



# Parul University

FACULTY OF ENGINEERING & TECHNOLOGY

BACHELOR OF TECHNOLOGY

(303105103) Open Source Software

1st SEMESTER

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

# Laboratory Manual

**(303105103) Open Source Software PRACTICAL BOOK**

**COMPUTER SCIENCE AND ENGINEERING DEPARTMENT**

## **PREFACE**

It gives us immense pleasure to present the first edition of *(303105103) Open Source Software* for the B.Tech. 1st year students for PARUL UNIVERSITY.

The Fundamental of Programming theory and laboratory courses at **PARUL UNIVERSITY, WAGHODIA, VADODARA** are designed in such a way that students develop the basic understanding of the subject in the theory classes and then try their hands on the computer learnt during the theoretical sessions.

This book is emphatically not focused on “The only open source software”. Understanding the fundamental ideals, principals, and techniques is the essence of a good programmer. Only well-designed code has a chance of becoming part of a correct, reliable, and maintainable system. Also, “the fundamentals” are what last: they will still be essential after today’s language and tools have evolved or been replaced.

We acknowledge the authors and publishers of all the books which we have consulted while developing this Practical book. Hopefully this *(303105103) Open Source Software* will serve the purpose for which it has been developed.

## **Instructions to students**

1. Every student should obtain a copy of laboratory Manual.
2. Dress Code: Students must come to the laboratory wearing.
  - i. Trousers,
  - ii. half-sleeve tops and
  - iii. Leather shoes. Half pants, loosely hanging garments and slippers are not allowed.
3. To avoid injury, the student must take the permission of the laboratory staff before handling any machine.
4. Students must ensure that their work areas are clean and dry to avoid slipping.
5. Do not eat or drink in the laboratory.
6. Do not remove anything from the computer laboratory without permission.
7. Do not touch, connect or disconnect any plug or cable without your lecturer/laboratory technician's permission.
8. All students need to perform the practical/program.

## CERTIFICATE

*This is to certify that*

*Mr./Ms \_\_\_\_\_ with enrolment no.*

*\_\_\_\_\_ has successfully completed his/her laboratory experiments in the  
(303105103) Open Source Software From the department of  
..... during the academic year .....*



Date of Submission:.....

Staff In charge:.....

Head of Department:.....

**INDEX**

**Class: 1st Sem**

**SUBJECT: Open Source Software**

**A.Y.20\_\_-20\_\_**

**SUBJECT CODE (303105103)**

sl. no	Experiment Title	Page no		Date of performance	Date of assessment	Marks out of 10	Sign
		Fr om	To				
	<b>Practical set 1</b>						
1	Demonstration Of Basic Linux Commands.						
2	Execute C program Using gcc Compiler						
3	Demonstration of gprof Command using linux.						
4	Create and Edit documents using Google Docs.						
5	Create Presentation Using Google Slides.						
6	Demonstration of different Arithmetic and Logical Formulas using Open Office Calc						
7	Use Of HTML to create Simple Web Page.						
8	Demonstration of MathML-a markup language for describing mathematical notation						
9	Demonstration of virtualization using Docker Container						
10	Demonstration of GitHub Facility						

## PRACTICAL 1

**AIM: Demonstration of Basic Linux commands.**

**Command shell: A program that interprets commands is Command shell.**

**Shell Script: Allows a user to execute commands by typing them manually at a terminal, or automatically in programs called shell scripts.**

**A shell is not an operating system .It is a way to interface with the operating system and run Commands.**

**BASH (Bourne Again Shell)**

**Bash is a shell written as a free replacement to the standard Bourne Shell (/bin/sh) originally written by Steve Bourne for UNIX systems.**

**It has all of the features of the original Bourne Shell, plus additions that make it easier to program with and use from the command line.**

**Since it is Free Software, it has been adopted as the default shell on most Linux systems.**

**BASIC LINUX COMMANDS:**

**pwd : Print Working Directory DESCRIPTION:**

**pwd prints the full pathname of the current working directory.**

**SYNTAX:**

**pwd**

**EXAMPLE:**

**\$pwd**

**/home/directory\_name**

**cd: Change Directory DESCRIPTION:**

**It allows you to change your working directory. You use it to move around within the hierarchy of your file system.**

**SYNTAX:**

**cd directory\_name**

**EXAMPLE:**

To change into "work directory" in "documents" need to write as follows.

**Input:** \$cd/documents/work

**cd..**

**DESCRIPTION:**

**Move up one directory.**

**SYNTAX:**

**cd..**

**EXAMPLE:**

If you are in work directory and want to go to documents then write **cd..**

You will end up in documents.

**ls** : list all the files and directories

**DESCRIPTION:**  
List all files and folders in the current directory in the column format.

**SYNTAX:**

**ls[options]**

**EXAMPLE:** Using various options

Lists the total files in the directory and subdirectories, the names of the files in the current directory, their permissions, the number of subdirectories in directories listed, the size of the file, and the date of last modification.

**ls-l**

List all files including hidden files

**ls-a**

**cat** **DESCRIPTION:**

**cat** stands for "catenate". It reads data from files, and outputs their contents. It is the simplest way to display the contents of a file at the command line.

**SYNTAX:**

**cat filename**

**EXAMPLES:**

**Print the contents of files mytext.txt and yourtext.txt**  
`cat mytext.txt yourtext.txt`

**Print the cpu information using cat command**  
`cat /proc/cpuinfo`

**Print the memory information using cat command**

`cat /proc/meminfo`

**head DESCRIPTION:**

**head**, by default, prints the first 10 lines of each FILE to standard output. With more than one FILE, it precedes each set of output with a header identifying the filename.

If no FILE is specified, or when FILE is specified as a dash ("-"), head reads from standard input.

**SYNTAX:**

`head [option]... [file/directory]`

**EXAMPLE:**

**Display the first ten lines of myfile.txt.**

`head myfile.txt`

`tail`

**DESCRIPTION:**

**tail** is a command which prints the last few number of lines (10 lines by default) of a certain file, then terminates.

**SYNTAX:**

`tail [option]... [file/directory]`

**EXAMPLE:**

**Output the last 100 lines of the file myfile.txt.**  
`tail myfile.txt -n 100`

**mv : Moving (and Renaming) Files**

**DESCRIPTION:**

The **mv** command lets you move a file from one directory location to another. It also lets you rename a file (there is no separate rename command).



**SYNTAX:**

**mv[option]sourcedirectory**

**EXAMPLE:**

**Moves the file myfile.txt to the directory destination-directory.mvmyfile.txt  
destination\_directory**

**Movethefilemyfile.txtintotheparentdirectory.**

**mvmyfile.txt../**

**In this case, if JOE1\_expenses does not exist, it will be created with the exact  
contentofjoe\_expenses,and joe\_expenses will disappear.**

**If JOE1\_expenses already exists, its content will be replaced with thatofjoe\_expenses  
(andjoe\_expenses willstill disappear).**

**mvjoe\_expensesJOE1\_expenses**

**mkdir : Make DirectoryDESCRIPTION:**

**Ifthespecifieddirectorydoesnotalreadyexist,  
mkdircreatesit.Morethanonedirectorymaybespecified whencallingmkdir.**

**SYNTAX:**

**mkdir[option]directory**

**EXAMPLE:**

**Create a directorynamed work.mkdirwork**

**cp : Copy FilesDESCRIPTION:**

**Thecpcommandisusedtomakecopyoffilesanddirectories.**

**SYNTAX:**

**cp[option]sourcedirectory**

**EXAMPLE:**

**Creates a copy of the file in the currently working directory named origfile. The copy will  
benamednewfile,and will belocated in theworkingdirectory.**

**cporigfilenewfile**

**rmdir : Remove Directory**DESCRIPTION:

The **rmdir** command is used to remove a directory that contains no other files or directories.

**SYNTAX:**

**rmdir**directory\_name

**EXAMPLE:**

Delete mydir directory along with all files and directories within that directory. Here, **-r** is for recursive and **-f** is for forcefully.

**rmdir -rf mydir**

**gedit**DESCRIPTION:

The **gedit** command is used to create and open a file.

**SYNTAX:**

**gedit**filename.txt

**EXAMPLE:**

To create a file named abc.sh **gedit abc.sh**

**man**DESCRIPTION:

Display an online manual page or man page.

**SYNTAX:**

**man**command

**EXAMPLE:**

To learn about listing files **man ls**

**echo**DESCRIPTION:

Display text on the screen.

**SYNTAX:**

**echo**yourtext

**EXAMPLE:**

Print Hello World on the screen **echo "Hello World"**

**clearDESCRIPTION:**

Used to clear the screen

**SYNTAX:**

**clear**

**EXAMPLE:**

Clear the entire screen

**clear**

**whoamiDESCRIPTION:**

whoami prints the effective user ID. This command prints the username associated with the current effective user ID.

**SYNTAX:**

**whoami[option]**

**EXAMPLE:**

Display the name of the user who runs the command. **whoami**

**wcDESCRIPTION:**

wc (word count) command, can return the number of lines, words, and characters in a file.

**SYNTAX:**

**wc[option]...[file]...**

**EXAMPLE:**

Print the byte counts of file myfile.txt **wc -c myfile.txt**

Print the line count of file myfile.txt

**wc -l myfile.txt**

Print the word counts of file myfile.txt **wc -w myfile.txt**

**grep**

**DESCRIPTION:**

grep command uses a search term to look through a file.

