

# Linux Tasks

## Task # 1) Create a new directory

### Syntax:

**mkdir directory\_name**

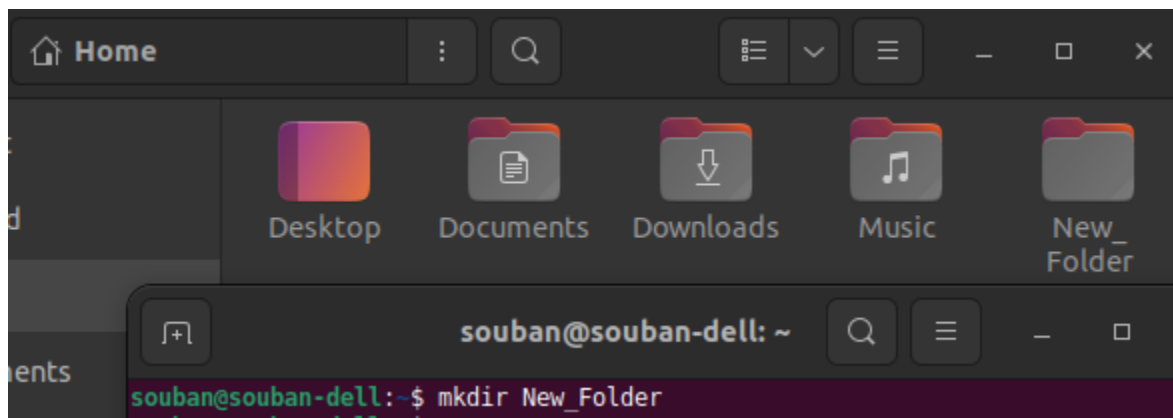
/create directory in the present path

**mkdir ~/path/directory\_name**

/create directory in any path from anywhere

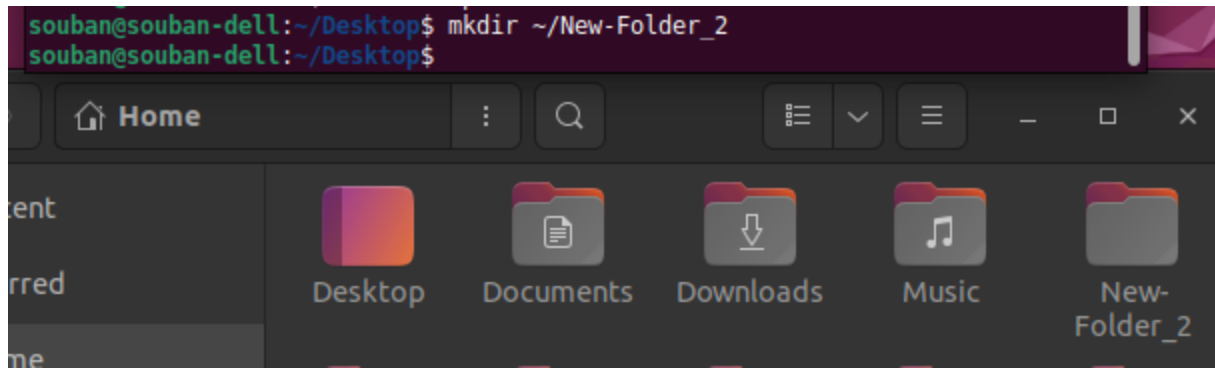
- a. Creating a new directory in Home '**New\_Folder**'

**Explanation:** for creating a new directory/folder you use mkdir (make directory) command. It will create the directory in the path you are present in. Right now the directory is created in the home directory. If we want to create another directory in it, we will first go to another path and use this command again. However there are other ways as well to do this.



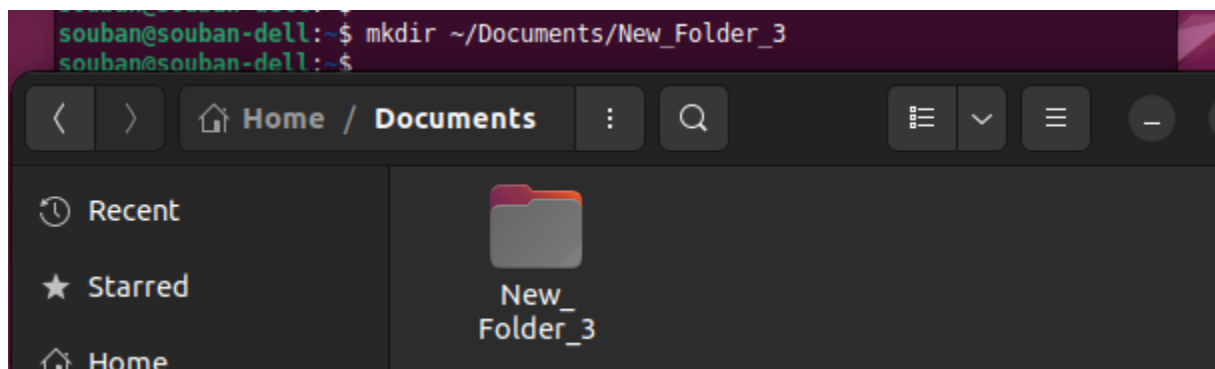
- b. Creating a new directory '**New\_Folder\_2**' in **Home** being in the **Desktop**.

**Explanation:** Now I have changed my directory/path to **Desktop** but I want to create a new directory to home. I will use the command **mkdir ~/New-Folder\_2** and it will create the new directory/folder at home.



- c. Creating a new directory '**New\_Folder\_3**' in **Documents Folder** being in the **Desktop**.

**Explanation:** We can also create a new directory inside another directory while being on some other directory. For example, we can create **New\_Folder\_3** inside the **Documents** folder while being in the home directory. The main syntax this depends on the command **mkdir** while using the sign **tilde** `~` and mentioning the path using **slashes** `/`.



**Note:** This was done in an explanatory way for my practice.

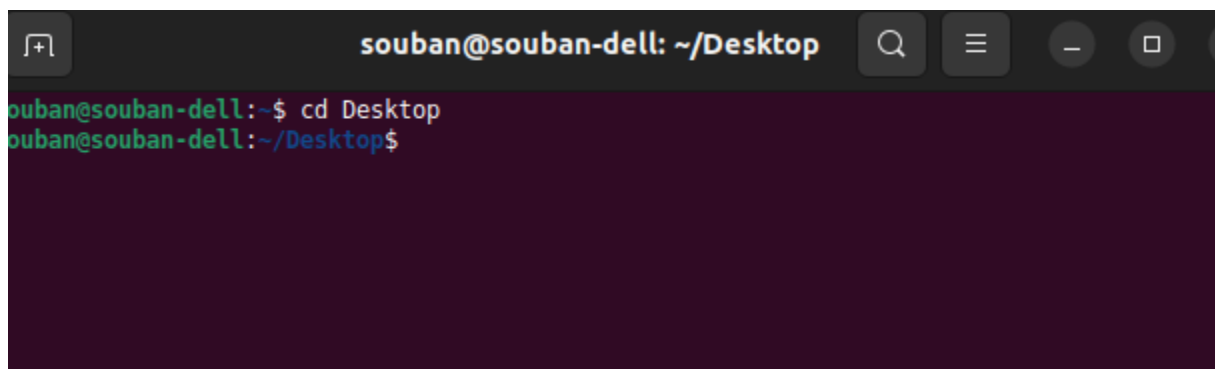
## Task # 2) Change to a different directory

### Syntax:

<code>cd</code>	/go to home directory
<code>cd directory_name</code> or <code>path/directory_name</code>	/go to specified directory from home
<code>cd ~/path/directory_name</code>	/go to specific directories from anywhere

