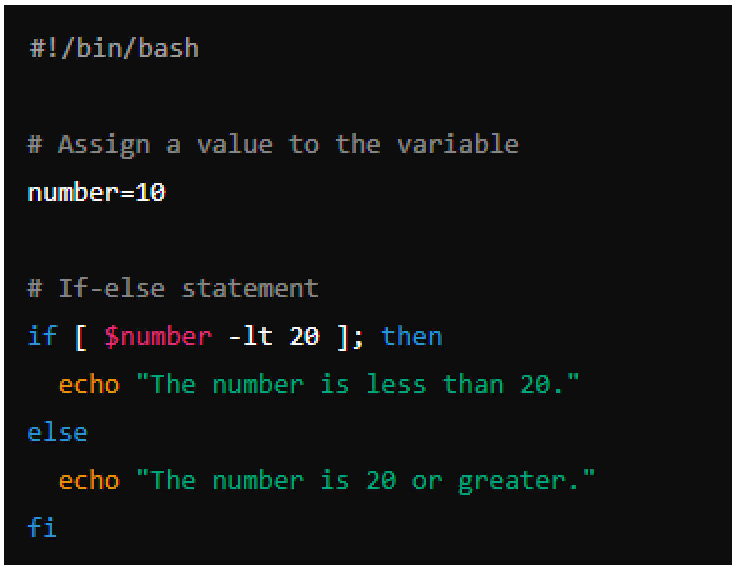
You can read input from the user and store it in a variable using the `read` command:



```
#!/bin/bash
echo "Enter your name:"
read name
echo "Hello, $name!"
```

Conditional Statements:

An example of an if-else statement in a shell script:

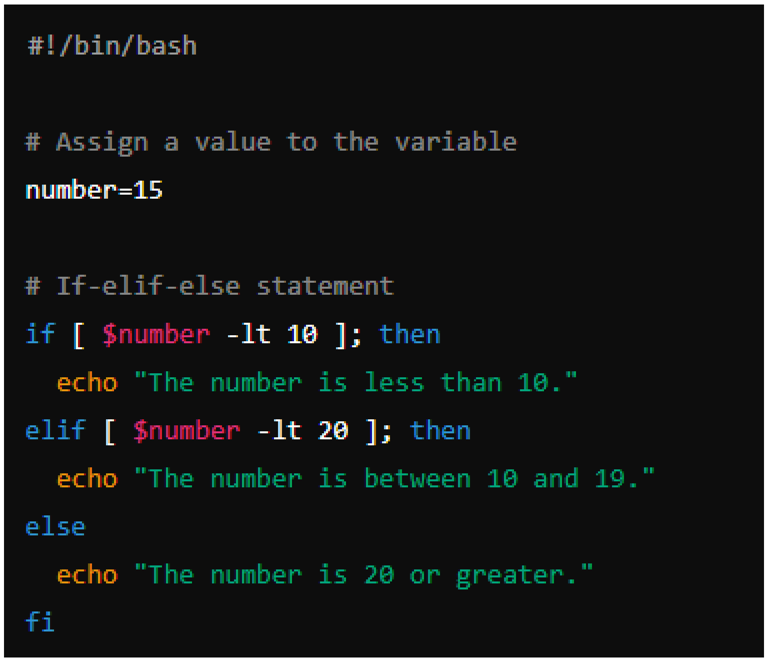


```
#!/bin/bash

# Assign a value to the variable
number=10

# If-else statement
if [ $number -lt 20 ]; then
  echo "The number is less than 20."
else
  echo "The number is 20 or greater."
fi
```

The if-elif-else structure in Bash allows you to check multiple conditions. It works similarly to if-else if-else in other programming languages.