**SYNTAX:**

**grep[option]...Pattern[file]...**

**EXAMPLE:**

**Search the word Hello in file named myfile.txt**  
**grep "Hello"myfile.txt**

**DESCRIPTION:**

**Display RAM details in Linux machine.**

**SYNTAX:**

**free**

**EXAMPLE:**

**To display the RAM details in Linux machine need to write following command.**  
**free**

**pipe ( | )DESCRIPTION:**

**Pipe command is used to send output of one program as a input to another. Pipes “|” help combine 2 or more commands.**

**SYNTAX:**

**Command1|command2**

**EXAMPLE:**

**Display lines of input files containing “Aug” and send to standard output**  
**ls-l|grep “Aug”**

## Practical 2

**AIM: Execute C Program using gcc compiler.**

**Step 1. Open up a terminal**

**Search for the terminal application in the Dash tool**



**Step 2. Use a text editor to create the C source code.**

**Type the command**

**gedit hello.c**

**and enter the C source code below:**

```
#include <stdio.h>
```

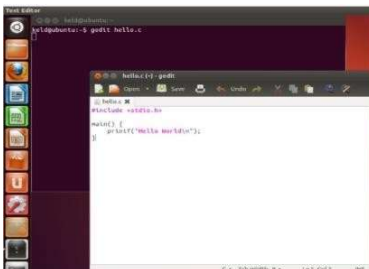
```
main()
```

```
{
```

```
printf("Hello World\n");
```

```
}
```

**Close the editor window.**



**Step 3. Compile the program.**

**Type the command**

**gcc -o hello hello.c**

**This command will invoke the GNU C compiler to compile the file hello.c and output (-o) the result to an executable called hello.**



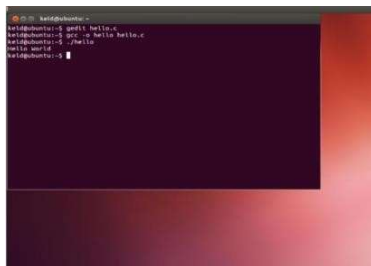
```
helip@helip:~$ gcc -o hello hello.c
helip@helip:~$ gcc -o hello hello.c
helip@helip:~$
```

**Step 4. Execute the program.**

**Type the command**

**./hello**

**This should result in the output  
HelloWorld**



```
helip@helip:~$ gcc -o hello hello.c
helip@helip:~$ gcc -o hello hello.c
helip@helip:~$ ./hello
HelloWorld
helip@helip:~$
```

## PRACTICAL 3

**AIM:** Demonstration of gprof command using Linux.

### **THEORY:**

The objective of profiling is to analyze your program code and see which part of the code is taking a large amount of time for execution such that the part of the code can be rewritten. This will enable the program to achieve desired execution speed. In addition, profiling can prove to be very handy in spotting codes that are potentially error-prone, and then they can be sorted out via refactoring.

Using the gprof is quite simple. First, you need to enable the profiling when you compile the code. Now, when you execute the program, profile data is generated. Finally, you can run the gprof tool on the profiling data generated during execution. This will produce an analysis file. This file contains several tables, including a flat profile and a call graph.

### **Enable profiling while compiling**

To enable profiling while compiling, simply add the -pg option in the compilation step. The pg flag generates extra code for profiling that is used by the gprof command. The following command illustrates how we can compile profiling information:

```
$ gcc -Wall -pg program.c program_new.c -o program
```

### **Execute the code to generate profile information**

The binary file generated above can be executed to generate profile information. The following line executes the code:

```
$/program
```

This will also generate a file gmon.out in the current working directory. To see what files are generated via execution of the above command, simply type ls as follows:

```
$ls
```

### **Run the gprof tool**

Now, we will run the gprof tool providing as an argument the output file and gmon.out file. This will produce the profiling information:

```
$ gprof program gmon.out > analysis.txt
```

## PROGRAM CODE:

### main.c file

```

1  #include "foo.h"
2
3  void function_main();
4  void function2_main();
5
6
7  void function_main()
8  {
9      int i=0, k=0, j = 0;
10
11      for(j=0;j<1000;j++)
12          for(i=0;i<100000;i++)
13              k = i%5;
14
15  }
16
17  void function2_main()
18  {
19      function2_foo();
20  }
21
22  int main()
23  {

```

```

24      function_main();
25      function2_main();
26      function2_foo();
27      return 0;
28  }

```

### foo.c file:

```

1  #include "foo.h"
2
3  void function_foo()
4  {
5      int i = 0, k = 0, j=0;
6      for(j=0;j<1000;j++)
7          for(i=0;i<100000;i++)
8              k = i%5;
9  }
10
11  void function2_foo()
12  {
13      int i = 0, k = 0, j = 0;
14
15      for(j=0;j<1000;j++)
16          for(i=0;i<100000;i++)
17              k = i + 5;
18  }

```

### Foo.h file

```

1  #ifndef FOO_H
2  #define FOO_H
3
4  #include <stdio.h>
5
6  void function_foo();
7  void function2_foo();
8
9  #endif

```

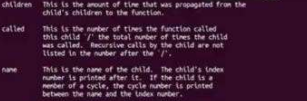


**OUTPUT:**

```

main() {
    FILE *f;
    f = fopen("main.c", "w");
    fprintf(f, "/*\n");
    fprintf(f, " * 1. This program is a simple example of a\n");
    fprintf(f, " * 2. program that uses the getopt\n");
    fprintf(f, " * 3. library to parse command line\n");
    fprintf(f, " * 4. options.\n");
    fprintf(f, " * 5. It is a simple example of a\n");
    fprintf(f, " * 6. program that uses the getopt\n");
    fprintf(f, " * 7. library to parse command line\n");
    fprintf(f, " * 8. options.\n");
    fprintf(f, " */\n");
    fprintf(f, " * 9. It is a simple example of a\n");
    fprintf(f, " * 10. program that uses the getopt\n");
    fprintf(f, " * 11. library to parse command line\n");
    fprintf(f, " * 12. options.\n");
    fprintf(f, " */\n");
    fprintf(f, " */\n");
    fclose(f);
}

```



The screenshot shows a C++ program with the following code:

```

1 // This program illustrates the recursive nature of this function.
2
3 #include <iostream>
4
5 using namespace std;
6
7 int func(int n)
8 {
9     // base case
10    if (n == 1)
11        return 1;
12    // recursive case
13    return n + func(n-1);
14 }
15
16 int main()
17 {
18     // call the function func()
19     int n = 5;
20     int res = func(n);
21     cout << "The value of the recursive function is: " << res << endl;
22     return 0;
23 }

```

The output of the program is:

```

The value of the recursive function is: 15

```

The program demonstrates a recursive function `func` that calculates the sum of integers from 1 to `n`. The base case is `n == 1`, returning 1. The recursive case is `n + func(n-1)`. The function is called with `n = 5`, and the result is printed as 15.

[illegible]