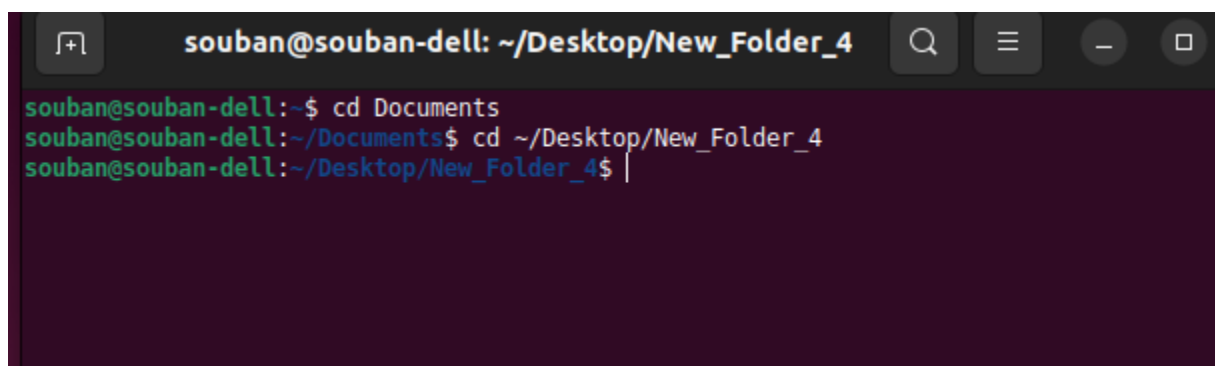
### Explanation

If we want to change our directory to some specified directory from **home**, let say from home to **Desktop** and if we want to go to a specific directory we will mention a path.



```
souban@souban-dell: ~/Desktop
souban@souban-dell:~$ cd Desktop
souban@souban-dell:~/Desktop$
```

If we want to change our directory directly to some other directory. We will add '~/' before the path we mention. Such as in the example below. We moved from **Documents** to **New\_Folder\_4** in **Desktop**



```
souban@souban-dell: ~/Desktop/New_Folder_4
souban@souban-dell:~$ cd Documents
souban@souban-dell:~/Documents$ cd ~/Desktop/New_Folder_4
souban@souban-dell:~/Desktop/New_Folder_4$ |
```

## Task # 3) List the contents of a directory

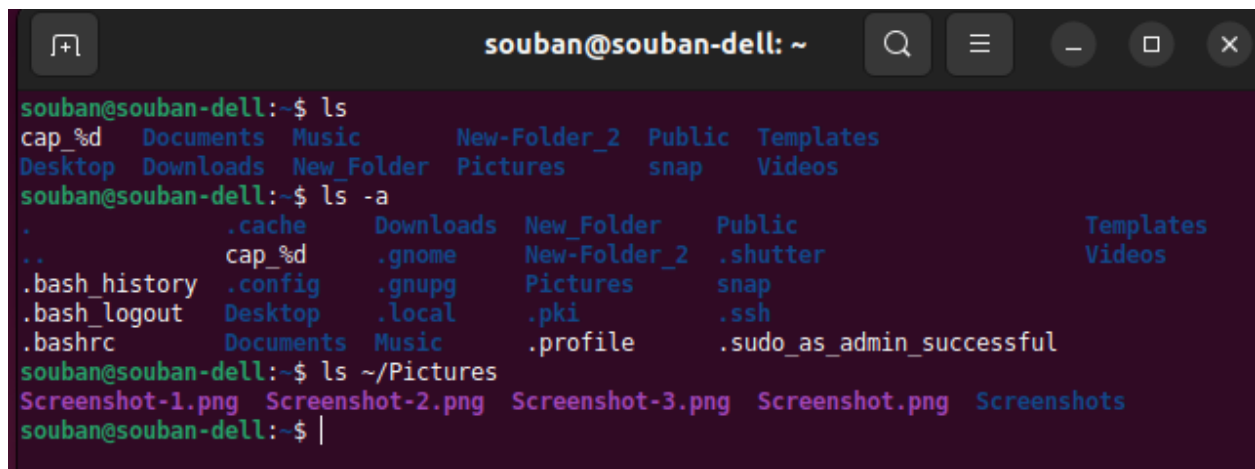
Syntax:

<b>ls</b>	/Shows all the contents files+folders without any arguments
<b>ls -a</b>	/Shows all the contents files+folders without any arguments
<b>ls ~/directory_name</b>	/Shows all the contents files+folders without any arguments

### Explanation

**Ls** command is used for listing the contents of any directory unless the directory path is provided. In the first, I enlisted the home directory, in the second place I enlisted all the contents along with hidden contents as well. There are plenty of other arguments which can be taken out from `–help`.

Lastly we can also enlist the contents of any directory by providing the directory name and path.

A terminal window titled 'souban@souban-dell: ~' with standard window controls. It shows three commands and their outputs. The first command is 'ls', which lists standard directories. The second command is 'ls -a', which lists all files including hidden ones like '.cache', '.gnome', and '.ssh'. The third command is 'ls ~/Pictures', which lists files in the Pictures directory, including several screenshot files.

```
souban@souban-dell:~$ ls
cap %d  Documents  Music      New-Folder_2  Public  Templates
Desktop Downloads  New Folder  Pictures      snap    Videos
souban@souban-dell:~$ ls -a
.          .cache    Downloads  New_Folder    Public      Templates
..         cap_%d    .gnome     New-Folder_2  .shutter    Videos
.bash_history .config  .gnupg     Pictures      snap
.bash_logout Desktop  .local     .pki          .ssh
.bashrc      Documents Music      .profile     .sudo_as_admin_successful
souban@souban-dell:~$ ls ~/Pictures
Screenshot-1.png Screenshot-2.png Screenshot-3.png Screenshot.png Screenshots
souban@souban-dell:~$ |
```

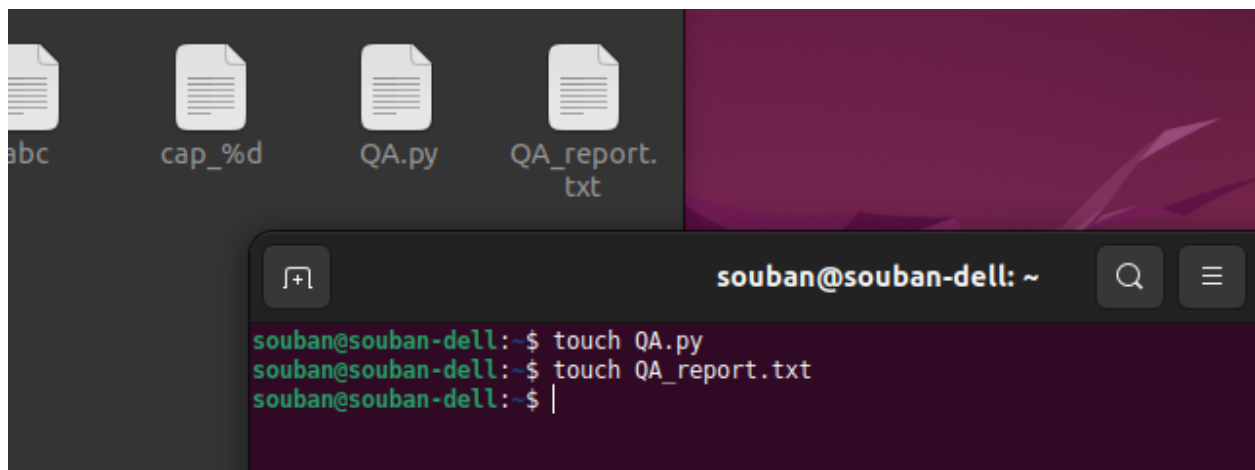
## Task # 4) Create a new file

Syntax:

<b>touch filename.(file type)</b>	/creates a file with specified file format
-----------------------------------	--

### Explanation

For creating file of whatever type or format we want we use touch commands along with filename and its type. If we don't mention file type. The terminal will create a text file in default.