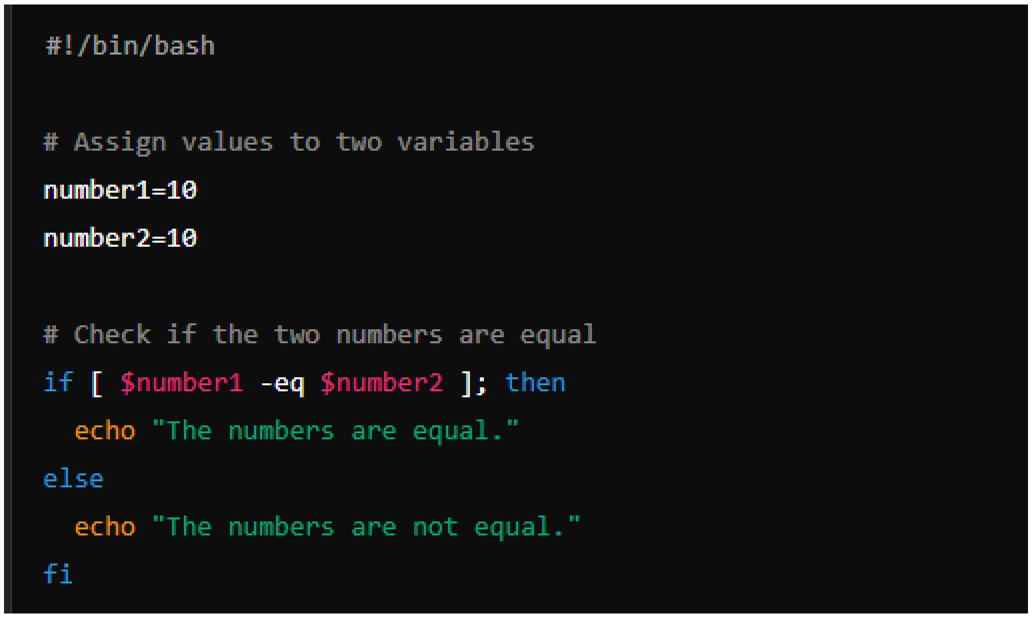
```bash
#!/bin/bash

# Assign a value to the variable
number=15

# If-elif-else statement
if [ $number -lt 10 ]; then
  echo "The number is less than 10."
elif [ $number -lt 20 ]; then
  echo "The number is between 10 and 19."
else
  echo "The number is 20 or greater."
fi
```

-eq:

-eq is used to compare two numbers for equality. It stands for "equal" and is used in conditional statements (like if) to check if two integers are equal.



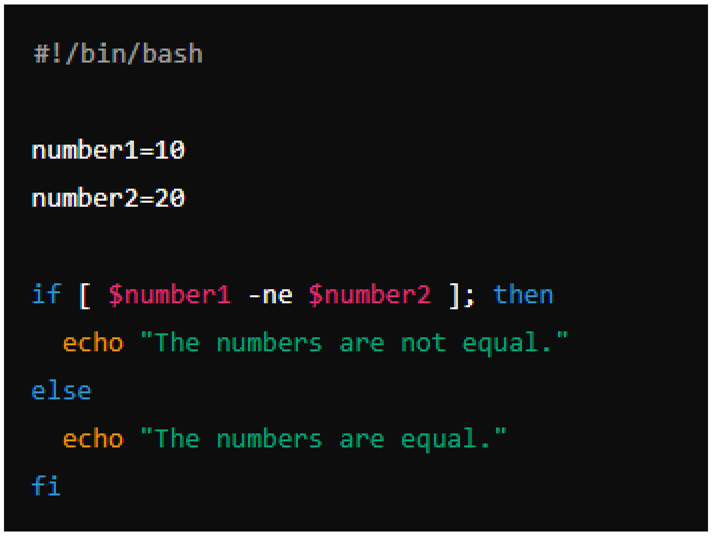
```bash
#!/bin/bash

# Assign values to two variables
number1=10
number2=10

# Check if the two numbers are equal
if [ $number1 -eq $number2 ]; then
  echo "The numbers are equal."
else
  echo "The numbers are not equal."
fi
```

-ne:

checks if two numbers are not equal.



