SHELL SCRIPTING:

*Any file where you put a series of linux commands, that can be called as Shell Script.
*Shell script is used to automate the manual and repetitive tasks.

Shebang:
#!/bin/bash -- we call it as shebang.
It invokes bash shell. If we don't write this line, it uses default shell.

To check the types of shells available:
cat /etc/shells

Example:
1.Sample.sh

```
#!/bin/bash
echo "Hi, this is Raj"
ls -lt
pwd
df -h
date
uname -a
```

To execute this,
./Sample.sh

How do you read arguments inside the shell script?
$1 - 1st argument
$2 - 2nd argument
$3 - 3rd argument
.
.
.
.
.
$n

Example:
1.Variables.sh

_____
#!/bin/bash

a=`pwd`
b=`ls -lrt`
c=`echo "I am from India"`

echo $a
echo $b
echo $c

_____

2.Pass_arguments.sh

_____
#!/bin/bash

echo "this is $1"

echo "I am from $2"

_____
To execute this,
./Pass_arguments.sh raj India

_____

3.read_name.sh

_____
#!/bin/bash

echo "enter you name"
read name
echo hello $name

_____

Special Characters:
These have special meaning to a shell. They are given below.

$0 -- name of the file itself
$# -- the total number of arguments passed to a script
$* -- All arguments passed to a script
$@ -- All arguments passed to a script stored array format
$$ -- process id of the current running process

$! -- it is a process id of the last command went into a background
$? -- status of the last executed command. Zero(0) means success & non-zero means failure.

Conditoinal statement:
IF statement:
Syntax:

```
if [ condition ]
then
    statement
fi
```

if-else statement:
Syntax:

```
if [ condition ]
then
   statement1
else
   statement2
fi
```

if-else-if:
Syntax:

```
if [ condition1 ]
then
   statement1
elif [ condition2 ]
then
   statement2
elif [ condition3 ]
then
   statement3
else
   statment4
```

```
fi
```

Examples:
1.check5.sh

```
#!/bin/bash

if [ $1 -eq 5 ]; then
        echo "$1 is five"
else
        echo "$1 is not five"
fi
```

2.big_2no.sh

```
#!/bin/bash

if [ $1 -gt $2 ]; then
        echo "$1 is big"
else
        echo "$2 is big"
fi
```

Restict the script to pass only 2 numbers:

```
#!/bin/bash

if [ $# -ne 2 ]; then
        echo "pass only 2 numbers"
exit 1
fi

if [ $1 -gt $2 ]; then
        echo "$1 is big"
else
        echo "$2 is big"
fi
```

WHILE LOOP: It is used to run a set of commands repeatedly until some ondition occurs.

Syntax:

```
while [ condition ]
do
  statement
done
```

Eg:
1.Print the numbers from 1 to 10.

```
#!/bin/bash
num=1
while [ $num -le 10 ]
do
        echo $num
        num=`expr $num + 1`
done
```

2.Write a script to find factorial of a given number.

```
#!/bin/bash

num=$1
fact=1

while [ $num -gt 0 ]
do
  fact=`expr $fact \* $num`
  num=`expr $num - 1`
done
echo "fact of $1 is $fact"
```

WHILE READ LINE: It is used to read each line in a file

syntax:

```
while read line
do
  echo $line
done < file
```

Eg:
1.Count no of words in each line of a file

```
#!/bin/bash

file=$1
count=1

while read line
do
  words=`echo $line | wc -w`
  echo "$count:$line"
  count=`expr $count + 1`
done < $file
```

FOR LOOP:

syntax:

```
for i in value1 value2 value3 value4
do
statement
done
```

Eg:
1. Print numbers from 1 to 10

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

```
for num in 1 2 3 4 5 6 7 8 9 10
do
	echo $num
done
```

_____

OR

_____

```
#!/bin/bash

for num in {1..10}
do
	echo $num
done
```

_____

2.Print numbers from 1 to 10 incrementing by 2

_____

```
#!/bin/bash

for num in {1..10..2}	--- initiate with 1 and increment by 2 and print upto 10
do
	echo $num
done
```

_____

3.Print the fruits names apple, banana, orange and mango.

_____

```
#!/bin/bash
for fruit in apple banana mango orange
do
	echo $fruit
done
```

_____

---

echo $? -- prints the status of last executed command (zero means success and non-zero means failure)

ls file_name -- print the file_name if it is present.

ls -l file_name -- print the file_name with details if it is present.
ls /home/ubuntu/directory -- lists the files and directories of some other directory from the current directory.

1.Write a shell script to identify whether the given file exist or not in a given directory.

```
#!/bin/bash

ls -l /home/ubuntu/given_dir_name/given_file_name

if [ $? -eq 0 ]; then
        echo "file exists"
else
        echo "file does not exist"
fi
```

Here if file does not exist, then it prints "file does not exist" along with the error message.
If we don't want to see error message, then we run below commands

ls -l 2> filename   -- it stores error message in a file
ls -l &> filename   -- it stores everything i.e both output and error message in a file
ls -l &> /dev/null   -- it sends both output and errors to bin i.e they will be stored nowhere.

```
#!/bin/bash

ls -l /home/ubuntu/given_dir_name/given_file_name &> /dev/null

if [ $? -eq 0 ]; then
        echo "file exists"
else
        echo "file does not exist"
fi
```

---

2. Write a shell script to retain recent 30 files in a perticular directory and delete other files.

```
#!/bin/bash
total=`ls -l | wc -l`
remaining=`expr $total - 30`

if [ $remaining -gt 0 ]; then
        ls -rt | head -$remaining | xargs rm -rf
        echo "Recent 30 files are retained and others files are deleted"
else
        echo "there are no more than 30 files"
fi
```