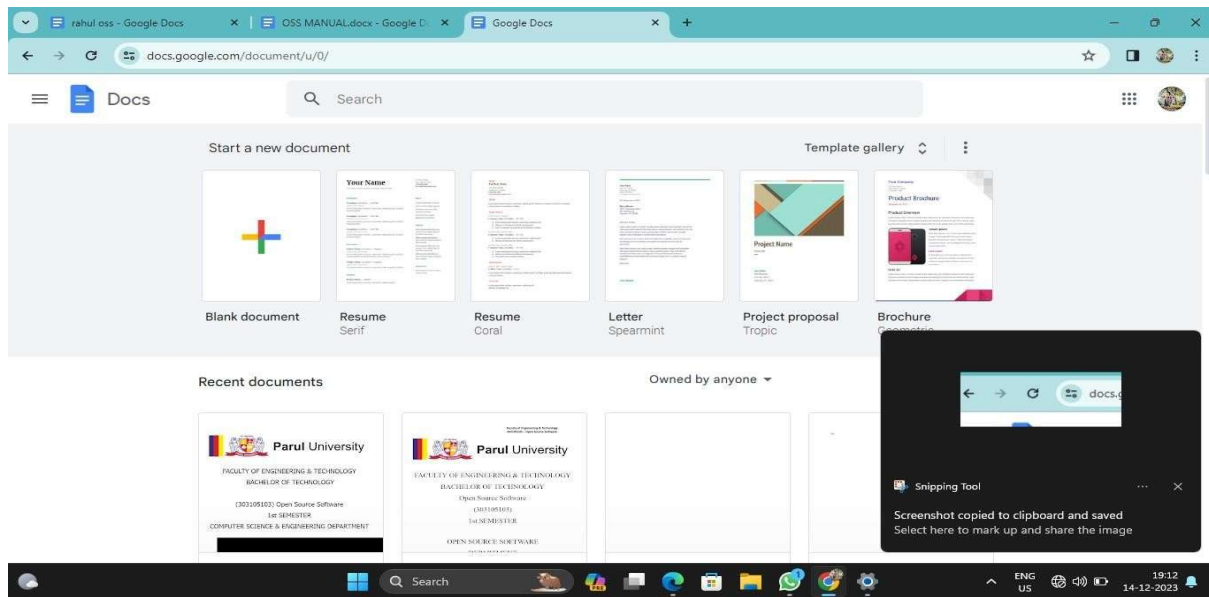
## Practical 4

**AIM: Create and Edit documents using Google Docs.**

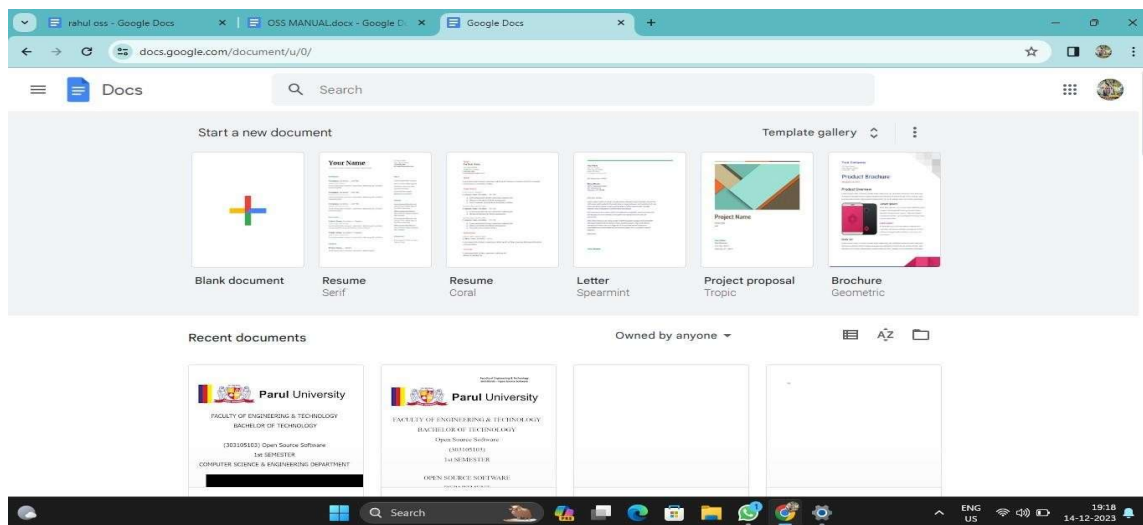
### Step 1: Create a document

**To create a new document:**

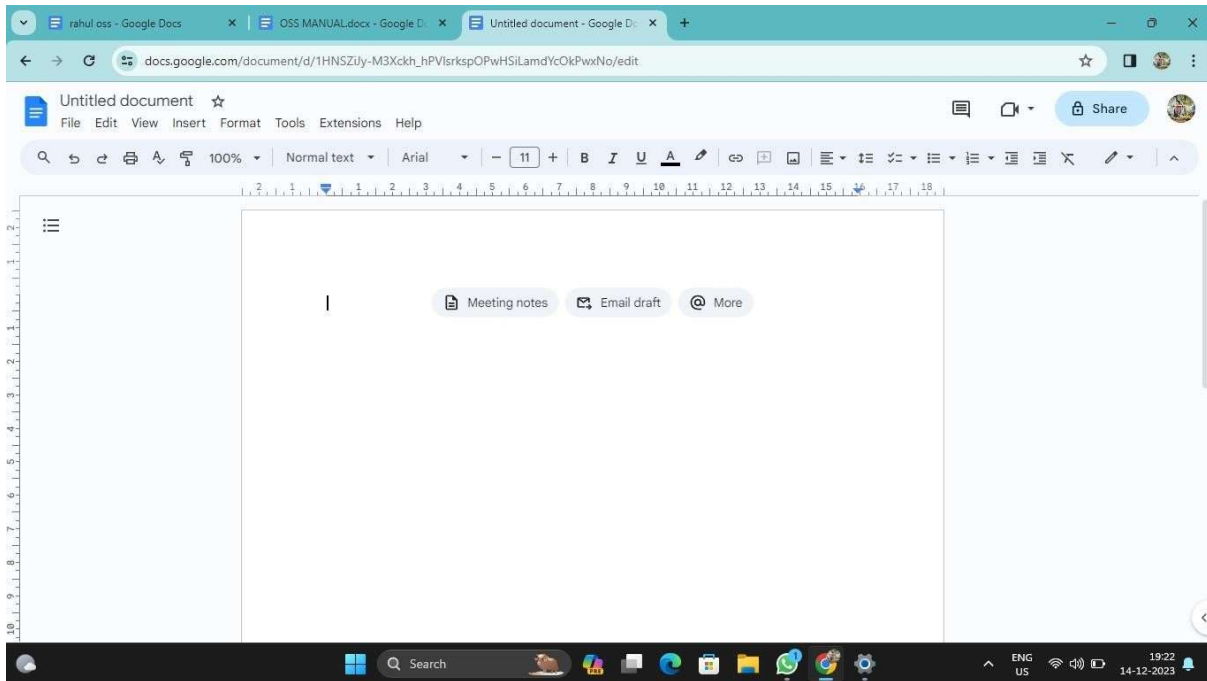
**1. On your computer, open the Docs home screen at docs.google.com.**



**2. In the top left, under "Start a new document," click Blank New.**



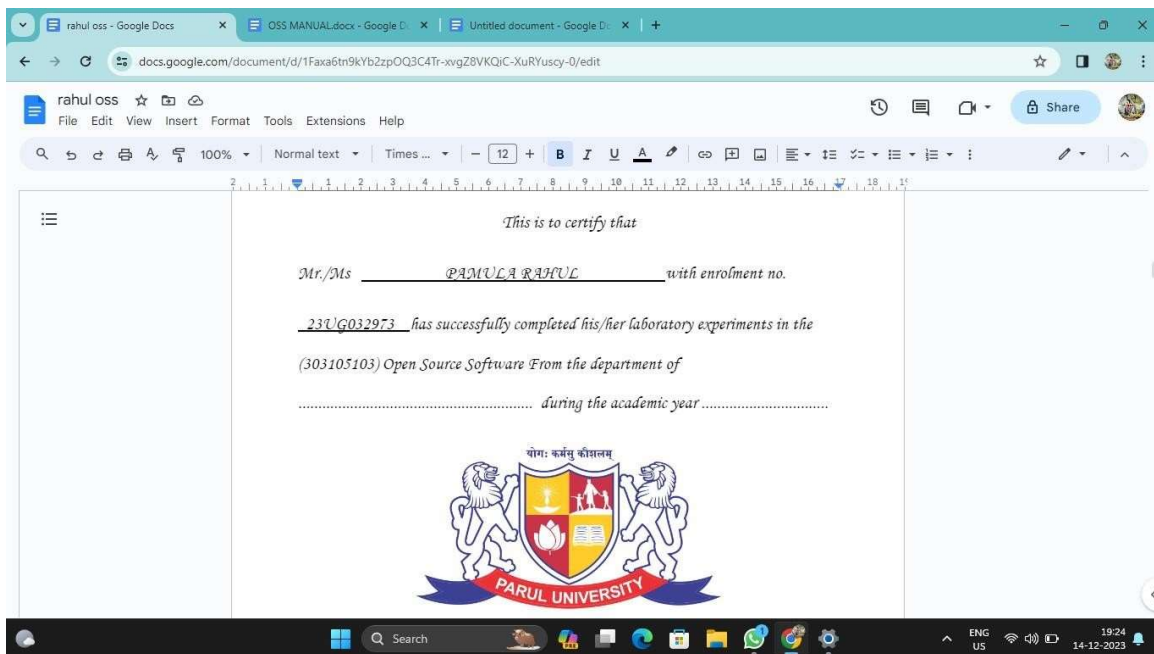
You can also create new documents from the URL [docs.google.com/create](https://docs.google.com/create).



## Step 2: Edit and format

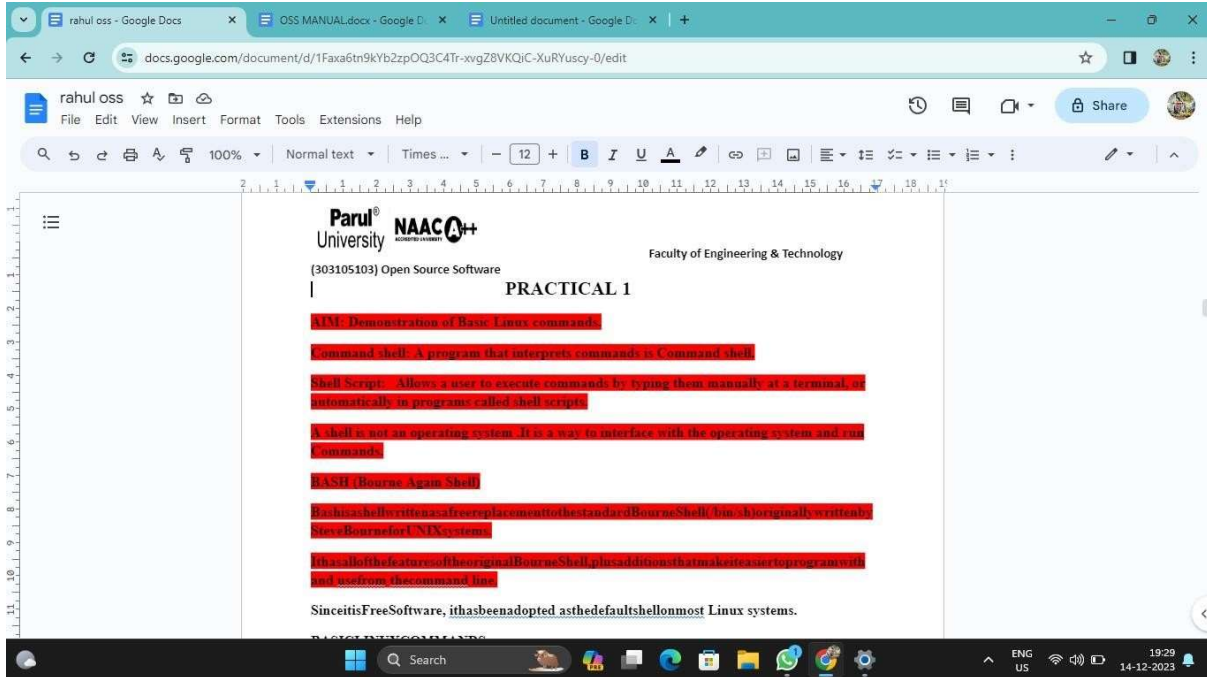
To edit a document:

### 1. On your computer, open a document in Google Docs.

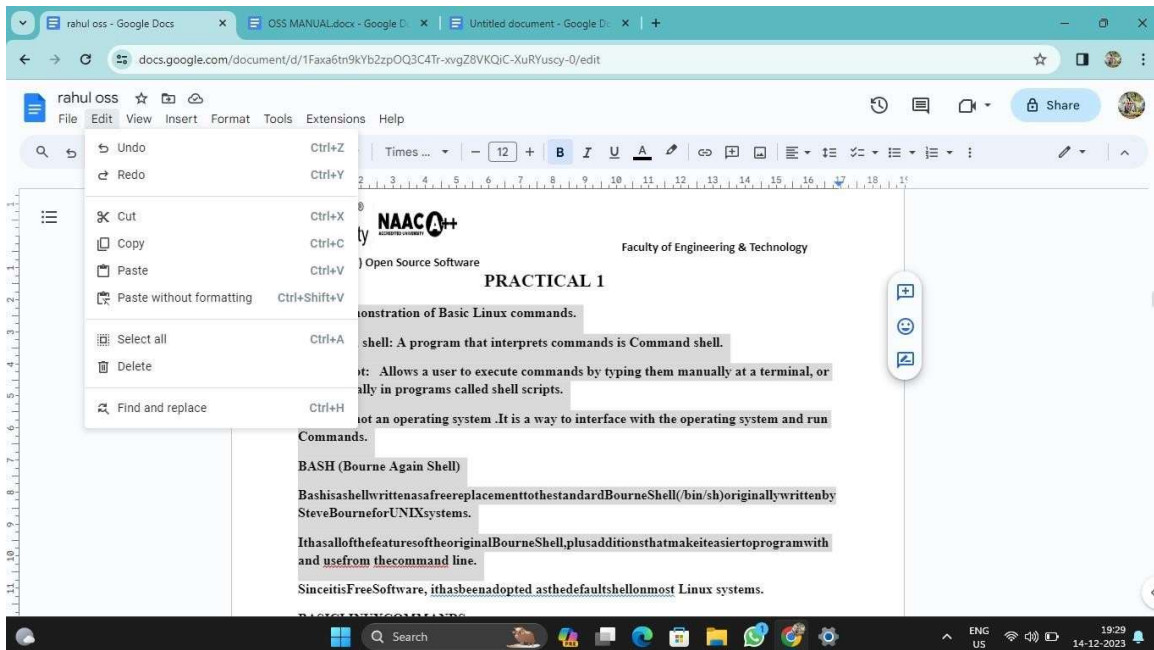


2. To select a word, double-click it or use your cursor to select the text you want to change.

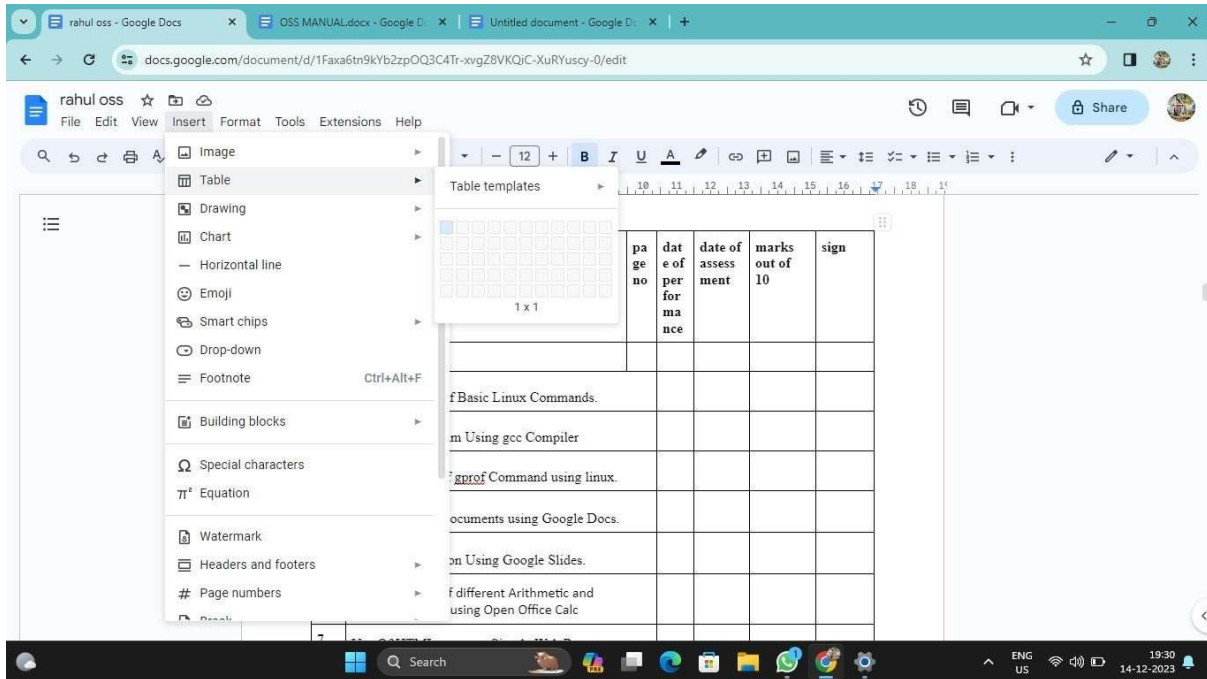
3.start editing.



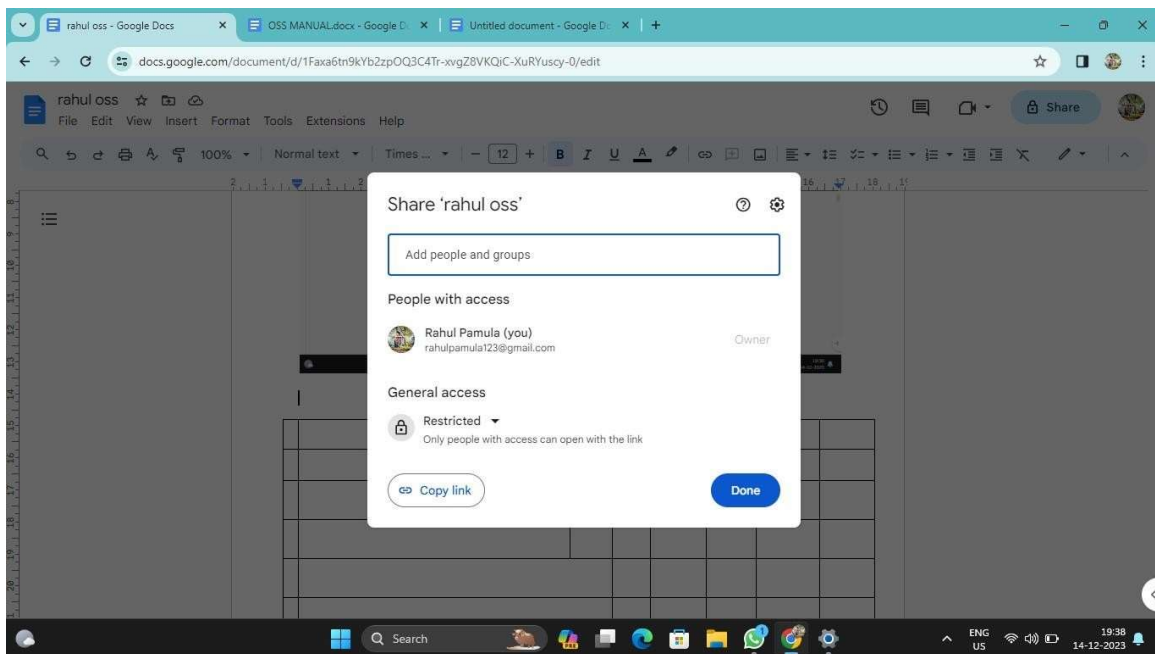
4.to undo or redo an action,at the top,click undo undo or redo redo



### Add a title, heading or table of contents



### step 3: share & work with others



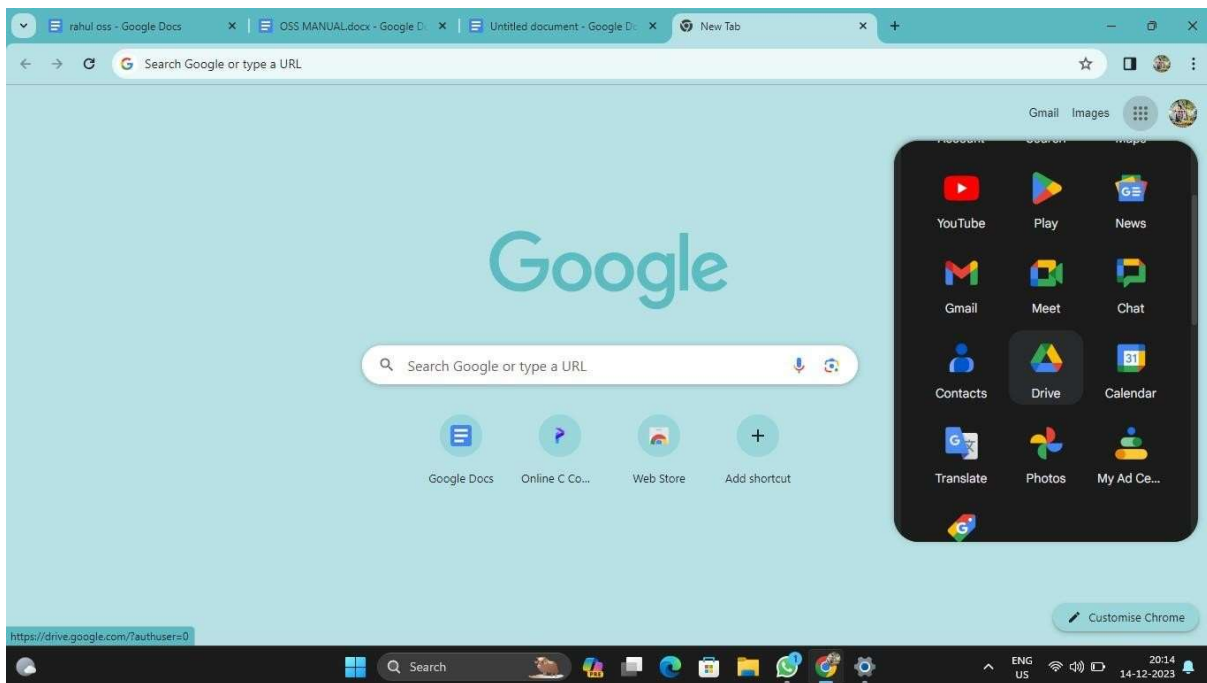
## Practical 5

**AIM: Create Presentation using Google Slides.**

**Go to Google's home page and click on the grid in the upper right hand corner.**

**click on the Drive icon. You will be redirected to a log in page if you are signed out, if not you will be taken to your Drive.**

**You can also just type in <https://slides.google.com>, log in if you haven't already, and will be taken to the Slides page.**