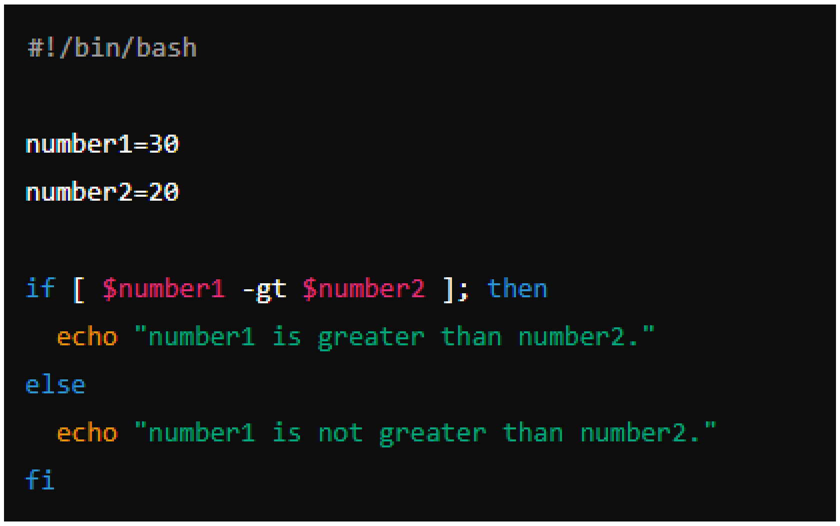
```bash
#!/bin/bash

number1=10
number2=20

if [ $number1 -ne $number2 ]; then
  echo "The numbers are not equal."
else
  echo "The numbers are equal."
fi
```

-gt:

checks if one number is greater than another.



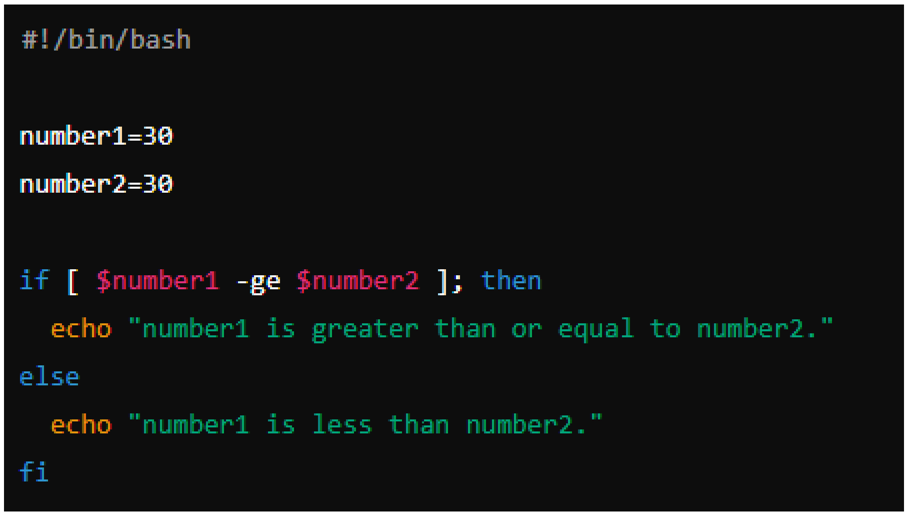
```bash
#!/bin/bash

number1=30
number2=20

if [ $number1 -gt $number2 ]; then
  echo "number1 is greater than number2."
else
  echo "number1 is not greater than number2."
fi
```

-ge:

which checks if one number is greater than or equal to another.

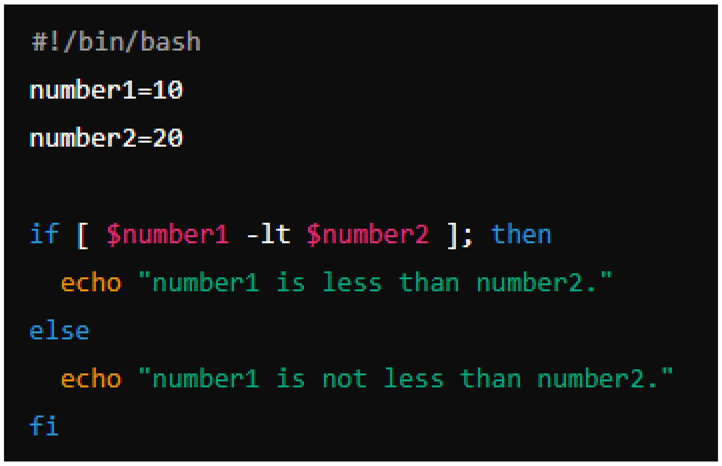


```bash
#!/bin/bash

number1=30
number2=30

if [ $number1 -ge $number2 ]; then
  echo "number1 is greater than or equal to number2."
else
  echo "number1 is less than number2."
fi
```