## Task # 5) Open a file in Text editor

### Syntax:

**cat > filename.(filetype)**

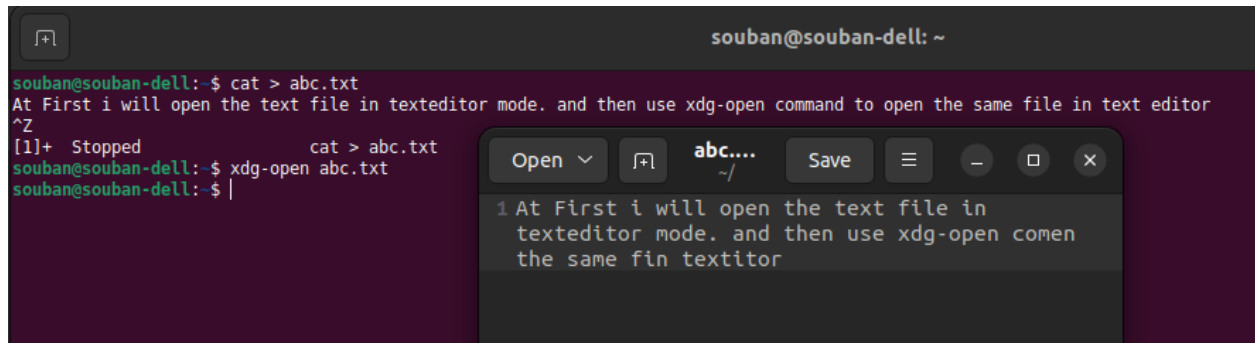
/open file in text editor in terminal

**xdg-open filename.(filetype)**

/open the file in text editor application

### Explanation

We can open the file in the terminal as a text editor by using the first command but if we want to open the file in text editor application **xdg-open filename.(filetype)**.



The image shows a terminal window and a text editor window. The terminal window has a title bar 'souban@souban-dell: ~'. The command prompt shows 'souban@souban-dell:~\$ cat > abc.txt'. Below this, there is a comment: 'At First i will open the text file in texteditor mode. and then use xdg-open command to open the same file in text editor'. The user presses '^Z', and the terminal shows '[1]+ Stopped cat > abc.txt'. Then, the user enters 'souban@souban-dell:~\$ xdg-open abc.txt' and the terminal shows 'souban@souban-dell:~\$ |'. The text editor window has a title bar 'abc.... ~/'. It has buttons for 'Open', 'Save', and window controls. The content of the file is: '1 At First i will open the text file in texteditor mode. and then use xdg-open comen the same fin textitor'.

## Task # 6) Copy a file to a new location

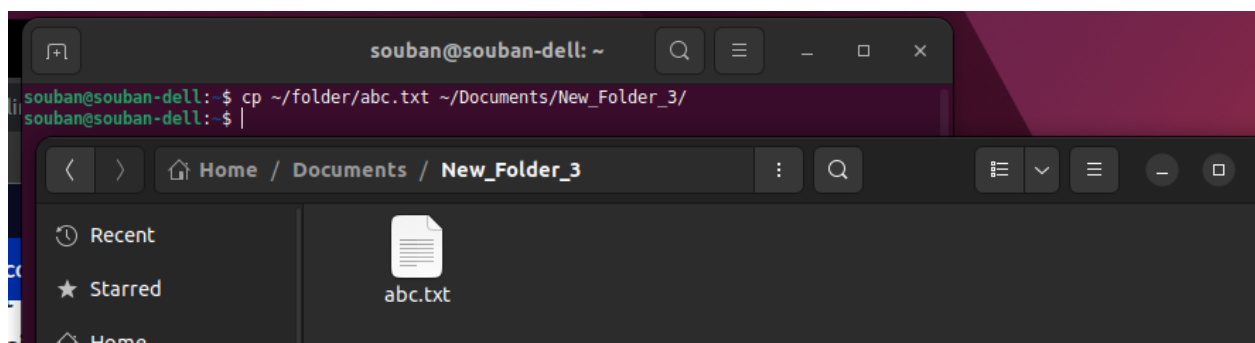
Syntax:

**cp** [~/path/filename.(filetype)] [~/path] /to copy the file to any destination from any source folder

**cp -r** [source] [destination] /copied the file from the specified source and putting it in the specified directory. If the directory is not present it will create it

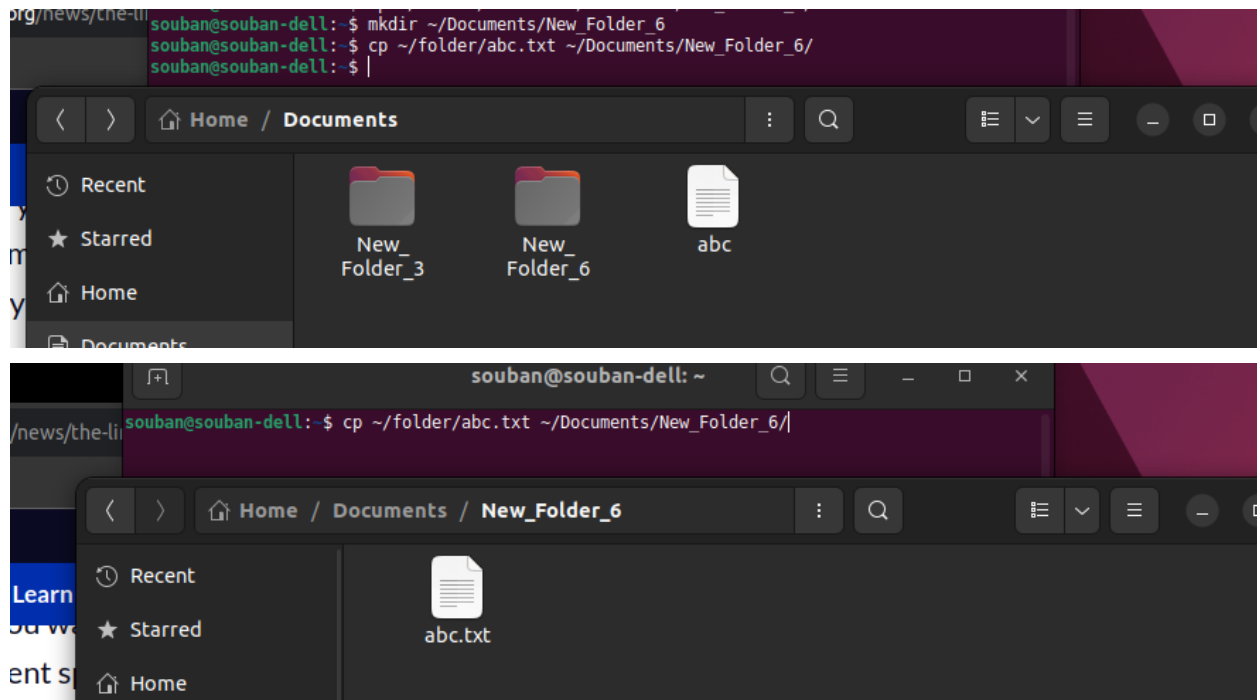
### Explanation

By using the above command we can copy the file from the source folder/directory to any other location/directory as far as we specify the path to it. In my command I am copying the file **abc.txt** from the directory folder located in **home** to the directory of **Documents** inside the folder named **New\_Folder\_3**.

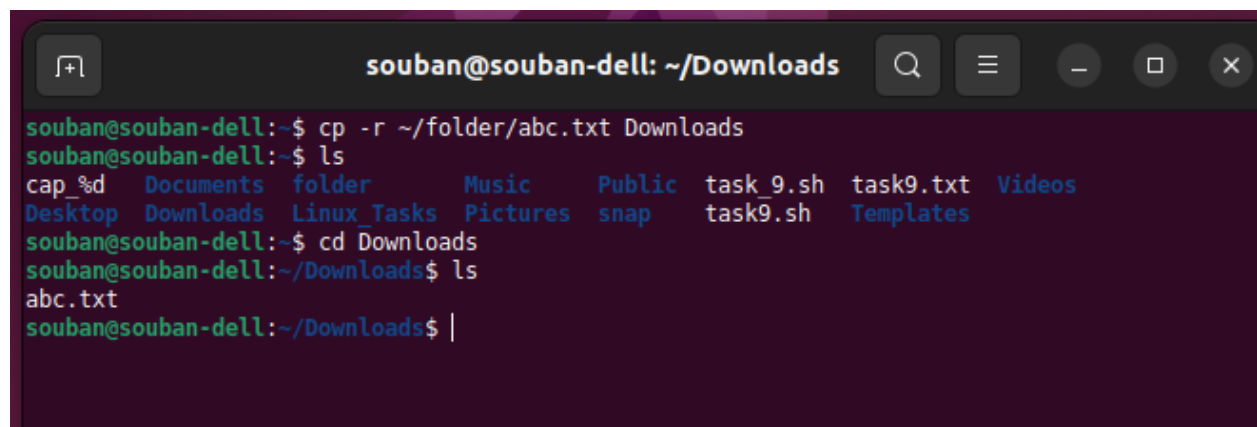


Similarly we can copy any file to any new directory by first creating the directory/location using the **mkdir** command and then copying the file to that location by using another command of **cp** as discussed above.