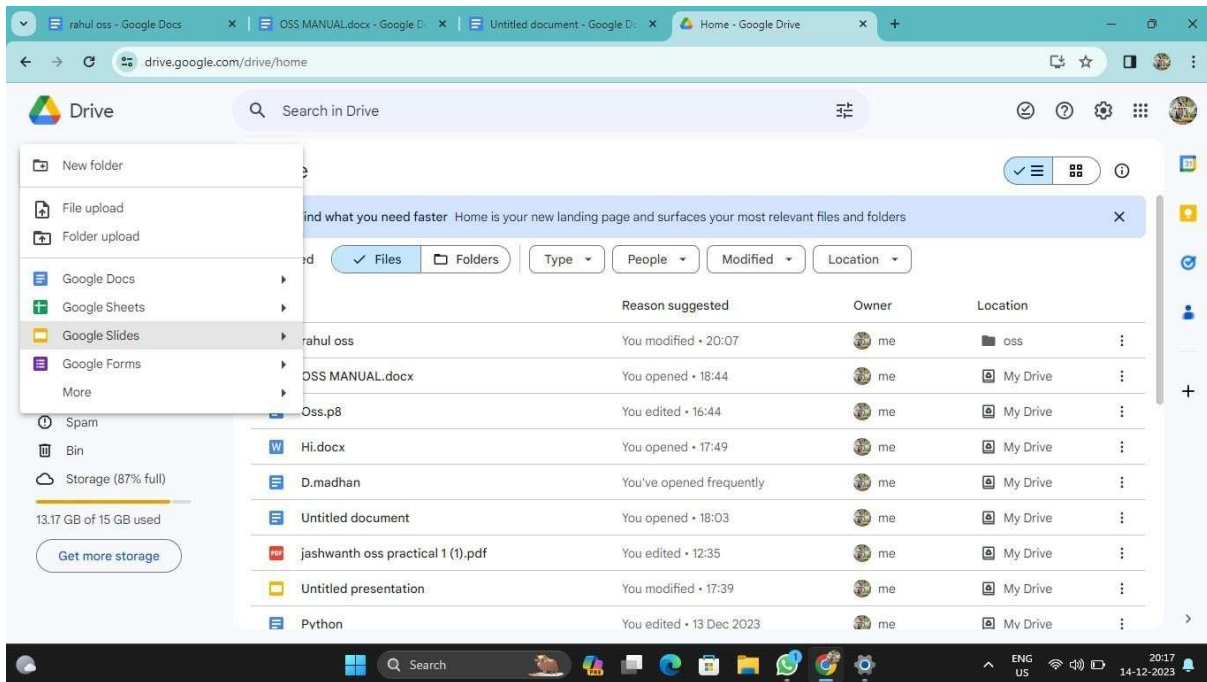


**From your Drive, click the blue New button on the left side of the page. Select "Google Slides"**

**from the drop-down menu.**

**For more options, hover over the arrow on the right edge of the Google Slides option, where a smaller drop-down menu will appear. From here you can select to create a presentation from a template or a blank slide**

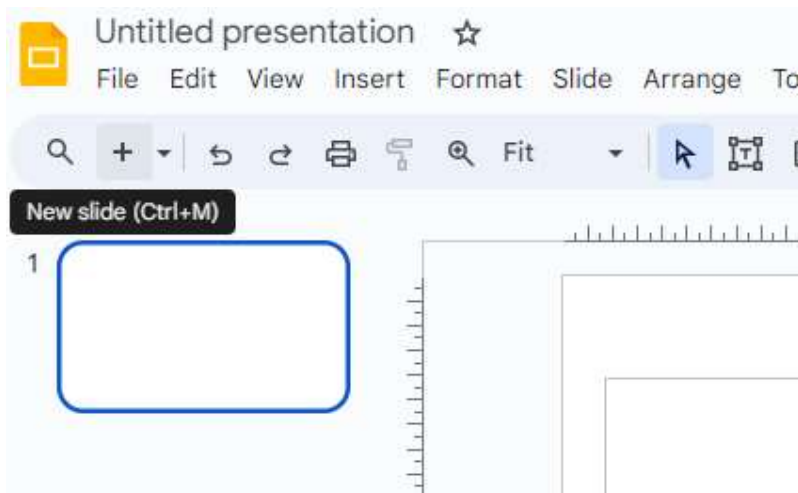




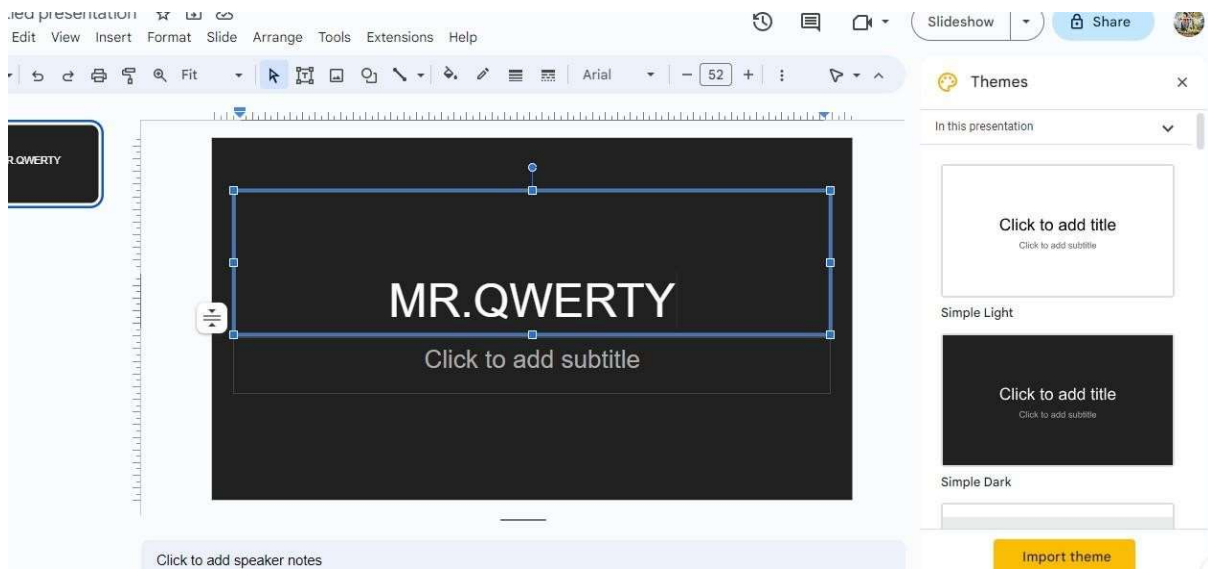
If you are on the slides page, select an option from the top of the page to create a new slide.

You can press the white square with a plus sign for a blank slide, or click one of the templates.

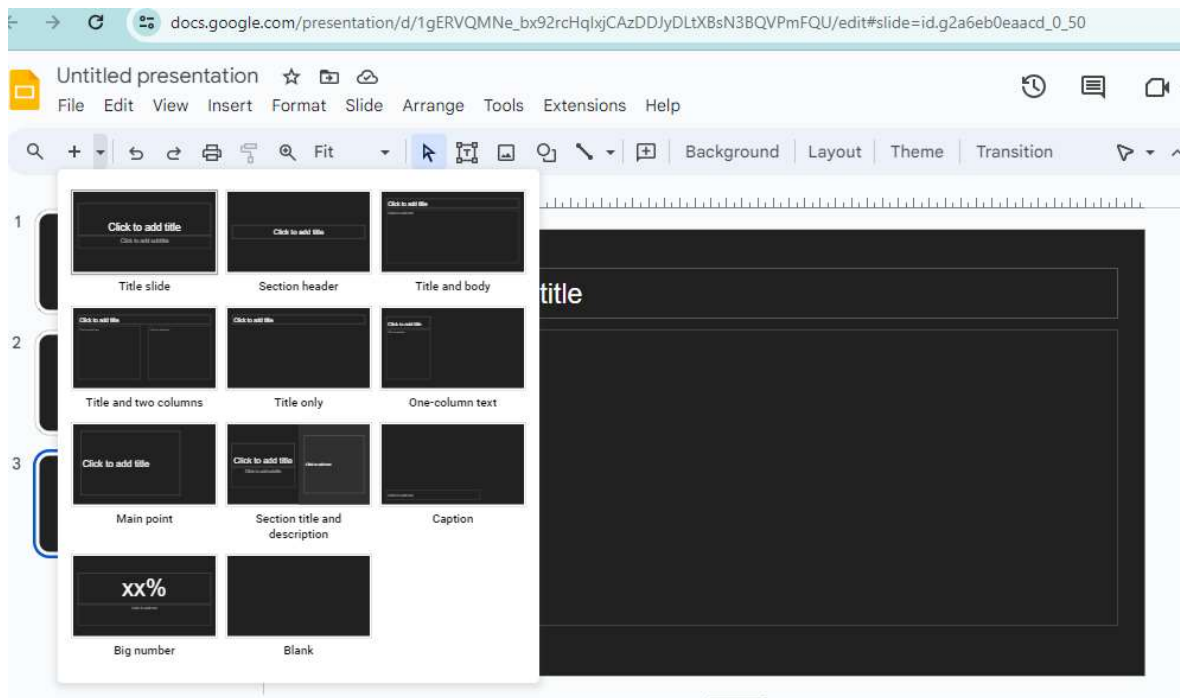
Click on the Template Gallery option, where more templates will show up.



Name your slide and select a theme.