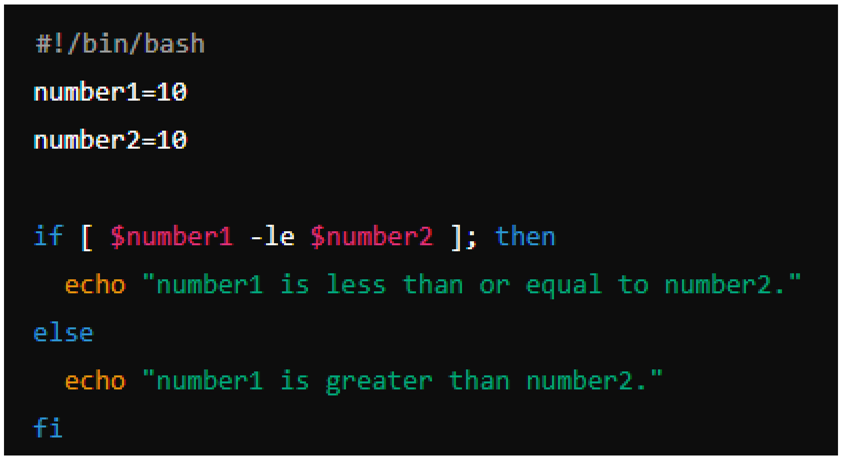
-lt (Less than):



```bash
#!/bin/bash
number1=10
number2=20

if [ $number1 -lt $number2 ]; then
  echo "number1 is less than number2."
else
  echo "number1 is not less than number2."
fi
```

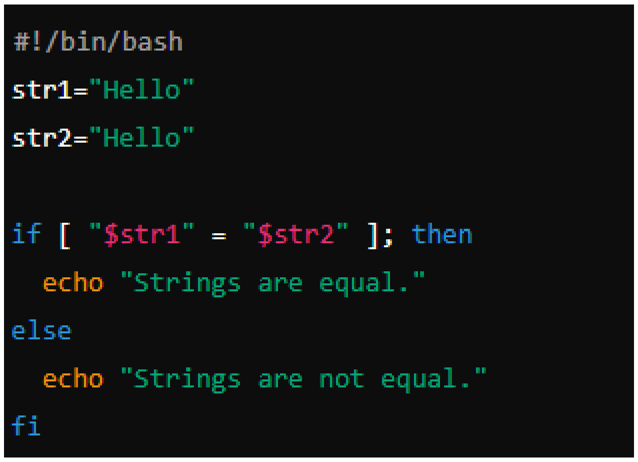
-le (Less than or equal to):



```
#!/bin/bash
number1=10
number2=10

if [ $number1 -le $number2 ]; then
  echo "number1 is less than or equal to number2."
else
  echo "number1 is greater than number2."
fi
```

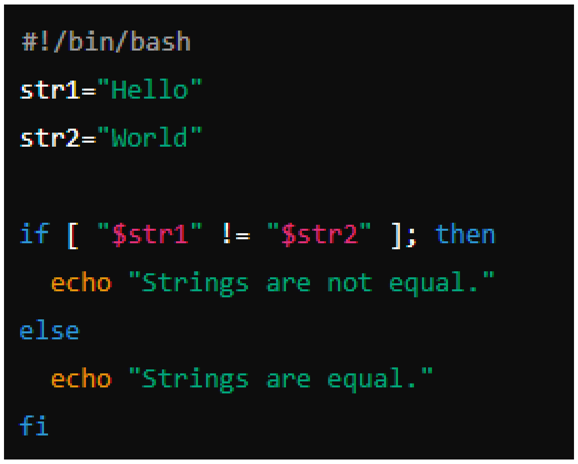
= (String equality):



```
#!/bin/bash
str1="Hello"
str2="Hello"

if [ "$str1" = "$str2" ]; then
  echo "Strings are equal."
else
  echo "Strings are not equal."
fi
```