Just like in the steps below, making a new directory as **New\_Folder\_6** in the **Documents** folder and later by using the command of cp I am copying the file abc.txt into it which is shown in the next figure.



Cp -r command is used to copy the file abc.txt from the source folder and then moving it to new location Downloads.



## Task # 7) Rename a file

Syntax:

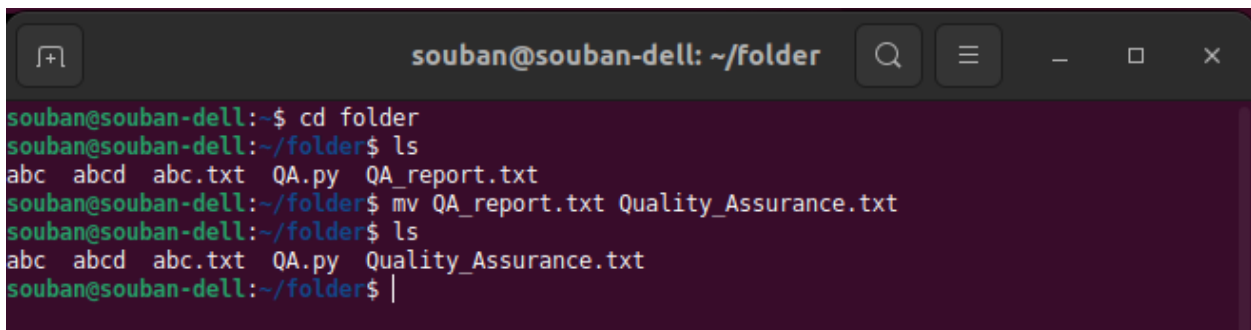
**mv [file1.(type)] [newfilename.(type)]**

/rename the file

### Explanation

By using the mv command and writing the name for the original file and then the new name for the file we can rename the file.

In the figure below, I have listed the files of the directory named **folder** and then renaming the original file named **QA.txt** into Quality\_Assurance.txt.

A terminal window titled 'souban@souban-dell: ~/folder' with search, menu, and window control icons. The terminal shows the following commands and output:

```
souban@souban-dell:~$ cd folder
souban@souban-dell:~/folder$ ls
abc  abcd  abc.txt  QA.py  QA_report.txt
souban@souban-dell:~/folder$ mv QA_report.txt Quality_Assurance.txt
souban@souban-dell:~/folder$ ls
abc  abcd  abc.txt  QA.py  Quality_Assurance.txt
souban@souban-dell:~/folder$ |
```

## Task # 8) Delete a file

Syntax:

**rm [filename.(type)]**

/remove file

**rm -rf [directory\_name]**

/remove the directory

### Explanation

Going to the desktop directory and listing all the files and folders. Creation of new file named as **newfile.txt** and then removing it using the rm command and displaying the contents of the directory again by listing it.



```
souban@souban-dell:~/folder$ cd ~/Desktop
souban@souban-dell:~/Desktop$ ls
New_Folder_4
souban@souban-dell:~/Desktop$ touch newfile.txt
souban@souban-dell:~/Desktop$ ls
newfile.txt  New_Folder_4
souban@souban-dell:~/Desktop$ rv newfile.txt

rv: command not found
souban@souban-dell:~/Desktop$
souban@souban-dell:~/Desktop$ rm newfile.txt
souban@souban-dell:~/Desktop$ ls
New_Folder_4
souban@souban-dell:~/Desktop$ |
```

Removing the whole directory from the path.

```
souban@souban-dell:~$ rm -rf folder
souban@souban-dell:~$ ls
cap_%d  Documents  Linux_Tasks  Pictures  snap      task9.sh  Templates
Desktop Downloads  Music        Public    task_9.sh task9.txt  Videos
souban@souban-dell:~$ |
```

## Mv Task # 9) Use variables to store and output data

### Explanation

This task is done in a .sh file by creating it using touch command and opening it with cat command. The variables are made termed as firstname, last name and company. An echo command which is similar to 'cout' in c++ and print as well. This echo is used to display the variables but with every variable in the echo statement we have to use \$ sign.

```
souban@souban-dell: ~
souban@souban-dell:~$ touch task9.sh
souban@souban-dell:~$ cat > task9.sh
#!/bin/bash
firstname="Souban"
lastname="Mehmood"
Company = "Emumba"

echo "My name is $firstname $lastname and I work at $Company."
^O
^X

^Z
[8]+  Stopped                  cat > task9.sh
```

```
souban@souban-dell: ~  
souban@souban-dell:~$ chmod +x task9.sh  
souban@souban-dell:~$ ./task9.sh  
./task9.sh: line 4: Emumba: command not found  
My name is Souban Mehmood and I work at  
souban@souban-dell:~$
```

## Task # 10) Create a simple script to automate a task

### Explanation

By touch command the file is created then by cat > command the file editor is opened and a simple task of opening the task9.txt file is done using the cat command on the filename mentioned inside the variable path\_of\_file.

```
souban@souban-dell:~$ touch task10.sh  
souban@souban-dell:~$ cat > task10.sh  
#!/bin/bash  
path_of_file="$Home/task9.txt"  
cat "$path_of_file"  
^O  
^X  
^X  
^Z  
[9]+ Stopped cat > task10.sh
```

```
task10.sh
1 #!/bin/bash
2 path_of_file="task9.txt"
3
4 if[ -f "$path_of_file"]; then
5 cat "$path_of_file"
6 else
7 echo "There is no such $path_of_file in the directory."
8 fi
9

souban@souban-dell:~$ chmod +x task10.sh
souban@souban-dell:~$ ./task10.sh
The script was made to open the file task9.txt in the home directory and tell a secret key "
347005".
souban@souban-dell:~$ |
```

Task # 11) Use if statement to make decisions in a script

Syntax:

cd /go to home directory

Explanation

This script is opening the text file by reading the name of file it is specified. First the if statement checks the correct file and as soon it is assessed correctly it will display the contents inside the file using cat command.

```
souban@souban-dell:~$ nano task10.sh
```

```
souban@souban-dell: ~
GNU nano 6.2 task10.sh *
#!/bin/bash
path_of_file="task9.txt"

if[ -f "$path_of_file"]; then
cat "$path_of_file"
else
echo "There is no such $path_of_file in the directory."
fi

souban@souban-dell:~$ chmod +x task10.sh
souban@souban-dell:~$ ./task10.sh
./task10.sh: line 4: syntax error near unexpected token `then'
./task10.sh: line 4: `if[ -f "$path_of_file"]; then'
souban@souban-dell:~$ nano task10.sh
souban@souban-dell:~$ chmod +x task10.sh
souban@souban-dell:~$ ./task10.sh
The script was made to open the file task9.txt in the home directory and tell a secret key "347005".
souban@souban-dell:~$
```

## Task # 12) Use loops to iterate over a set of data

Syntax:

Explanation

Creating the .sh file using nano and editing the file, declaring the string of fruits as per the figure below. For loop is written which will return the variable fruits array along with the string 'i like'. For the length of array the iteration will run.

```
souban@souban-dell:~$ nano task12.sh
souban@souban-dell:~$ chmod +x task12.sh
souban@souban-dell:~$ ./task12.sh
I like apple
I like banana
I like cherry
I like date
I like blueberry
souban@souban-dell:~$ |
```

```
task12.sh
~/
Open  Save  task12.sh
backup_script_compression.sh  data_extractor.py  task12.sh
1 #!/bin/bash
2 fruits=("apple" "banana" "cherry" "date" "blueberry")
3 for fruit in "${fruits[@]"; do
4     echo "I like $fruit"
5 done
```

```
souban@souban-dell:~/practice$ chmod +x practice.sh
souban@souban-dell:~/practice$ ./practice.sh
apple
banana
pineapple
souban@souban-dell:~/practice$
```

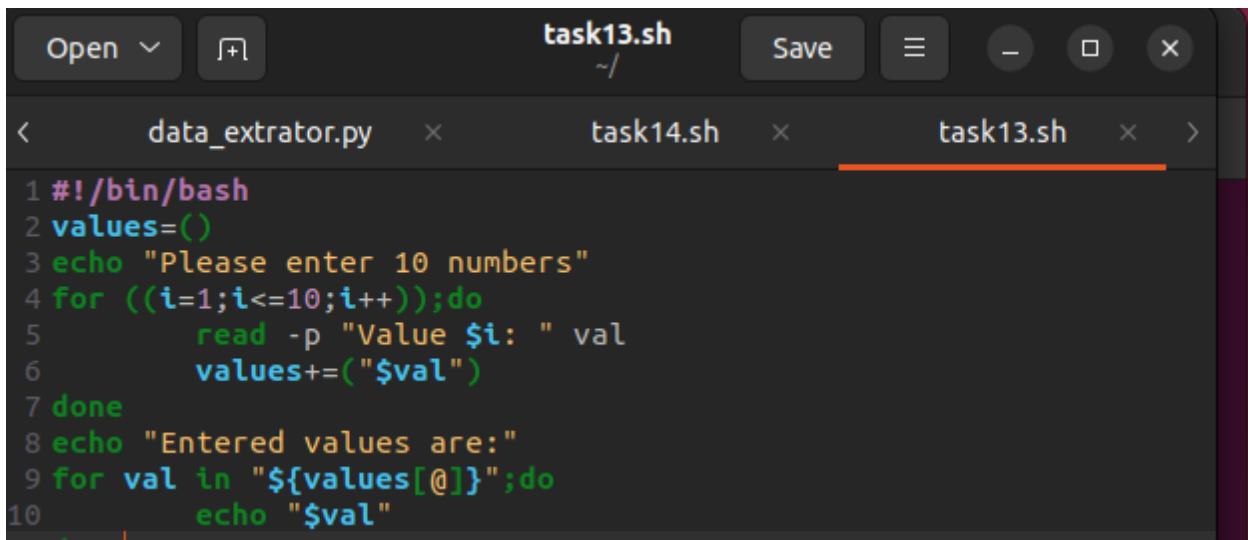
```
practice.sh
1 #!/bin/bash
2 string=("apple" "banana" "pineapple")
3 counter=0
4 while [ "$counter" -lt "${#string[@]}" ]; do
5     echo "${string[counter]}"
6     ((counter++))
7 done
```

## Task # 13) Read input from the user

### Explanation

Declaring the value argument for collecting the values by user. Displaying the message for the user to enter the 10 numbers. For loop for 10 iteration where it is taking the user input from the terminal inside the i variable. Once it is done 10 times, the values are displayed on the screen with a for loop by using the argument values

```
souban@souban-dell:~$ nano task13.sh
souban@souban-dell:~$ chmod +x task13.sh
souban@souban-dell:~$ ./task13.sh
Please enter 10 numbers
Value 1: 1
Value 2: 23
Value 3: 54
Value 4: 21
Value 5: 76
Value 6: 43
Value 7: 232
Value 8: 65
Value 9: 999
Value 10: 111
Entered values are:
1
23
54
21
76
43
232
65
999
111
souban@souban-dell:~$ |
```



```
task13.sh
~/
Save
data_extrator.py × task14.sh × task13.sh ×
1 #!/bin/bash
2 values=()
3 echo "Please enter 10 numbers"
4 for ((i=1;i<=10;i++));do
5     read -p "Value $i: " val
6     values+=("$val")
7 done
8 echo "Entered values are:"
9 for val in "${values[@]};do
10     echo "$val"
11 done
```

```

practice.sh
1 #!/bin/bash
2 string2=()
3
4 for((i=1; i<4; i++));do
5     echo "enter $i strings"
6     read user_input
7     string2+=("$user_input")
8 done
9
10 echo "you motto : "
11 for str in "${string2[@]}";do
12     echo "$str"
13 done

```

```

souban@souban-dell:~/practice$ chmod +x practice.sh
souban@souban-dell:~/practice$ ./practice.sh
enter 1 strings
all is well
enter 2 strings
that
enter 3 strings
ends well
you motto :
all is well
that
ends well
souban@souban-dell:~/practice$ |

```

Task # 14) Use command line arguments to customize a script

Explanation

```
souban@souban-dell: ~  
souban@souban-dell:~$ $Souban  
souban@souban-dell:~$ chmod +x task14.sh  
souban@souban-dell:~$ ./task14.sh  
Usage: ./task14.sh <name>  
souban@souban-dell:~$ ./task14.sh Souban  
Hello, Souban! Welcome to the script.  
souban@souban-dell:~$
```

```
task14.sh  
~/  
Open Save  
backup_script_compression.sh data_extrator.py task14.sh  
1#!/bin/bash  
2if [ $# -lt 3 ]; then  
3    echo "Usage: $0 <name1> <name2> <name3>"  
4    exit 1  
5fi  
6  
7for name in "$@";do  
8    echo "Hello, $name!"  
9done  
10echo "Welcome to the script."
```

```
souban@souban-dell: ~  
souban@souban-dell:~$ $Souban  
souban@souban-dell:~$ chmod +x task14.sh  
souban@souban-dell:~$ ./task14.sh  
Usage: ./task14.sh <name>  
souban@souban-dell:~$ ./task14.sh Souban  
Hello, Souban! Welcome to the script.  
souban@souban-dell:~$ nano task14.sh  
souban@souban-dell:~$ chmod +x task14.sh  
souban@souban-dell:~$ ./task14.sh Souban Usman Hamza  
Hello, Souban!  
Hello, Usman!  
Hello, Hamza!  
Welcome to the script.  
souban@souban-dell:~$ |
```

## Task # 15) Redirect output to a file

Syntax:

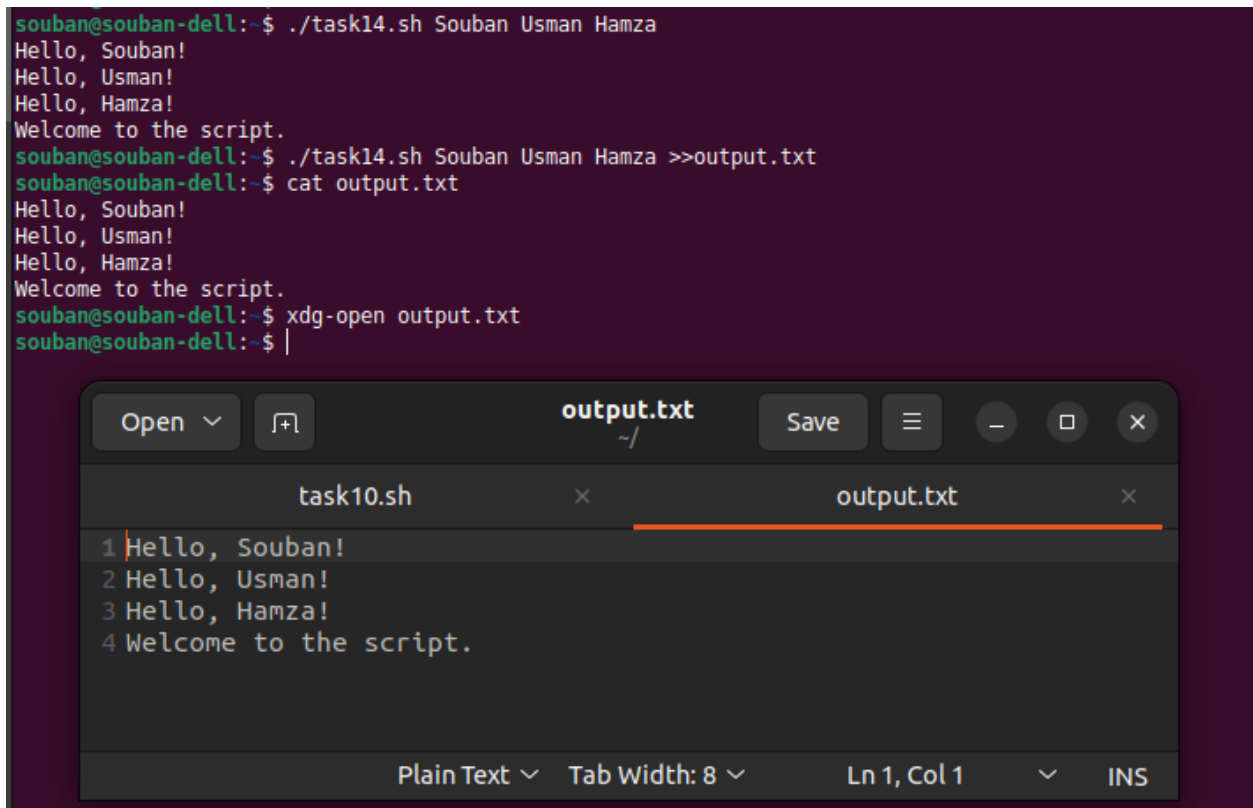
[object ] >> [defined output]



## Explanation

I am executing and running the script of task 14 and then displaying the output contents by first storing them inside a text file and later in the next command line i am displaying the text file.

```
souban@souban-dell:~$ ./task14.sh Souban Usman Hamza
Hello, Souban!
Hello, Usman!
Hello, Hamza!
Welcome to the script.
souban@souban-dell:~$ ./task14.sh Souban Usman Hamza >>output.txt
souban@souban-dell:~$ cat output.txt
Hello, Souban!
Hello, Usman!
Hello, Hamza!
Welcome to the script.
souban@souban-dell:~$ xdg-open output.txt
souban@souban-dell:~$ |
```

The image shows a terminal window with a dark purple background. The terminal output shows the execution of a script named 'task14.sh' with arguments 'Souban Usman Hamza'. The script prints 'Hello, Souban!', 'Hello, Usman!', 'Hello, Hamza!', and 'Welcome to the script.'. The output is then redirected to a file named 'output.txt' using the command 'cat output.txt'. The file is then opened using 'xdg-open output.txt'. Below the terminal window, a text editor window titled 'output.txt' is shown, displaying the same output as the terminal. The text editor has a dark theme and shows the following text: '1 Hello, Souban!', '2 Hello, Usman!', '3 Hello, Hamza!', and '4 Welcome to the script.'. The status bar at the bottom of the text editor shows 'Plain Text', 'Tab Width: 8', 'Ln 1, Col 1', and 'INS'.

## Task # 16) Chain commands together using pipes

### Syntax:

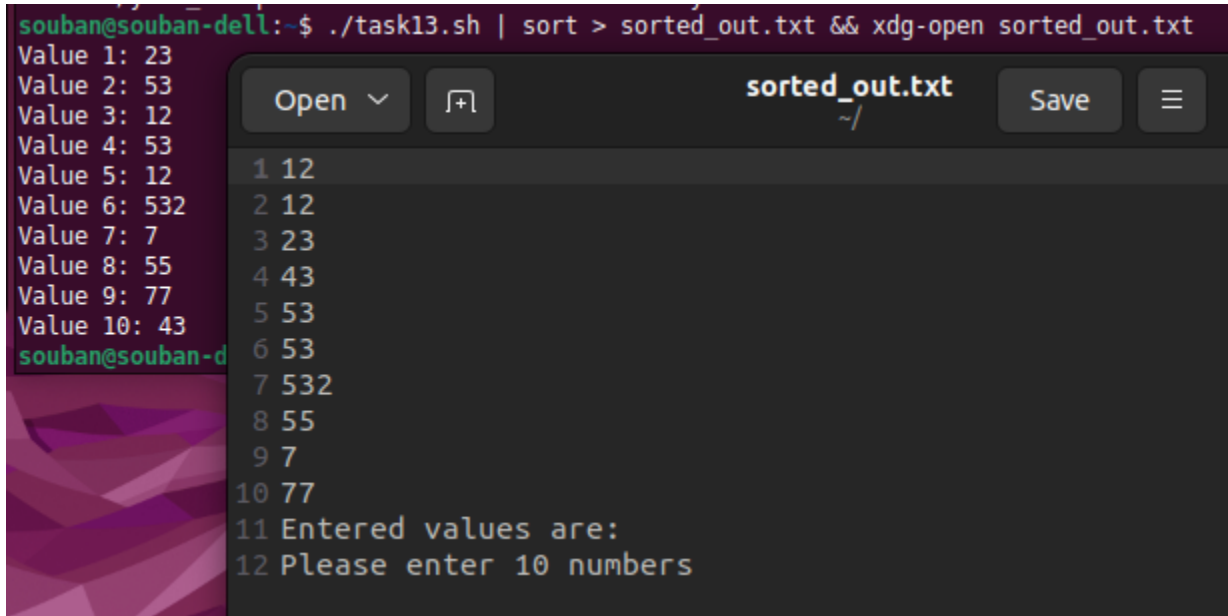
| and &&

## Explanation

These are basically the logic operators 'OR' & 'AND'. By OR means 'it has to do this along with this' and 'AND' means it has to do this and this as well'. which means in the command displayed below it is running the script along with sorting the output into a text

file and lastly the previous commands are completed so it is opening the sorted command in the text editor.

```
souban@souban-dell:~$ ./task13.sh | sort > sorted_out.txt && xdg-open sorted_out.txt
Value 1: 23
Value 2: 53
Value 3: 12
Value 4: 53
Value 5: 12
Value 6: 532
Value 7: 7
Value 8: 55
Value 9: 77
Value 10: 43
souban@souban-dell:~$
```



The screenshot shows a terminal window on the left and a text editor window on the right. The terminal window displays the command `./task13.sh | sort > sorted_out.txt && xdg-open sorted_out.txt` and its output, which lists 10 values. The text editor window, titled `sorted_out.txt`, shows the sorted output of the script, which is a list of 10 lines, each containing a number and a string. The text editor window also has buttons for `Open`, `Save`, and a menu icon.

```
1 12
2 12
3 23
4 43
5 53
6 53
7 532
8 55
9 7
10 77
11 Entered values are:
12 Please enter 10 numbers
```