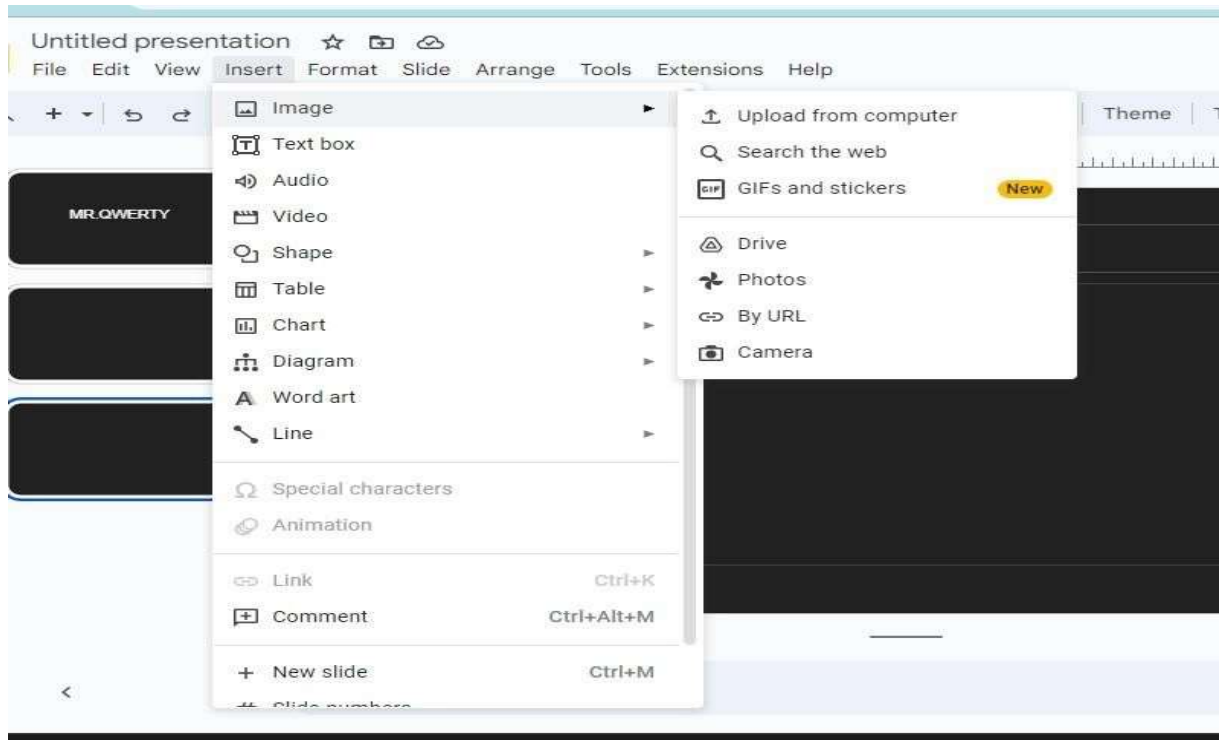


**Add new slides.**

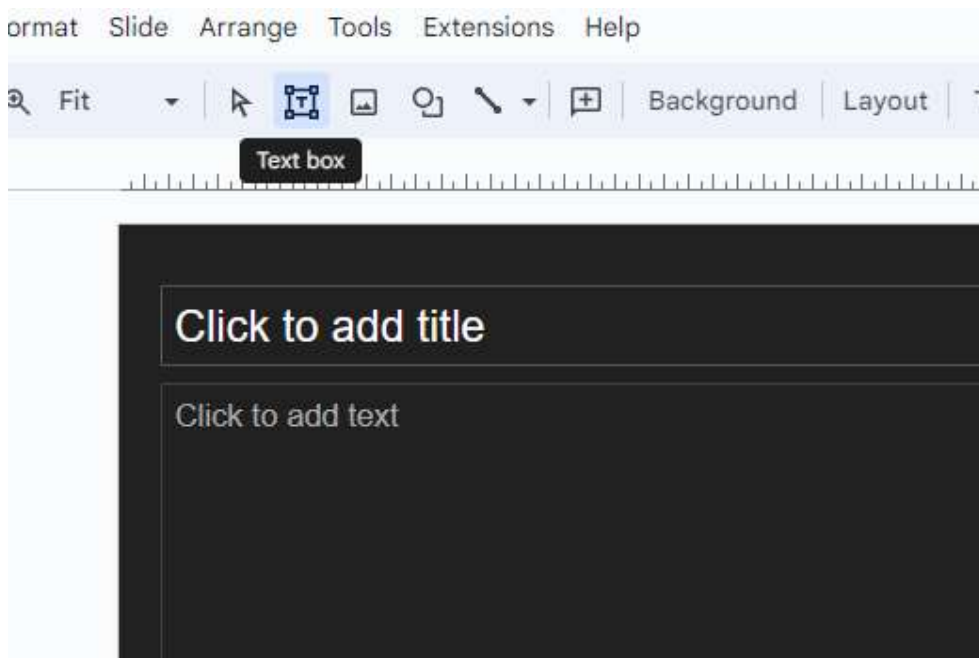




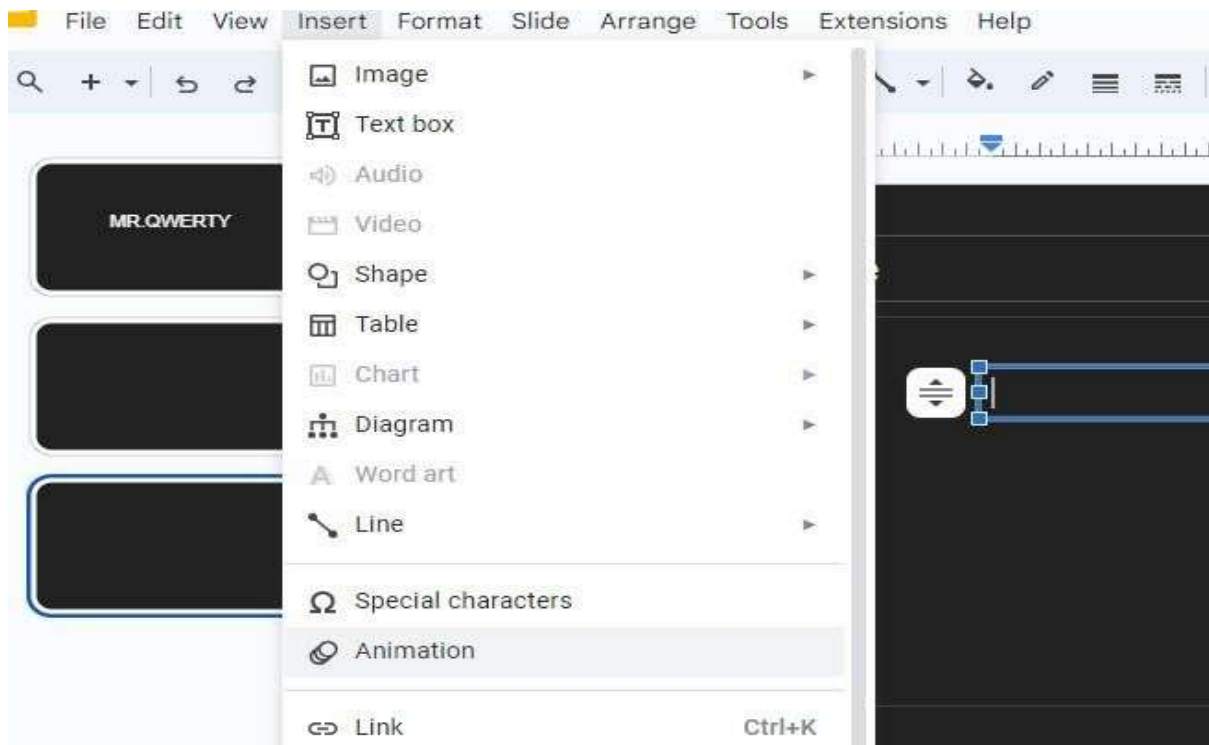
## Insert image



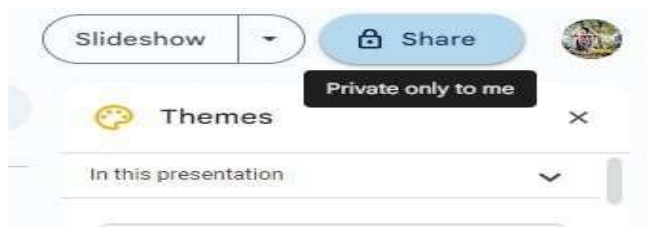
## Add text



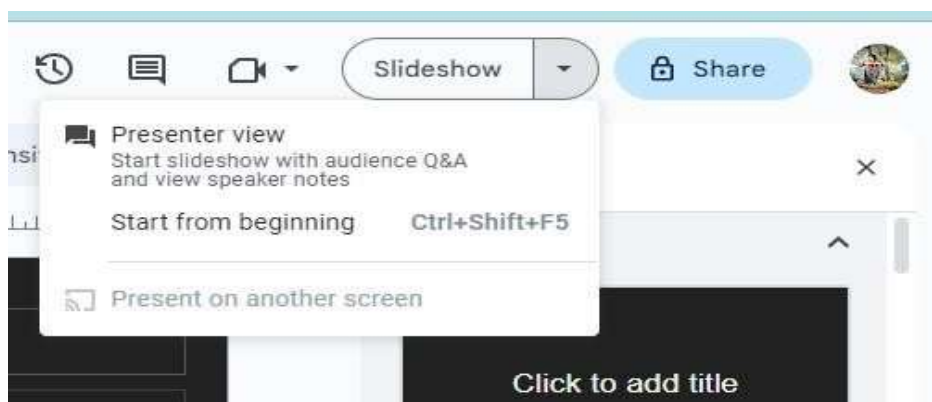
## Animate



When you are done editing, press the share button to edit permissions for your presentation.



View the finished product by clicking "Present" on the upper right hand corner.



## Practical 6

**AIM: Demonstration of different Arithmetic and Logical Formulas using OpenOffice Calc.**

OpenOffice Calc, the spreadsheet program offered free of charge by [openoffice.org](http://openoffice.org), allows you to perform calculations on data entered into the spreadsheet.

OpenOffice Calc formulas for basic number crunching, such as addition or subtraction, as well as more complex calculations such as payroll deductions or averaging a student's test results.

In addition, if you change the data, Calc will automatically recalculate the answer without you having to re-enter the formula.

The following step Create and use a basic formula in OpenOffice Calc.