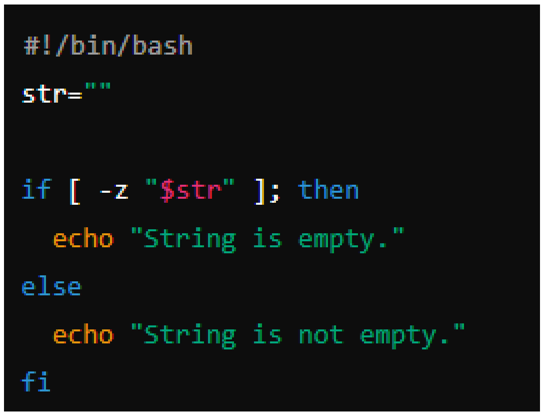
!= (String inequality):



```bash
#!/bin/bash
str1="Hello"
str2="World"

if [ "$str1" != "$str2" ]; then
  echo "Strings are not equal."
else
  echo "Strings are equal."
fi
```

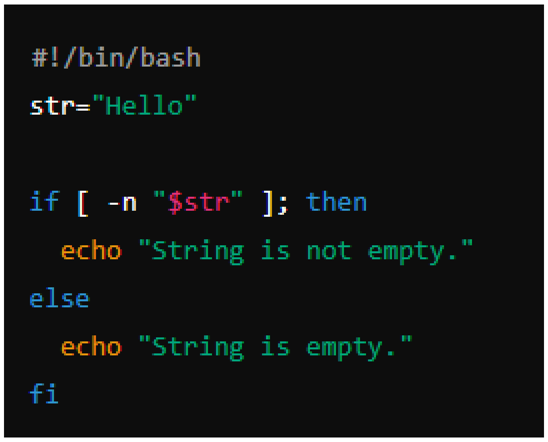
-z string (String is empty):



```bash
#!/bin/bash
str=""

if [ -z "$str" ]; then
  echo "String is empty."
else
  echo "String is not empty."
fi
```

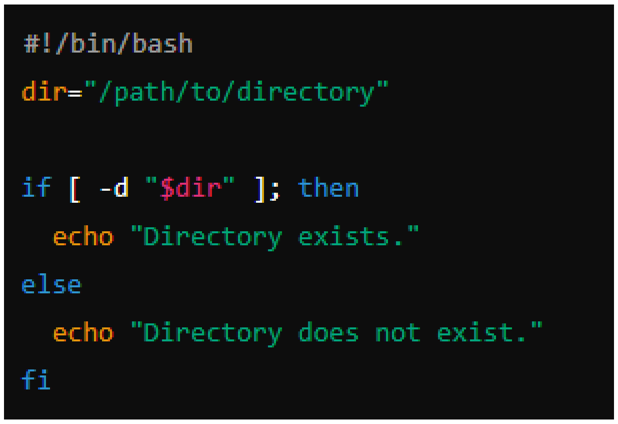
-n string (String is not empty):



```bash
#!/bin/bash
str="Hello"

if [ -n "$str" ]; then
  echo "String is not empty."
else
  echo "String is empty."
fi
```

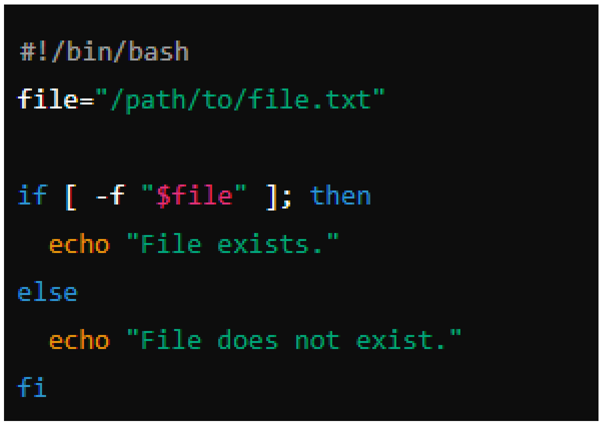
-d (Directory exists):



```
#!/bin/bash
dir="/path/to/directory"

if [ -d "$dir" ]; then
  echo "Directory exists."
else
  echo "Directory does not exist."
fi
```

