

# Lets Write your first python Script

Step 1 → Understand Variables  
& Data types

Step 2 → Do some magic with them

Step 3 → Run it.

---

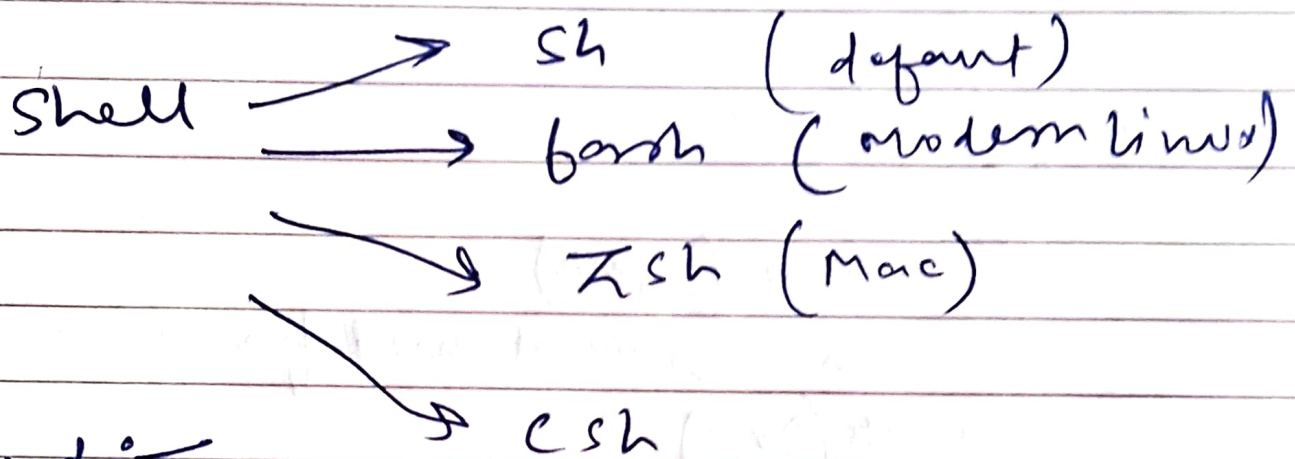
## shell Scripting.

- ① Basic script & giving permission.
- ② Variable, Constant, User inputs
- ③ if condition [ ] fi,
- ④ package installer scripts
- ⑤ Loops
- ⑥ Arguments, functions
- ⑦ Error handling
- ⑧ awk, sed, grep, find
- ⑨ server bkp & dump project,

# Shell Scripting for Devops.

• sh files - Collection of shell commands.

chmod +x executable.



Command :

>> printenv (print all environment Variables)

★ Variable → \$ Lagter hai

# if condition - if this or that.

if [ else, fi  
if [ "/etc/passwd" ] ;

if [ conditions ] ; then  
if commands ; then

if [ ]  
else [ ]  
elif [ ]  
fi  
✓

loops :-

① for loop :-

```
for i in {1...5}
do
```

```
echo $i
done
```

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

<< Comments. (Starts)

This is comment used for upcoming pgm

Comments (Over)

```
for i in {1...5}
```

```
do
```

```
read -p "Enter the user name" user_name
```

```
sudo useradd -m $user_name
```

```
echo "User $user_name added suc"
```

```
done.
```

```
# mkdir -p day{1..90}
rm -r day{1..90}
```

```
→ for i in {1..90}
do
```

```
    mkdir -p day$i
```

```
done.
```

define krdo kha le suru kma hon

```
→ for (( i=1 ; i<=10 ; i++ ))
do
```

```
    echo "$i"
```

```
    echo "hello"
```

```
done.
```

kha tak  
jaega

increase  
hoga  
ya  
decrease.

## ★ Argument

Terminal

```
$> mkdir;
```

↳ argument

## Real time Use

```
{ echo "install $1"
  sudo apt update &&
  sudo apt install $1 -y
  echo "Succ install $1"
```

Ex: \$> ./filename.sh ankit ankit

\$0 \$1 \$2

Note \$@

\$#

→ print all arguments,

→ total number of arguments

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

```
if [ $# -eq 0 ]
```

```
then
```

```
echo "please file pass a file as argument"
```

```
echo " Usage : ./if_file_exist.sh <filepath>
```

```
exit 1
```

```
fi
```

```
if [ -f $1 ]
```

```
then
```

```
echo "file exists"
```

```
else
```

```
echo "file not exist"
```

```
fi
```

① Functions : [reusable] nota hai

```
function_name() {  
    echo "haldi lagao"  
    echo "pani ne dalo"  
}
```

} format.

for calling > haldi # function call  
                    └─ function name.

① Example :

#! /bin/bash

<< usage

• /function.sh hello → argument (main)

inside function call

install\_package docker.io → argument (local)

usage

echo "\$1" is the main argument pass to script

# function define

```
install_package() {
```

```
    echo "$1" is local argument
```

```
    sudo apt install $1
```

```
}
```

install\_package docker.io.

- function banne ke rakh do and use krw while you call your function
- Argument pass krna do function me usko local argument bolenge.

## ★ Create User.

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

```
<<Cmt
```

- take user name as input
- take password as input
- create the user

```
Cmt
```

```
read -p " Username" Username
```

```
read -p " password" password
```

```
if id "$Username" & /dev/null; then
    echo "The user $Username exists"
```

```
exit 1
```

```
else
```

```
    echo "the user $Username will be created"
```

```
fi
```

```
sudo useradd -m $Username -p $password
```

```
echo "User $Username added successfully"
```

`#!/bin/bash.`

`source ./"_file name"`

`create_user` } → function

## ★ Error handling :-

Example :-

`mkdir josh`

`echo "do prod work"`

Output

Error : - - - - -

do prod work.

→ `#!/bin/bash`

`set -e`

shell script  
Kya Krta hai  
ki agar error aya  
to next line ko execute  
Krta hai error ko  
ignore Krta hai.  
that's why.  
error handling required

`<try> || <catch>`

OR,  
OR,

Note : Every error to be handled differently.  
[mkdir, not mounted, disk]

`<<error deted>> || { fallback logic to handle error }`