## Task # 17) Use grep to search for text within a file

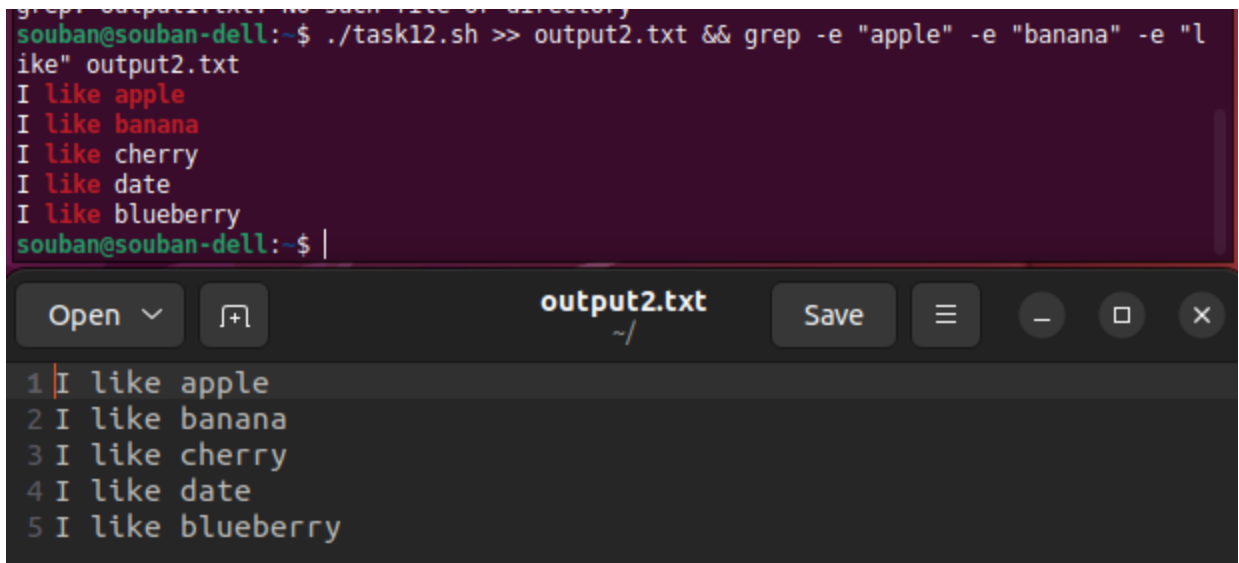
Syntax:

**Grep -e "word"**

### Explanation

First i ran the old script file of the task which was generating some string lines. Next the output of the file is stored in a text file named output.txt and then chained these command with grep which is used for searching the text withing a file and highlighting on the console. So i used multiple texts like 'apple', 'banana' and 'like'.

```
grep: output1.txt: No such file or directory
souban@souban-dell:~$ ./task12.sh >> output2.txt && grep -e "apple" -e "banana" -e "l
ike" output2.txt
I like apple
I like banana
I like cherry
I like date
I like blueberry
souban@souban-dell:~$ |
```



The image shows a terminal window and a file viewer window. The terminal window displays the execution of a shell script `./task12.sh` which appends its output to `output2.txt`, followed by a `grep` command that filters lines containing 'apple', 'banana', or 'like'. The output of the script is shown in the file viewer window, which is titled `output2.txt` and shows five lines of text.

Line	Content
1	I like apple
2	I like banana
3	I like cherry
4	I like date
5	I like blueberry

## Task # 18) Use sed to replace text within a file

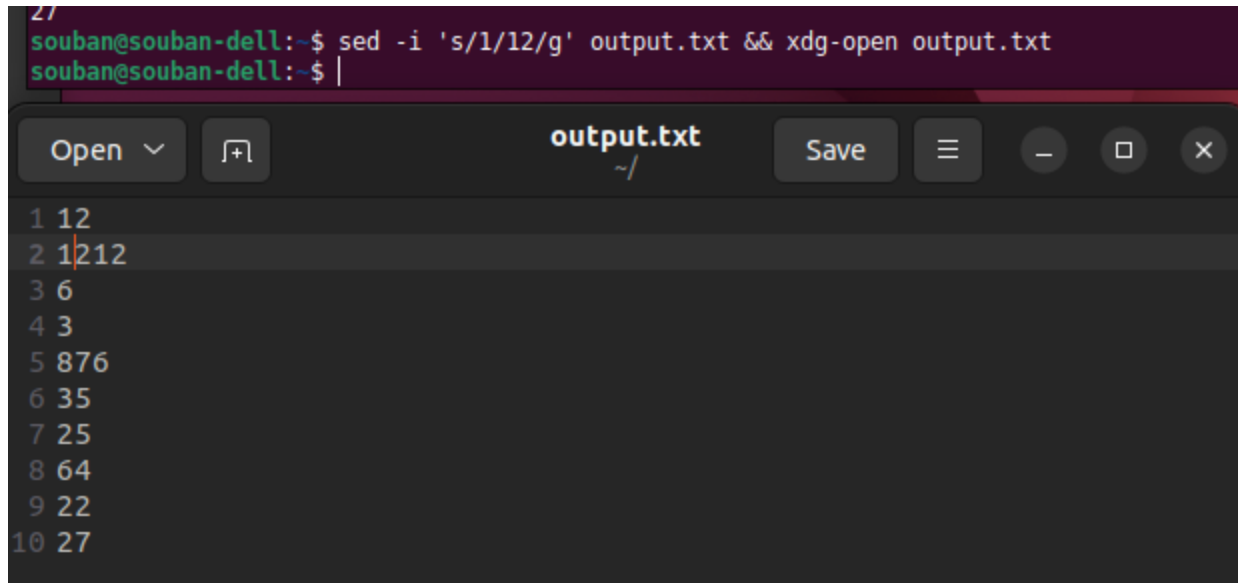
Syntax:

**Sed -i 's/word/word/g' filename.type**

### Explanation

-i in this is used for case insensitive search and the search objects and replacements are 1 into the integer 12. Also I chained the command with `xdg-open` to open the file with replacements.

```
27
souban@souban-dell:~$ sed -i 's/1/12/g' output.txt && xdg-open output.txt
souban@souban-dell:~$ |
```



```
1 12
2 1212
3 6
4 3
5 876
6 35
7 25
8 64
9 22
10 27
```

## Task # 19) Find and replace text in multiple files

**Syntax:**

```
sed -i -e 's/word1/word2/g' -e 's/word1/word2/g' file1.type file2.type
```

### Explanation

In the figure below it is first the display of all the contents inside the two text files and then in the next we used sed command along with argument -i that makes it insensitive search and -e for separate script or command. Then mentioning the word replacements in each segments along with the same order is mentioned the files needed to be accessed and replaced.

```
souban@souban-dell: ~
souban@souban-dell:~$ cat output.txt -e output2.txt
12$
1212$
6$
3$
876$
35$
25$
64$
22$
27$
I like pinepinepineapple$
I like banana$
I like cherry$
I like date$
I like blueberry$
souban@souban-dell:~$ sed -i -e 's/banana/pineapple/g' -e 's/12/313/g' output.txt output
2.txt
souban@souban-dell:~$ cat output.txt -e output2.txt
313$
313313$
6$
3$
876$
35$
25$
64$
22$
27$
I like pinepinepineapple$
I like pineapple$
I like cherry$
I like date$
I like blueberry$
souban@souban-dell:~$
```

Task # 20) Count the number of files in a directory and its subdirectories

Syntax:

```
find ~/path -type f |wc -l
```

/finding the file and counting the files and directories inside the path

Explanation

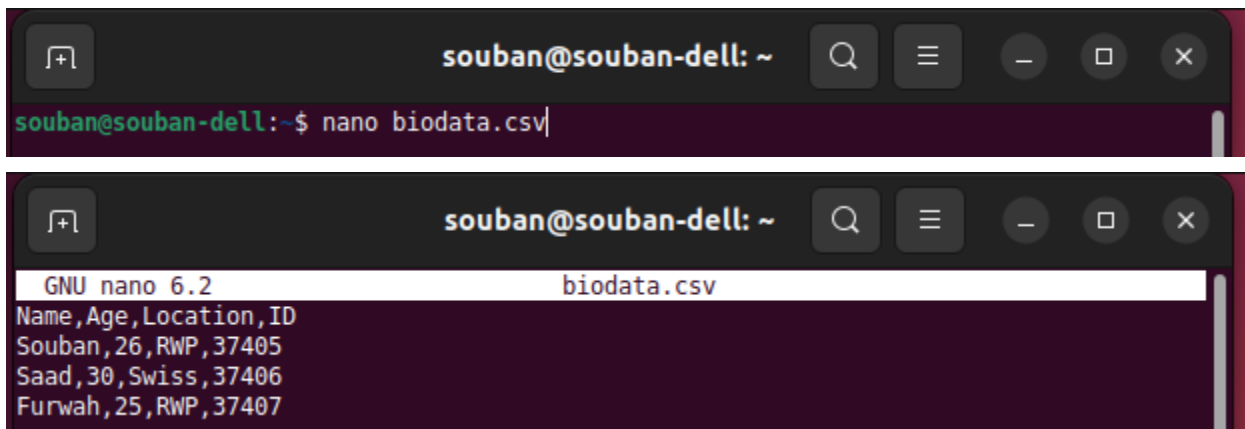
I am chaining the commands of find and count. We first write the command for finding the directory which is given. Then write the type -f for regular search and then after chaining the command with another one which is wc -l which will count the number of files.

```
souban@souban-dell:~$ find ~/ -type f | wc -l
34099
souban@souban-dell:~$ find ~/Pictures -type f | wc -l
26
souban@souban-dell:~$ find ~/Downloads -type f | wc -l
1
souban@souban-dell:~$
```

