```
: << "que1"
#Create a variable name with your name and print:
echo "Enter your name:"
read name
echo "Hello, my name is: $name"
que1

: << "que2"
#Assign two numbers to variables and print their sum, difference, product, and quotient.
echo "Enter the first number:"
read num1
echo "Enter the second number:"
read num2
echo "The sum of the both number is $((num1+num2))\nThe difference b/w the numbers is $((num1-num2))\nThe product of the
numbers is $((num1*num2))\nThe quetient of the both no. is $((num1/num2))"
que2

: << "que3"
#difference b/w single and double quote
name="Tirupati"
echo "hello $name"
echo 'hello $name'
que3


: << "que4"
#Ask the user for a number and compare the no. with 100:
read -t 5 -p "Enter a number in 5 seconds: " num
if [ $num -lt 100 ]; then                       #notice here the space around "[" and "]" i.e. compulsory
    printf "Your number is less than 100\n"
elif [ $num -eq 100 ]; then                     #notice here the space around "[" and "]" i.e. compulsory
    printf "Your number is equalto 100\n"
else
    printf "Your number is greater than 100\n"
fi              #End of the if/else block
que4


: << "que5"
#Take a filename as input and check: is exists, is directory → print "It’s a directory"
read -p "Enter the file name: " path
if [ -d "${path}" ]; then       #"${path}" is used for path with spaces"
    printf "It's a directory"
elif [ -e "${path}" ]; then
    printf "Yes, file exists!"
else
    printf "The file doesn't exits"
fi
que5

: << "que6"
#Print numbers from 1 to 10 using: for loop, while loop
echo "printing 1-5 using for loop"
for i in 1 2 3 4 5      #"{1..5}" can be used for range
do                  #starting of the loop block
    echo "$i"
done                #ending of the loop block

echo "printing 1-5 using while loop"
n=1
while [ $n -lt 6 ]
do
    printf "$n, "
    n=$((n+1))
done
printf "\n"         #for line break in the output
echo "All done"
que6


: << "que7"
#Write a script to calculate the factorial of a number using a while loop.
read -p "Enter a number to calculate the factorial: " num
if [ $num -lt 0 ]; then             #dont forget the space around [ and ] and use of "then" after if condition
    echo "Please enter a positive integer"
else
    fact=1
    count=1
    while [ $count -le $num ]
        do
        fact=$((fact*count))
        count=$((count+1))
        done
echo "The factorial of $num is $fact"
fi
que7
```

```
: << "que8"
#Print all .txt files in the current directory using a loop.
read -p "Enter any file extension to list those files here: " ext
file_array=(*.$ext)                  #if no .$ext file found then the array will contain the literal "*.$ext" as its only
element
if [ ${#file_array[@]} -eq 1 ]       #checking if array is empty
then
    echo "No any .$ext file found in the current directory i.e \"$(pwd)\""
else
    echo "All .$ext files are listed below with the help of loop"
    count=1
    for i in ${file_array[@]}
    do
        echo "$count. $i"
        count=$((count+1))
    done
fi
que8



: << "que9"
#Use ! to check: If true is negated → print "false"
if  ! false     #here [ ! false ] will not work, because [ ] is used for test command
then
    echo "false is negated as true"
fi
que9



: << "que10"
#Create a calculator script that: Takes two numbers and an operator ( + , - , * , / ) Prints the result
read -p "Enter first number: " num1
read -p "Enter second number: " num2
read -p "Enter the operator: " opt
echo "$num1$opt$num2 = $((num1 $opt num2))"

echo "With another method"
read -p "Enter any expression (e.g. 2 * 10): " -a exp   # -a is used to store each space separated value in an array as its
element
echo "${exp[@]} = $((${exp[@]}))"   #here $((${exp[@]})) is similar as $((exp[0] exp[1] exp[2])) i.e. array elements are
used as variables
que10


#: << "que11"
#Make a script that keeps asking the user for input until they type "exit" .
while true
do
    read -p "Enter your name: " name
    if [ "$name" == "exit" ]
    then
        echo "Exiting the script"
        break
    else
        echo "Hello, $name"
    fi
done
#que11

: << "important Notes"
########## Important flags ###########
-p == one line prompt, like [read -p "Enter your name: " name]
-t == timeout in seconds, like [read -t 5 -p "Enter your name" name]
-n == no. of characters, like [read -n 1 -p "Enter a character: " char]
-s == silent, like [read -s -p "Enter your password: " pass] (no output shown on screen)
-sp == silent with prompt, like [read -s -p "Enter your password: " pass] (no output shown on screen)
#Note: the flags are key value pairs, notice the flags values just after the flags


########## Ending of code blocks ###########
fi == ends if block
done == ends loop block
esac == ends case block
} == ends function block
Block ending in Bash


########## Special/researved Variables ###########
$0 → script name
$1 , $2 , ... → arguments passed to script
$# → number of arguments
$$ → process ID of script
$? → exit status of last command
```

```
########## File test operators ###########
[ -e file.txt ] # exists
[ -f file.txt ] # regular file
[ -d dir ]      # directory
[ -r file.txt ] # readable
[ -w file.txt ] # writable
[ -x file.txt ] # executable


########## File test operators ###########
range(1,6) in python == {1..5} in bash

########## Loop control statements ###########
break        == exit loop
continue     == skip to next iteration of loop
exit         == exit script
:            == no-op (does nothing), similar to "pass" in python
important Notes
```