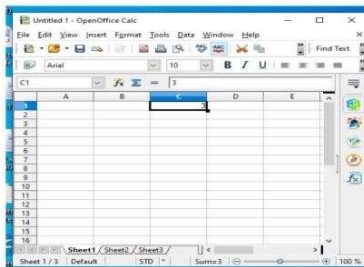


### Entering the Data

The steps used to create this formula are the same ones to follow when writing more complex formulas. The formula will add the numbers 3 + 2. The final formula will look like this:

$$= C1 + C2$$

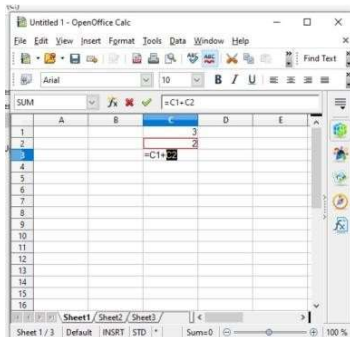
Select the cell C1 and enter 3, then press Enter.



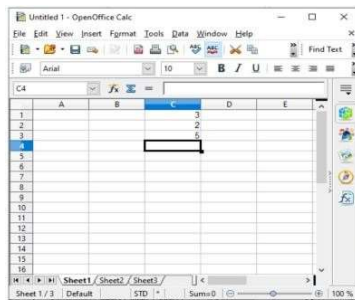
Select the cell C2 and enter 2, then press Enter.



Now select cell C3. This is where we'll enter the basic addition formula.



When creating formulas in Open Office Calc, you always start by typing the equals sign. Type it in the cell where you want the answer to appear.



Following the equals sign, we add in the cell references of the cells containing our data.

By using the cell references of our data in the formula, the formula will automatically update the answer if the data in cells C1 and C2 changes.

`=Sum(b2:g2)`

`=If(AND(b2>33,c2>33,d2>33...),”PASS”,”FAIL”)`

## Practical 7

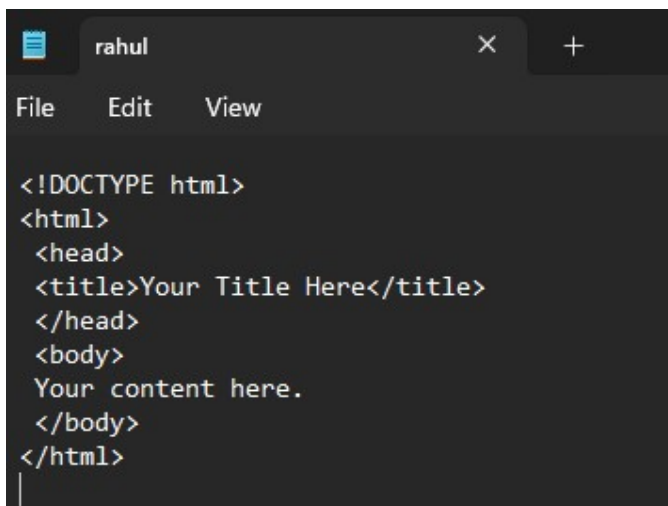
**AIM:** Use of HTML to create simple web page.

**Step 1:** Open a Text Editor (Notepad)

**Step 2:** Write Some HTML Code

We are now going to add the HTML boiler plate code. This is the code that will allow the browser

to correctly display your webpage.

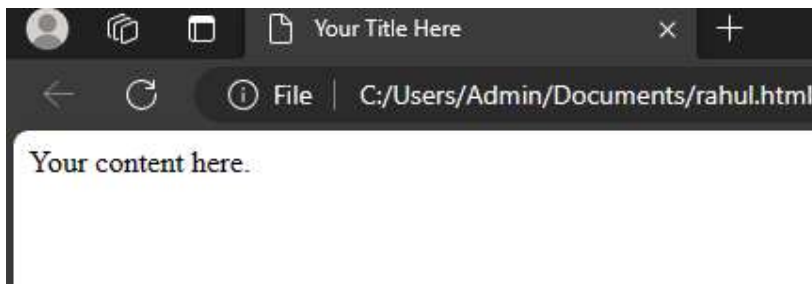


```
<!DOCTYPE html>
<html>
  <head>
    <title>Your Title Here</title>
  </head>
  <body>
    Your content here.
  </body>
</html>
```

Save the html page by pressing CTRL + S or click on file option then save option. Ensure that you

name the file in the following format: “name” then “.html” examples index.html, cooking.html.

this.

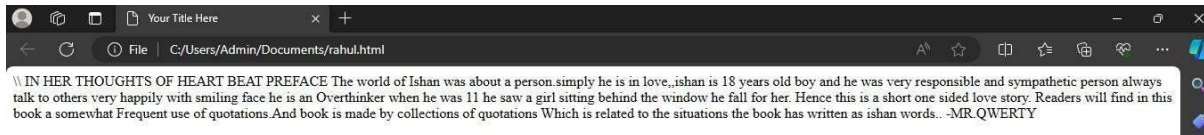


### Step 3: Add the Text Content

Add the content between the body tags.

Refresh the webpage on the browser. It will look like a blob of text with no paragraphs or headings

shown below.



### Step 4: Add the HTML Tags

#### Headings

First, we shall tackle the headings. Check which part or the text was the main heading. In our example it is “Insect eating: The ultimate guide to eating crickets”.

Place this text between h1 tags as shown below:

```
<H1>IN HER THOUGHTS OF HEART BEAT</H1> <H2>PREFACE</h2>
```

Save your work and check the result on the browser.



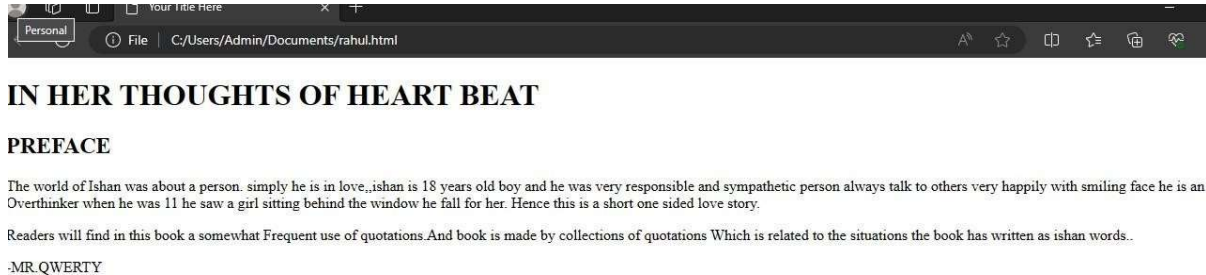
#### Paragraphs

For the paragraphs of text, place each paragraph of text between the <p></p> html tags.

Example:

```
<H1>IN HER THOUGHTS OF HEART BEAT</H1> <H2>PREFACE</h2>
<P> The world of Ishan was about a person.
simply he is in love,,ishan is 18 years old boy and he was very responsible and sympathetic person always talk to others
very happily with smiling face he is an Overthinker when he was 11 he saw a girl sitting behind the window he fall for her
Hence this is a short one sided love story.</P>
Readers will find in this book a somewhat Frequent use of quotations.And book is made by collections of quotations Whic
is related to the situations the book has written as ishan words..<P> -MR.QWERTY</P>
```

## view the result



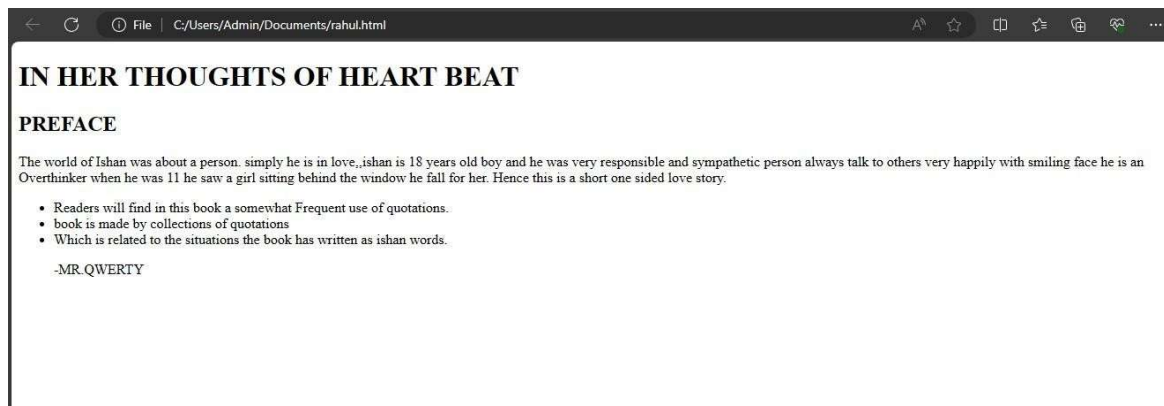
## Step 6: Add a List

Lists make reading a group of things easier on our eyes and brain. Lets add a bullet points type list:

```
Hence this is a short one sided love story.</P>
<ul>
  <li> Readers will find in this book a somewhat Frequent use of  quotations.</li><li> book is made by collections of
quotations</li><li> Which is related to the situations the book has written as ishan words.</li><P> -MR.QWERTY</P>

</body>
</html>
```

Refresh and view the result on the browser.





## Step 7: Add Images

We are going to use the HTML's img tag `` to add

your images on the webpage

```

```



## IN HER THOUGHTS OF HEART BEAT

### PREFACE

The world of Ishan was about a person, simply he is in love, ishan is 18 years old boy and he was very responsible and sympathetic person always talk to others very happily with smiling face he is an Overthinker when he was 11 he saw a girl sitting behind the window he fall for her. Hence this is a short one sided love story.

- Readers will find in this book a somewhat Frequent use of quotations.
- book is made by collections of quotations
- Which is related to the situations the book has written as ishan words.

-MR.QWERTY-

<https://drive.google.com/file/d/1CTKW78DSyVmnfOJcgt8R9qVZXXLj9EK9/view?usp=drivesdk>

## Step 8: Embed a Youtube Video

Search for relevant video on youtube. Once you have found it, click on share button/ link. You will get a popup. Click on embed option

```
width="310"
```

```
height="160"
```

```
src="https://www.youtube.com/embed/BwC4WRKi5QY"
```

```
frameborder="0"
```

```
allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture"
```

```
allowfullscreen
```

```
>>/iframe>
```





## Practical 8

**AIM:**demonstration of mathml a markup language for describing mathematical notation

### MathML

**MathML (Mathematical Markup Language)** is an XML-based markup language used to represent mathematical notations and formulas.