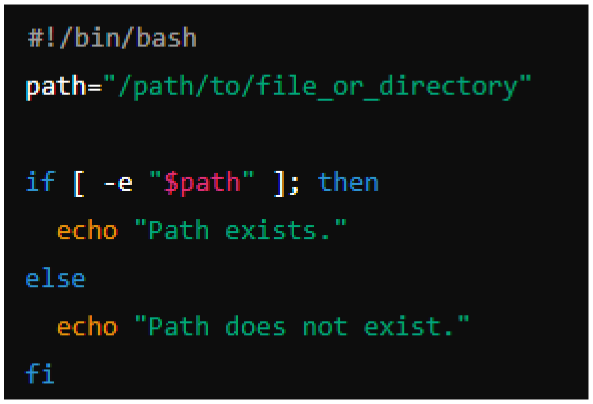
-f (File exists and is a regular file):



```
#!/bin/bash
file="/path/to/file.txt"

if [ -f "$file" ]; then
  echo "File exists."
else
  echo "File does not exist."
fi
```

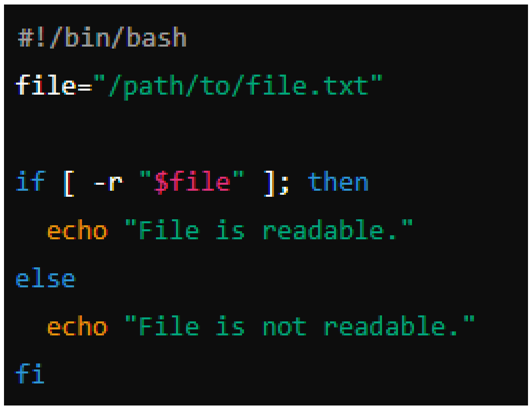
-e (File or directory exists):



```
#!/bin/bash
path="/path/to/file_or_directory"

if [ -e "$path" ]; then
  echo "Path exists."
else
  echo "Path does not exist."
fi
```

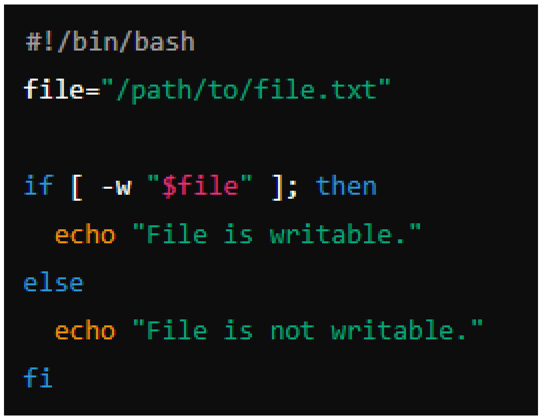
-r (File is readable):



```
#!/bin/bash
file="/path/to/file.txt"

if [ -r "$file" ]; then
  echo "File is readable."
else
  echo "File is not readable."
fi
```

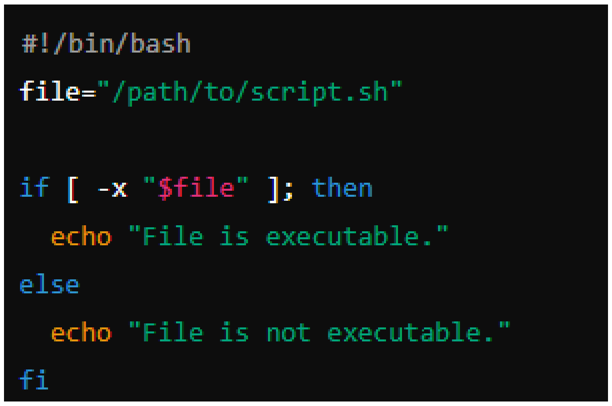
-w (File is writable):



```
#!/bin/bash
file="/path/to/file.txt"

if [ -w "$file" ]; then
  echo "File is writable."
else
  echo "File is not writable."
fi
```

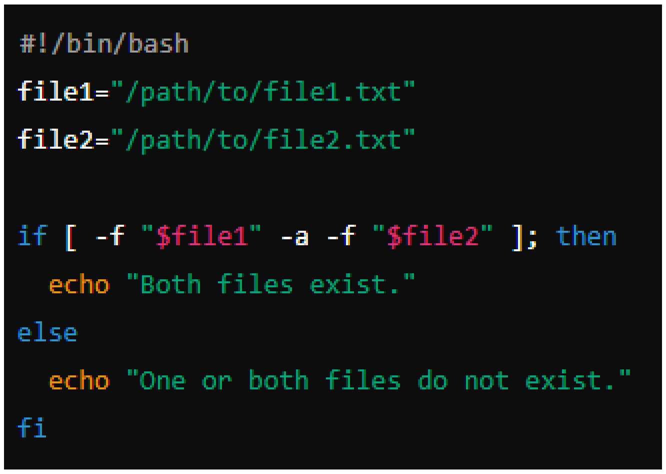
-x (File is executable):



```
#!/bin/bash
file="/path/to/script.sh"

if [ -x "$file" ]; then
  echo "File is executable."
else
  echo "File is not executable."
fi
```

