=> Shell Scripting :- { Variables, Operators, Functions } { if-else, loops }

=> sha-bang in Linux :-

--> sha-bang is used to specify which shell we should be using to process our script file

--> Writing sha-bang is recommended but not mandatory

```
 #! /bin/bash
```

```
#! /bin/bash

echo "Enter Your First Name"

read FNAME

echo "Enter Your Last Name"

read LNAME

echo "Your Full Name : $FNAME $LNAME "
```

Variable -> To store info/Data (value) ->

-> Variables will represent Data in key-value format

id = 4

name = Roham

age = 18


~~id~~ id = 4)

-> we don't have data types in shell scripting

-> System Variables / Environment variables

-> User Defined Variables

-> Variables which are pre defined and using by our system is referred as System variables

```
$ echo $SHELL
$ echo $USER
 $ echo $PATH
```

--> Variables which are created as per requirement of script --> User Defined Variables

```
    FNAME=ankit    LNAME=Singh  AGE=22
```

To Access and print the data of variables created the following syntax will be used

$ echo $VARIABLE_NAME --> $ echo $FNAME

-->We can access all the environmental variables using ' env ' command

$ export COURSE=DevOpsWithAWS --> Create a variable and set variable using terminal

$ echo $COURSE --> Get the Value/Data from variable

$ unset COURSE --> Unset the value from Variable

How to set variables permanently ?

.bashrc file

We use .bashrc file to set variable data permanently for the user

Every User will have their own .bashrc file

To see all hidden file we can use --> ls -la

$ cat .bashrc --> see the content of .bashrc file

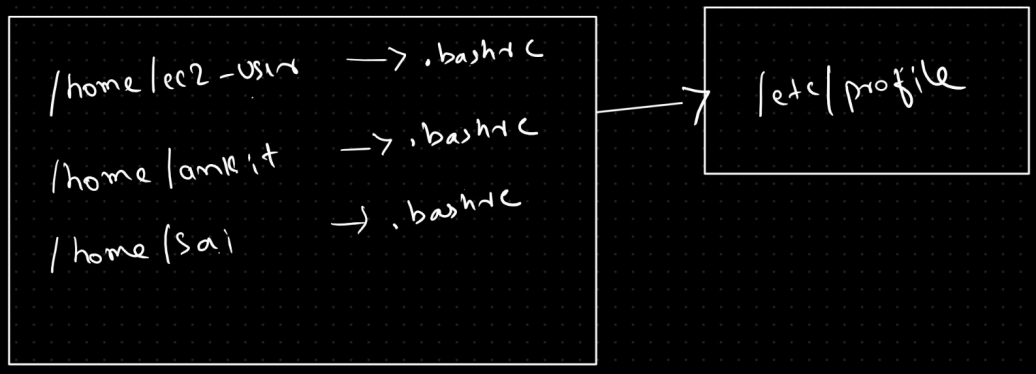
$ vi .bashrc --> open the .bashrc file

--> Add the variables at the end of file

INSTRUCTOR = NAVIN_REDDY
COURSE = GEN_AI

$ source .bashrc --> Will apply changes we have made to .bashrc file

To Acess the data of those variables --> echo $INSTRCTOR
echo $COURSE




--> Setting variables for all users in Linux

$ cat /etc/profile
--> If any variable data has be common for all users of Linux vm then we may have to add those variables /etc/profile

--> Rules to work with Variables

--> Variable name must not start with digits --> 1name --> name1

--> It Should not have special characters --> , @, $, #

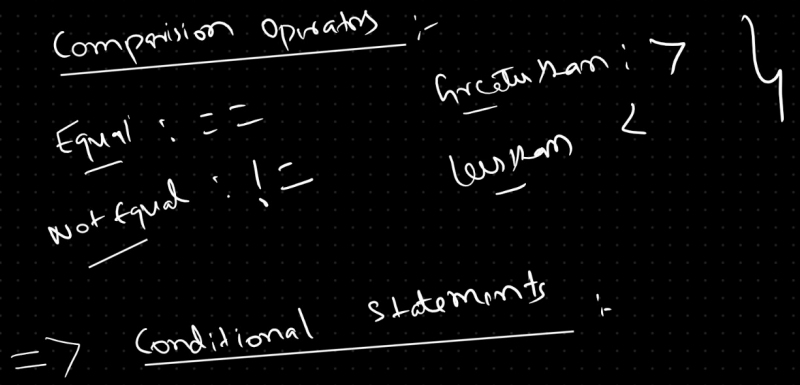
NOTE : Its highly RECOMMENDED to use UPPERCASE for variables name

Operator :
--> Operators are used to perform operations on variables/data

Arithematic Operations

Addition : $ ((num1 +num2 ))
Subtraction : $ ((num1 -num2 ))
Multiplication : $ ((num1 * num2 ))
Division : $ ((num1 / num2 ))
Modulas : $ ((num1 % num2 ))

```
#! /bin/bash

echo "Enter First Number to ADD"

read FNUM

echo "Enter Second Number to ADD"

read SNUM

echo "Result of Addition : $((FNUM+SNUM))"
```



--> Execute commands based on conditions --> Conditional Statements

```
if [ condition ]; then

//statements

else

//statements

fi
```

```
#! /bin/bash

echo "Enter the First Number"

read NUM1

echo "Enter the Second Number"

read NUM2

if [ $NUM1 -eq $NUM2 ]; then

        echo "Numbers Are Equal"

else
        echo "Numbers are Not Equal"
fi


#! /bin/bash

echo "Enter The Number"

read NUM1

if [ $NUM1 -gt 0 ]; then
        echo "Positive Number"

elif [ $NUM1 -lt 0 ]; then
        echo "Negative NUmber"
else
        echo "Number is Zero"

fi
```

Looping statements :-

↳ execute statements multiple times

→ Conditional Based loops (while)

→ Range Based Conditional Based loop (for)

For Loop :-

```
for( (initialization ; Cond" ; modification)
do
    ≡ statements
done
```

```
#! /bin/bash

for(( i=1; i<=6; i++))

do
      echo "$i"

done
```

while loop → execute statements until condition is true

```
#! /bin/bash

echo "Enter the Number"

read NUM

while [ $NUM -le 6 ]

do
      echo "$NUM"

let NUM++

done
```

Infinite loop → loop which runs continously without stopping

```
#! /bin/bash

echo "Enter the Number"

read NUM

while [ $NUM -le 6 ]

do
      echo "$NUM"

done
```

⇒ Script to print Number from 10 to 1

```
Num=10;
while [ $NUM -gt 0 ]
do
    echo "$NUM"
    let NUM--
done
```

Functions / Methods :-

↳ Perform some task / Action / Activity

→ Divide big task into multiple small tasks.

→ Functions are re-usable.

Syntax :-

```
function functionName () {
    ≡ / function body {=}
}

# call function
functionName
```

=

```
function greeting () {
    echo "Hello A!!"
    echo "welcome to Telwin"
    echo "Oops!!"
}
```

//writing function which will read file name from user and print the content of ~~file~~ greeting

```
#! /bin/bash

function doTask(){

    echo "Enter the file name"

    read FILENAME

    cat $FILENAME
}

doTask
```

## function which will read file name from user and if file is present then it will print the content of file if its not present then it will create a new file

```
#! /bin/bash

function fileManager(){

        echo "Enter the file name"

        read FILENAME

        if [ -f "$FILENAME" ]; then
                echo "file is present the the content is hwon below "
                cat $FILENAME

        else
                echo "File is not present with name hence creating new file"
                touch $FILENAME
                echo "File is created"
        fi
}
fileManager
```