Bash Script

1. **Print**

```
echo Gautam
```

It simply prints Gautam

2. **Read Value From User**

```
read varName
echo "Hello $varName"
```

It takes input from user and store into **varName** then ‘echo’ will print **“Hello ‘user input’”**

3. Loops

I. For Loop: A for loop is used to iterate over a list of items. Here is an example:

```
#!/bin/bash

for i in 1 2 3 4 5
do
    echo "Number: $i"
done
```

In this example, the for loop iterates through the list of numbers from 1 to 5 and prints each number to the console.

II. While Loop: While loop is used to perform a specific action while a condition is true. Here is an example:

```
#!/bin/bash

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

In this example, the while loop performs the action of printing the current value of the counter variable to the console while the condition $counter -le 5 (which means "counter is less than or equal to 5") is true.

III. Until Loop: An until loop is similar to a while loop, but it performs a specific action until a condition is true. Here is an example:

```
#!/bin/bash

counter=1
until [ $counter -gt 5 ]
do
 echo "Number: $counter"
 ((counter++))
done
```

In this example, the until loop performs the action of printing the current value of the counter variable to the console until the condition $counter -gt 5 (which means "counter is greater than 5") is true.

4. Arithmetic Operation

In Bash, you can perform arithmetic operations using the $((...)) syntax or the ((...)) syntax, and store the result in a variable. Here's an example:

```
#!/bin/bash

# Using $((...)) syntax
read a
read b
c=$((a + b))     # Addition
d=$((a - b))     # Subtraction
e=$((a * b))     # Multiplication
f=$((a / b))     # Division (integer division)
g=$((a % b))     # Modulo

echo "c = $c"
echo "d = $d"
echo "e = $e"
echo "f = $f"
echo "g = $g"

# Using ((...)) syntax
((a++))          # Increment
((b--))          # Decrement
((c += 10))      # Addition assignment
((d -= 5))       # Subtraction assignment
((e *= 2))       # Multiplication assignment
((f /= 2))       # Division assignment
((g %= 2))       # Modulo assignment
```

```
echo "a = $a"
echo "b = $b"
echo "c = $c"
echo "d = $d"
echo "e = $e"
echo "f = $f"
echo "g = $g"
```

In this example, the arithmetic operations are performed using both $((...)) syntax and ((...)) syntax. The result of each operation is stored in a variable, which can then be used later in the script. The echo command is used to display the value of each variable.
Note that in Bash, all variables are treated as strings by default, so you need to use the $((...)) or ((...)) syntax to perform arithmetic operations.

5. Logical Operator & Condition

In Bash, you can use the if...else statement and logical operators to make decisions based on conditions. Here's an example:

```
#!/bin/bash

a=5
b=3

# Using if...else statement with logical operators
if [[ $a -gt $b && $a -ne 0 ]]; then
    echo "$a is greater than $b and not equal to zero"
else
    echo "$a is not greater than $b or equal to zero"
fi

# Using if...elif...else statement with logical operators
if [[ $a -gt $b ]]; then
    echo "$a is greater than $b"
elif [[ $a -eq $b ]]; then
    echo "$a is equal to $b"
else
    echo "$a is less than $b"
fi
```

In this example, the if...else statement is used to test whether the value of $a is greater than the value of $b and not equal to zero. The && logical operator is used to combine two conditions. If the condition is true, the echo command will display a message indicating that

$a is greater than $b and not equal to zero. If the condition is false, the else block will execute and display a different message.
The if...elif...else statement is also demonstrated in this example. It is used to test whether the value of $a is greater than, equal to, or less than the value of $b. The -gt and -eq operators are used to test for greater than and equal to, respectively. If none of the conditions are true, the else block will execute and display a message indicating that $a is less than $b.
Note that the [[...]] syntax is used to enclose the conditions in the if statements, and logical operators are used to combine conditions. Logical operators available in Bash include && (logical AND), || (logical OR), and ! (logical NOT).

6. **Declare Array and Read value**

```
#!/bin/bash

# Prompt the user to enter the number of integers
echo "Enter the number of integers:"
read n

# Declare an array to store the integers
declare -a nums

# Take n integers from the user and store them in the array
echo "Enter $n integers:"
for (( i=0; i<$n; i++ ))
do
    read nums[$i]
done

# Calculate the sum of the integers
sum=0
for (( i=0; i<$n; i++ ))
do
    sum=$((sum + ${nums[$i]}))
done

# Calculate the average of the integers
avg=$((sum / n))

# Print the average
echo "The average of the $n integers is $avg"
```

In this script, we use the declare command to create an array called nums. We then prompt the user to enter the number of integers they want to input and use a for loop to read the integers from the user and store them in the array.

We then use another for loop to iterate over the elements of the array and calculate their sum. Finally, we calculate the average by dividing the sum by the number of integers and print it out.