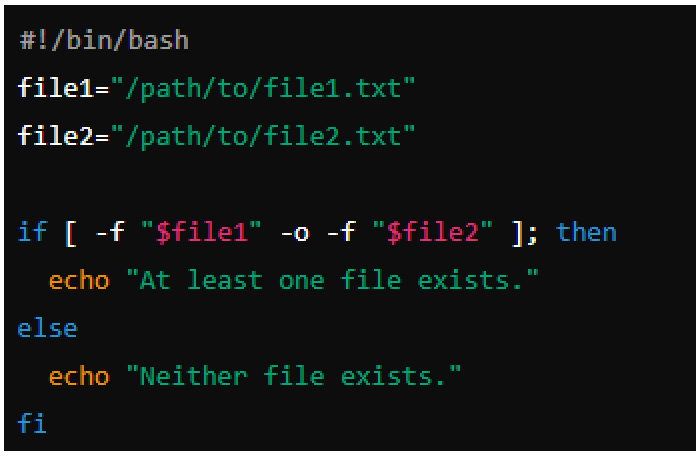
-a (Logical AND):



```bash
#!/bin/bash
file1="/path/to/file1.txt"
file2="/path/to/file2.txt"

if [ -f "$file1" -a -f "$file2" ]; then
  echo "Both files exist."
else
  echo "One or both files do not exist."
fi
```

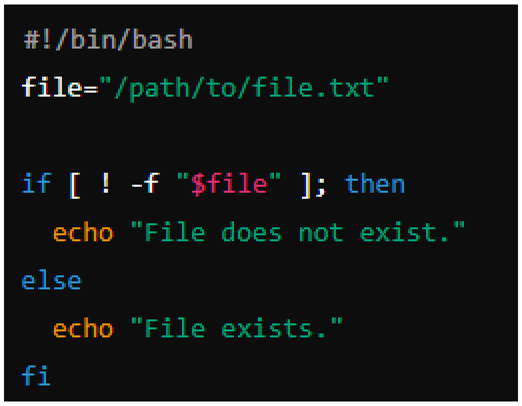
-o (Logical OR):



```bash
#!/bin/bash
file1="/path/to/file1.txt"
file2="/path/to/file2.txt"

if [ -f "$file1" -o -f "$file2" ]; then
  echo "At least one file exists."
else
  echo "Neither file exists."
fi
```

! (Logical NOT):

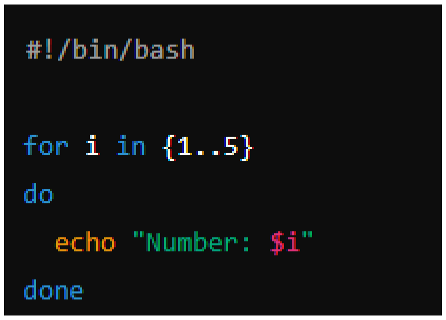


```bash
#!/bin/bash
file="/path/to/file.txt"

if [ ! -f "$file" ]; then
  echo "File does not exist."
else
  echo "File exists."
fi
```

For loop:

Below script prints numbers from 1 to 5.



```bash
#!/bin/bash

for i in {1..5}
do
  echo "Number: $i"
done
```