## Task # 21) Parse a CSV file and extract data

### Explanation

By using the nano command we create the csv file with the name biodata. Then we enter the data inside it with Name, Age, Location, ID as shown below. Now after saving the file we have to create a python script for scraping the data from the file.



```
souban@souban-dell: ~
souban@souban-dell:~$ nano biodata.csv
```

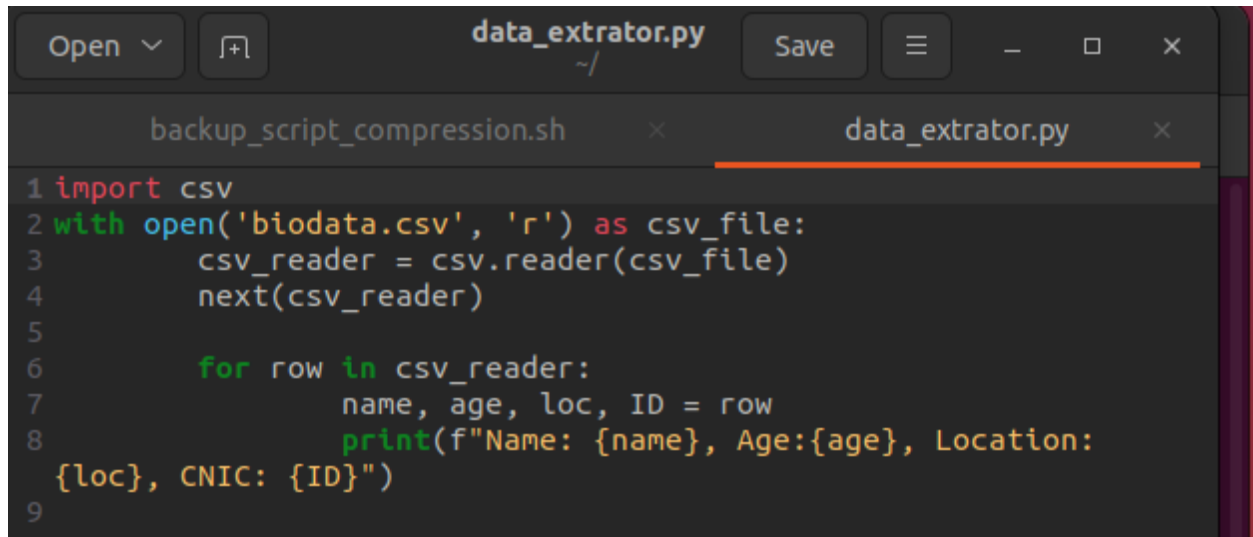
  

```
GNU nano 6.2 biodata.csv
Name, Age, Location, ID
Souban, 26, RWP, 37405
Saad, 30, Swiss, 37406
Furwah, 25, RWP, 37407
```

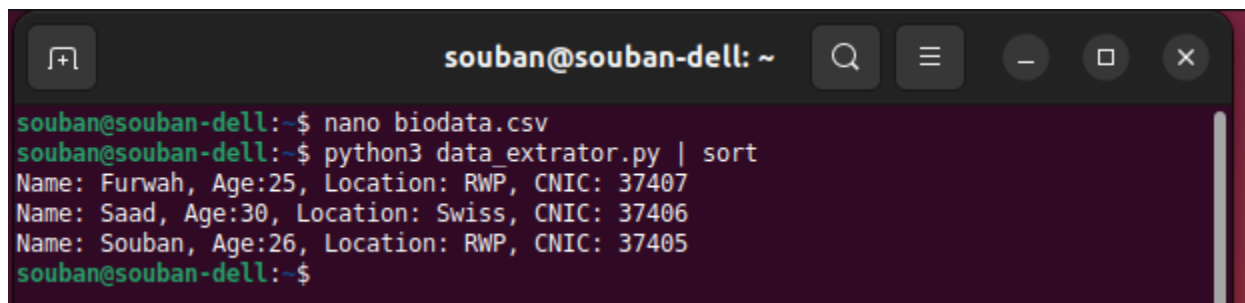
For the python file, we create the file using nano and start writing the script. Initially we need to import the library for importing csv. Now recursively we need to open the csv file and load the file. For that a syntax of with open() is used and the filename is inserted.

Next the python engine has to read the data from the csv file using the csv.reader() command. And for skipping the first line we use the command 'next'

Now a for loop in a row format is ran with printing sequence of Name, Age, Location and CNIC.

A screenshot of a code editor window. The title bar shows 'data\_extractor.py' and a file icon. The editor has two tabs: 'backup\_script\_compression.sh' and 'data\_extractor.py'. The 'data\_extractor.py' tab is active and shows the following Python code:

```
1 import csv
2 with open('biodata.csv', 'r') as csv_file:
3     csv_reader = csv.reader(csv_file)
4     next(csv_reader)
5
6     for row in csv_reader:
7         name, age, loc, ID = row
8         print(f"Name: {name}, Age:{age}, Location:
9         {loc}, CNIC: {ID}")
```

A screenshot of a terminal window. The title bar shows 'souban@souban-dell: ~'. The terminal shows the following commands and output:

```
souban@souban-dell:~$ nano biodata.csv
souban@souban-dell:~$ python3 data_extractor.py | sort
Name: Furwah, Age:25, Location: RWP, CNIC: 37407
Name: Saad, Age:30, Location: Swiss, CNIC: 37406
Name: Souban, Age:26, Location: RWP, CNIC: 37405
souban@souban-dell:~$
```

## Task # 22) Create a backup script that compresses files

### Explanation

At first I created the .sh file through touch command. Now the script inside it works in such a manner that a variable is created with the name "backup\_storage" which is storing the compressed file in home location.

Now a while statement is originated with a statement to check if the path entered by the user through read command is accurate or not. If it's not correct, the console will print the message to enter the path again. Once the path is corrected then a command for backup\_name of the file. Afterwards a directory for storage of the file is created and by using tar command the backup file which is created is stored in the backup storage path..

```
souban@souban-dell:~$ sudo bash backup_script_compression.sh
Please enter the path for file compression:
/home/souban/test folder
Filename/Directory: /home/souban/test folder
Backup completed successfully: test_folder_backup.tar.gz
```

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

```
backup_storage=~/. # Store backups in the home directory
```

```
# Taking user input for the directory/file to backup
```

```
while true; do
```

```
    echo "Please enter the path for file compression:"
```

```
    read -r backup_dir
```

```
    # Checking if the file or directory is valid
```

```
    if [ -e "$backup_dir" ]; then
```

```
        echo "Filename/Directory: $backup_dir"
```

```
        break
```

```
    else
```

```
        echo "File or directory is not valid or the path is incorrect."
```

```
    fi
```

```
done
```

```
# Creating a backup filename
```

```
backup_name=$(basename "$backup_dir")
```

```
# Directory for storage/backup of the file
```

```
backup_filename="${backup_name}_backup.tar.gz"
```

```
# Create the compressed backup archive
```

```
tar -czf "$backup_storage/$backup_filename" -C "$backup_dir" .
```

```
# Check if the backup was successful
```

```
if [ $? -eq 0 ]; then
```

```
    echo "Backup completed successfully: $backup_filename"
```

```
else
```

```
    echo "Backup failed."
```

```
fi
```