Arithmetic Operations in Linux

The following syntax to calculate the **sum** of two integers in a shell script:

Method 1: Using **expr command** with quotes

```
Example1: sum=`expr $num1 + $num2`

Example2: sum=$(expr $num1 + $num2)
```

Method 2: Using **directly** sum with the shell

Example: sum=$(($sum1+$sum2))

```
Example using shell script

#! /bin/bash

a=10 b=5
```

```
sum=$(($a + $b))  sub=$(($a - $b))

mul=$(($a * $b))      or  mul=$(($a \* $b))

div= $(($a / $b))  rem= $(($a % $b))

echo "sum is $sum sub is $sub mul is $mul div is $div rem is $rem"
```

Method3 :work during **Run Time**, Input numbers and store result in a variable

Example :

```
read -p  "first no.  is"  no1          or    read -n  "first no. is "
read -p  "second no. is "  no2         or    read -n  "second no. is "
Sum =  $($no1+$no2)
 Echo "sum is:  $sum"
```

Loops Scripting in Linux

Most languages have the concept of loops, **for** and **while** loops are used in the Bourne shell.

For Loops : **for loops iterate through a set of values until the list is exhausted:**

Example1 : file name is for.sh

```
#!/bin/bash
for i in 1 2 3 4 5
do
  echo "Loop ... number is $i"
done

_____________________
Result :
Loop .... number 1
Loop .... number 2
Loop .... number 3
Loop .... number 4
Loop .... number 5
```

Example2 : file name for2.sh

```
#!/bin/sh
for i in first 1 * 2 last
do
  echo "Looping ... i is set to $i"
done
```

Homework:Try the code without the `*` , then it again with the `*` . Try it with the `*` surrounded by double quotes, and try it preceded by a backslash (`\*`)

While Loops

Example :file name is while.sh

```
#!/bin/sh
input_string=begin
while [ "$input_string" != "end" ]
```

```
do
  echo " type (end to quit)"
  read INPUT_STRING
  echo "print the: $input_string"
done
```

In this example the echo and read statements will run indefinitely until you type "end" when prompted.

If .. then statement in Bash : *Example*

```
if [ $a == $b ]                    or   if [ $a != $b ]
hen
  echo "a is equal to b"
fi
```

Case statement in Bash : Example :

```
echo "Which color do you like?"
echo "1 - Blue"
echo "2 - Red"
read color;
case $color in
1) echo "Blue is a primary color.";;
2) echo "Red is a primary color.";;
*) echo "This color is not available;;
 esac
```

The += Operator in Bash

The '+=' operator to concate two variables, strings

Var1="Hello" or var1=100 Var2=150 or var1=begin , var2=end

Var2+=" World" or var3=$var1$var2 or var4=" ${var1} ${var2} "

echo $Var1 or echo $Var3 or echo $var4

Result : Hello World **Result** : 100150 **Result** : begin end