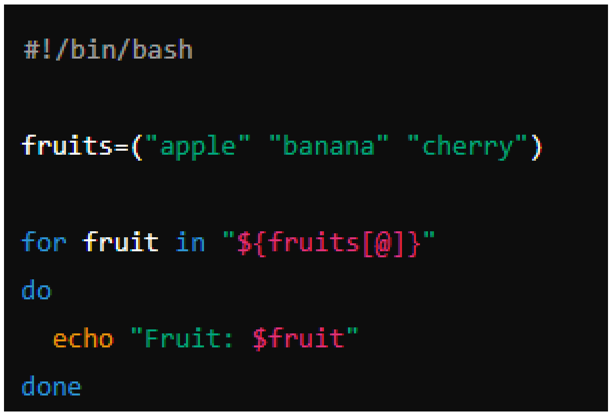
for ((i=1; i<=5; i++))

do

  echo "Iteration $i"

done

Iterating Over a List of Strings.



```bash
#!/bin/bash

fruits=("apple" "banana" "cherry")

for fruit in "${fruits[@]}"
do
  echo "Fruit: $fruit"
done
```

while loop

i=1

while [$i -le 5]

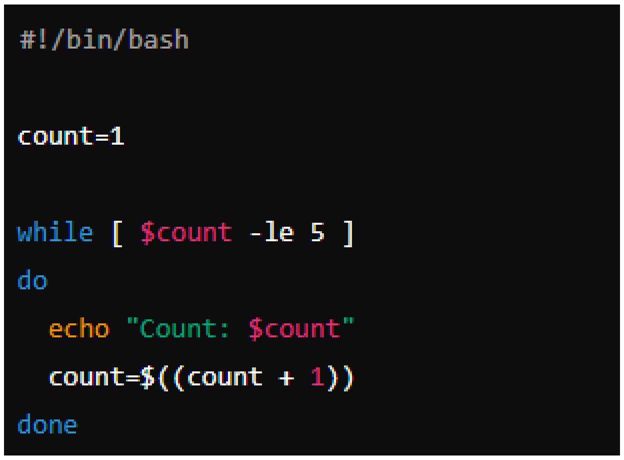
do

  echo "Iteration $i"

  ((i++))

done

While loop:



```
#!/bin/bash

count=1

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

Tasks:

I. Write a script that prints your name, age, and favorite color using the echo command. **[Estimated Time: 4 mins]**
II. Write a script that checks if a given number is positive or negative. **[Estimated Time: 4 mins]**
III. Write a script that checks the value of a variable and prints whether it is "small" (less than 10), "medium" (10 to 20), or "large" (greater than 20). Use if-elif-else. **[Estimated Time: 5 mins]**
IV. Write a script that prints numbers from 1 to 5 using a `for` loop. **[Estimated Time: 4 mins]**
V. Write a script that uses a `while` loop to count from 1 to 5 and prints each number. **[Estimated Time: 4 mins]**
VI. Write a script that calculates the sum of two numbers and prints the result. **[Estimated Time: 4 mins]**
VII. Write a script that checks if a given string is empty or not. **[Estimated Time: 4 mins]**
VIII. Write a script that checks if a file exists and is readable. **[Estimated Time: 4 mins]**
IX. Write a script that checks if a file exists and is writable. **[Estimated Time: 3 mins]**
X. Write a script that checks if a directory does not exist. Use the ! operator. **[Estimated Time: 4 mins]**
XI. Write a script that categorizes a number as "negative", "zero", "small positive" (1-10), or "large positive" (greater than 10). **[Estimated Time: 6 mins]**
XII. Write a script that checks if a file exists and is both writable and executable. Use the -a operator to combine the conditions. **[Estimated Time: 5 mins]**
XIII. Write a script that checks if a file is either readable or writable. **[Estimated Time: 5 mins]**
XIV. Write a script that iterates over numbers from 1 to 10 and prints whether each number is even or odd. **[Estimated Time: 5 mins]**
XV. Write a script that uses a while loop to print numbers from 1 to 10, but stop if the number 5 is reached. **[Estimated Time: 5 mins]**

XVI. Write a script that checks if a directory exists and is writable, or if a file exists and is readable. Use both -a and -o operators. **[Estimated Time: 7 mins]**

XVII. Write a script that uses a for loop to print numbers from 1 to 10 but skips printing the number 7. **[Estimated Time: 5 mins]**

XVIII. Write a script that creates an infinite `while` loop that prints "Running..." until a certain condition is met (e.g., a specific input from the user), then breaks out of the loop. **[Estimated Time: 7 mins]**

Task Submission Guidelines:

1. Make a word document and paste all solutions there and save it as pdf or odt.
2. Include your name and roll no. at the front page.
3. Files other than pdf or odt will not be accepted and will be marked as 0.