What is shell?

Bash shell is a command language interpreter that executes commands.

Some Available Shells	
Sh	Shell, lacks interactivity
Csh	C-shell
Ksh	Korn-shell
Bash	Bourne again shell

Opening A Terminal	
Search for Terminal in Dash Or CTRL + ALT + T	
$	User
#	Root

Printing On Terminal	
echo	Print command
echo “Hi”	Prints Hi
echo `date`	Prints the current date Where ` ` (back-tick) refers to built-in-command
echo $SHELL	Prints the current shell in use
clear	Clears the screen

Bang Line
Every shell script starts with this line, indicating the location of the shell
#!/bin/bash

Sample Shell Script	
#!/bin/bash *# Name of this file is "test.sh"* *echo " Hello!!! How are you?"*	
Executing Script	
chmod +x test.sh	Change the script to an executable
./test.sh	Execute the script

Variables	
User Variables	Defined by the user.
System Variables	Built in variable.

User Variables

```
v1 = 100
echo 'v1'           #prints v1
echo '$v1'              #prints $v1
echo "$v1"              #prints 100
```

System Variables

```
echo $BASH_VERSION #prints version of the current bash installed

echo $HOME          #prints path of home. echo $HOME = echo ~

echo $USER              #prints name of the user e.g. ankit

echo $HOSTNAME #prints hostsname e.g. xenial
```

Manual Of Every Command	
man	Displays manual of the argument. e.g. *man echo*

Present Working Directory	
pwd	Displays present working directory
pwd –L	Logical address of the pwd
pwd –P	Physical address of the pwd

Listing Content of Directory	
ls	List directory contents
ls –r	List content in reverse order.
ls -l	Long list of contents.
ls -a	Lists all contents.
ls -h	Human readable.
ls –all	All content + long list.
ls –R	Recursive call entill the end to nesting.
ls –F	List with type: /=directory *=executable @=shortcut.

Changing Directory	
cd	change directory to login directory (generally $HOME)
cd ~	cd to $HOME
cd ..	One level up
cd /~user	Login directory of user
cd ../../	Two level up

File Creation	
touch	
touch abc.xyz	Creates a file abc.xyz
gedit	
gedit abc.xyz	Opens gedit and once you save it creates a file abc.xyz with the written contents.
cat	
cat abc.xyz	Prints the content of specified file.

Redirection	
cat > abc.xyz	Creates a file abc.xyz or overwrites as an empty file if the file already exists.
cat abc.xyz >> def.uvw	Appends the content of abc.xyz at the end of def.uvw
cat abc.xyz def.uvw > ghi.rst	Creates a file ghi.rst and adds the content of abc.xyz and def.uvw to ghi.rst.
CTRL+D	Save the file with EOF (end of file) character.

File Permission	
Category	
Owner | Group | Other	
Permissions	
Read | Write | Execute	
Granting All Permissions	
Owner	-rwx------
Group	----rwx---
Other	-------rwx
Note: First - stands for directory status	
Values	
Read	4
Write	2
Execute	1
Granting Few Permissions	
r--	4
rw-	6
rwx	7
chmod Command	
chmod u=rwx,g=rwx,o=rwx abc.xyz	
chmod u=7,r=7,o=7 abc.xyz	
chmod 777 abc.xyz	

Creating Directories	
mkdir	Creates an empty directory
mkdir dirname	Creates an empty directory dirname
Creating Nested Directories	
mkdir --parent a/b/c or *mkdir -p a/b/c*	Creates directory a containing a directory b, which contains a directory c
. and .. Directory	
.	Current Directory
..	Parent Directory
mkdir ./{d,e}	Creates child directory d, e in current directory
mkdir --parent a/b/c/{d,e} ?	

Removing Directories	
rmdir	Removes the directory(ies), only if they are empty
Removing Nested Directories	
rmdir -p a/b/c or *rmdir a/b/c a/b a*	Starts removing from the child node
Try with "-v" switch (v= verbose, which outputs info at terminal)	
What If Directories Are Not Empty?	
rmdir -p x/y/z	Only works if directory(s) are empty i.e., containing nothing but directories. Otherwise, it removes all child-nodes till the directory containing any other file(s)

Removing Files/Directories	
rm	Remove file/directory, by default only files are deleted not the directories
rm -r a	Remove directories and their contents recursively

Coping Files/Directories	
cp	Copies files and directories
cp f1 f2	1. f2 doesn't exists => copy 2. f2 exists => overwrites
cp f1 dir1	copies f1 to dir1, overwrites if f1 exists in dir1
cp f1 f2 -i dir2	copies f1 f2 to dir2, interactive, ask before overwriting
cp -r dir1 dir3	1. dir3 doesn't exists => creates dir3 & copies dir1's content in dir3 2. dir3 exists => copies dir1 and it's content in dir3.

Moving/Renaming Files	
mv	Moves (renames) files
mv f1 f2	1. f2 doesn't exists => renames f1 to f2 2. f2 exists => overwrites f2, removes f1.
mv f1 f2 dir1	moves f1 f2 in dir1 and if any of them exists in dir1 replaces the files in destination

Expressions [expr]

```
#!/bin/shell

number=2

echo $number

if [ $number -gt 0 ]

then

echo "Positive Number"

else

echo " Negative Number"
fi
```

Expressions [expr]

```
#!/bin/shell

read number

echo $number

if [ $number -gt 0 ]

then

echo "Positive Number"

else

echo " Negative Number"
fi
```

Expressions [expr] or test command For Mathematics				
Math Operator in Shell Script	Meaning	Mathematical Statements	But in Shell	
			For **test** statement with if	For **[expr]** statement with if
-eq	is equal to	5 == 6	if test 5 -eq 6	if [5 -eq 6]
-ne	is not equal to	5 != 6	if test 5 -ne 6	if [5 -ne 6]
-lt	is less than	5 < 6	if test 5 -lt 6	if [5 -lt 6]
-le	is less than or equal to	5 <= 6	if test 5 -le 6	if [5 -le 6]
-gt	is greater than	5 > 6	if test 5 -gt 6	if [5 -gt 6]
-ge	is greater than or equal to	5 >= 6	if test 5 -ge 6	if [5 -ge 6]

Expressions [expr] or test command For Mathematics	
Operator	**Meaning**
! expression if ! [$number -gt 0]	Logical NOT

if [expression1 -a expression2]
 	
if [$var -gt 0 -a $var -lt 50]
 	
if [$var -gt 0] && [$var -lt 50]
	Logical AND

if [expression1 -o expression2]
 	
if [$var -gt 10 -o $var -lt 5]
 	
if [$var -gt 10] || [$var -lt 5]
	Logical OR

Expressions [expr] or test command For Mathematics	
Operator	**Meaning**

case word **in**
  pattern1)
    Statement(s) to be executed if pattern1 matches
    ;;
  pattern2)
    Statement(s) to be executed if pattern2 matches
    ;;
  pattern3)
    Statement(s) to be executed if pattern3 matches
    ;;
esac
 	
* **=> is default pattern**	