It is designed to be compatible with XML and HTML, making it suitable for displaying math on webpages or in XML-based documents.

Conceptually, MathML consists of two main strains of markup: Presentation markup is used to display mathematical expressions; and Content markup is used to convey mathematical meaning.

These two strains, along with other external representations, can be combined using parallel markup.

**Example:**

Here's a simple example of MathML that represents the equation of a line: Xml

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
```

```
<mrow>
```

```
mi>y</mi> <
```

```
mo>=</mo> <
```

```
mi>m</mi> <
```

```
mi>x</mi> <
```

```
mo>+</mo> <
```

```
mi>b</mi> <
```

```
</mrow>
```

```
</math>
```

**In this example:**

**<math>:** The root element that defines the MathML content.

**<mrow>:** Represents a row or sequence of mathematical expressions.

**<mi>:** Represents a mathematical identifier (in this case, variables).

**<mo>:** Represents an operator or symbol.

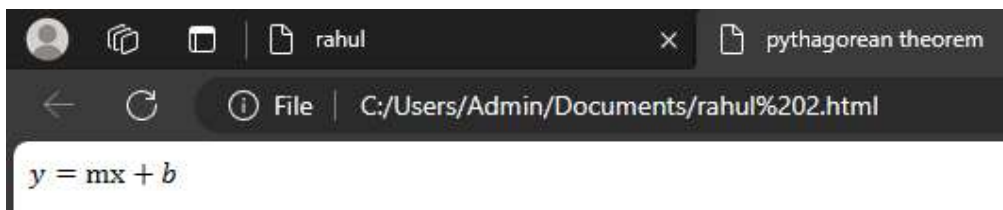
This MathML code represents the equation of a line in slope-intercept form ( $y = mx + b$ ), where "m" and "b" are constants and "x" and "y" are variables.

To view this MathML example, you can use a MathML-enabled browser like Firefox, or you can try an online MathML rendering tool.

Copy and paste the MathML code into one of these environments, and you should see the equation of a line displayed.

```
File Edit View
<!DOCTYPE html>
<html>
<head>
<title>pythagorean theorem </title>
</head>
<body>
<math>
<mrow>
<mi>y</mi>
<mo>=</mo>
<mi>mx</mi>
<mo>+</mo>
<mi>b</mi>
</math>
</body>
</html>
|
```

Please note that MathML support may vary in different environments, so make sure you're using a MathML-capable platform to view the example.



## Practical 9

**Aim: demonstration of virtualization using docker container**

### DOCKER

**Docker is a popular platform for virtualization using containerization.**

**It allows you to create, deploy, and manage lightweight, portable containers that run applications and their dependencies consistently across different environments**

**Docker containers include all dependencies (frameworks, libraries, etc.)**

**to run an application in an efficient and bug-free manner. Docker Containers have the following benefits:**

- 1. Light-weight**
- 2. Applications run in isolation**
- 3. Occupies less space**
- 4. Easily portable and highly secure**
- 5. Short boot-up time**

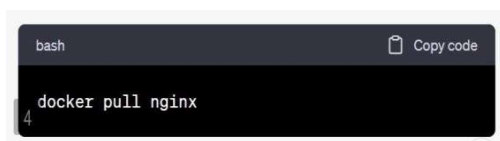
**step-by-step demonstration of virtualization using Docker**

**Step 1: Install Docker** First, ensure you have Docker installed on your system. You can download and install Docker Desktop for Windows or macOS from the official Docker website. For Linux users, you can follow the instructions for your specific distribution.

**Step 2: Pull a Docker Image** Docker images are the blueprints for creating containers. Let's start by pulling a simple image, like the official Nginx web

**server image:**

**Open a terminal (or command prompt) and run the following command to pull the Nginx image:**



```
bash Copy code
4 docker pull nginx
```

**Step 3: Run a Docker Container** Now that you have the Nginx image, you can create a container based on it. Run the following command to start an Nginx container:

```
bash Copy code
docker run -d -p 8080:80 nginx
```

This command tells Docker to run the Nginx container in the background (-d) , and it maps port 8080 of your host system to port 80 of the container (-p 8080:80).

**Step 4: Access the Container** You can access the Nginx web server running inside the Docker container by opening your web browser and navigating to

<http://localhost:8080>.

**Step 5: View Running Containers** To see the list of running containers, run:

```
bash Copy code
docker ps
```

**Step 6: Stop and Remove the Container** When you are done, you can stop the Nginx container by running:

```
bash Copy code
docker stop <CONTAINER_ID>
```

Replace <CONTAINER\_ID> with the actual ID of the running container (you can get the ID from the docker ps command).

To remove the container, run:

```
bash Copy code
docker rm <CONTAINER_ID>
```

**Step 7: Clean Up**

If you don't plan to use the Nginx image anymore, you can remove it

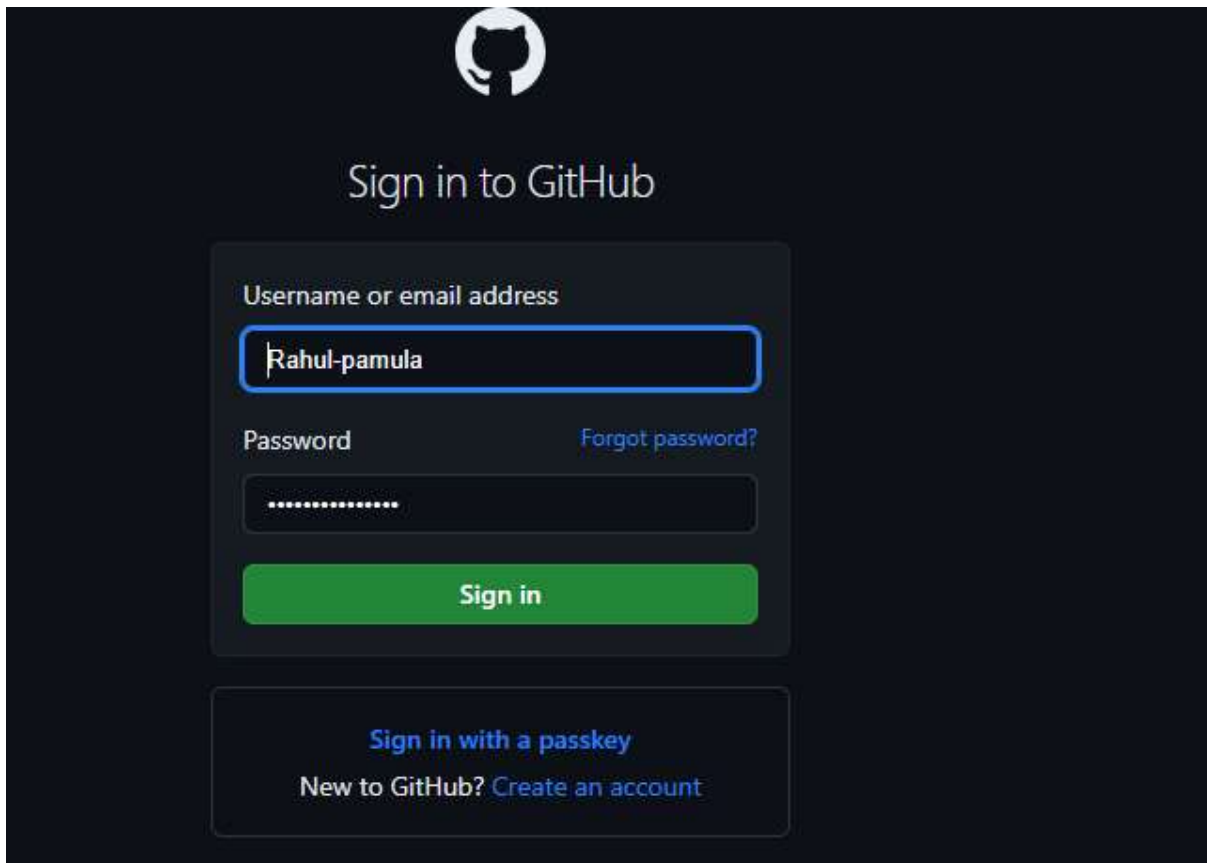
```
bash Copy code
docker rmi nginx
```

These are the basic steps to demonstrate virtualization using Docker containers. Docker offers much more functionality, such as creating custom images, managing volumes for persistent data, and orchestration using tools like Docker Compose or Kubernetes. It's a powerful tool for building, shipping, and running applications in a consistent and isolated environment.

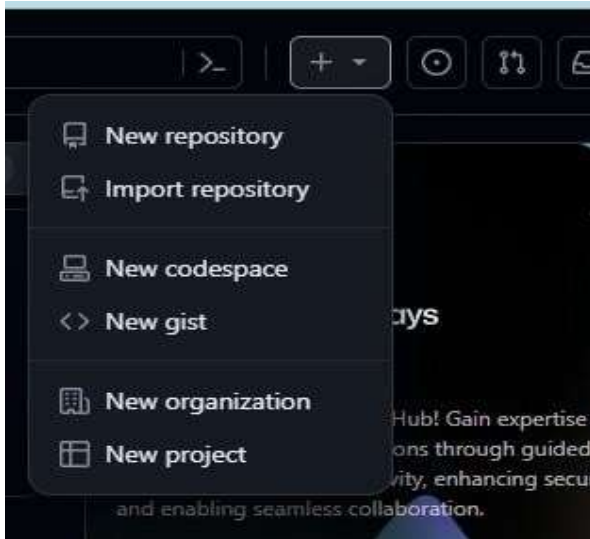
## Practical 10

GitHub is a popular web-based platform for hosting and collaborating on code repositories. It provides facilities for version control, issue tracking, code review, and team collaboration. Here's a demonstration of some basic GitHub facilities:

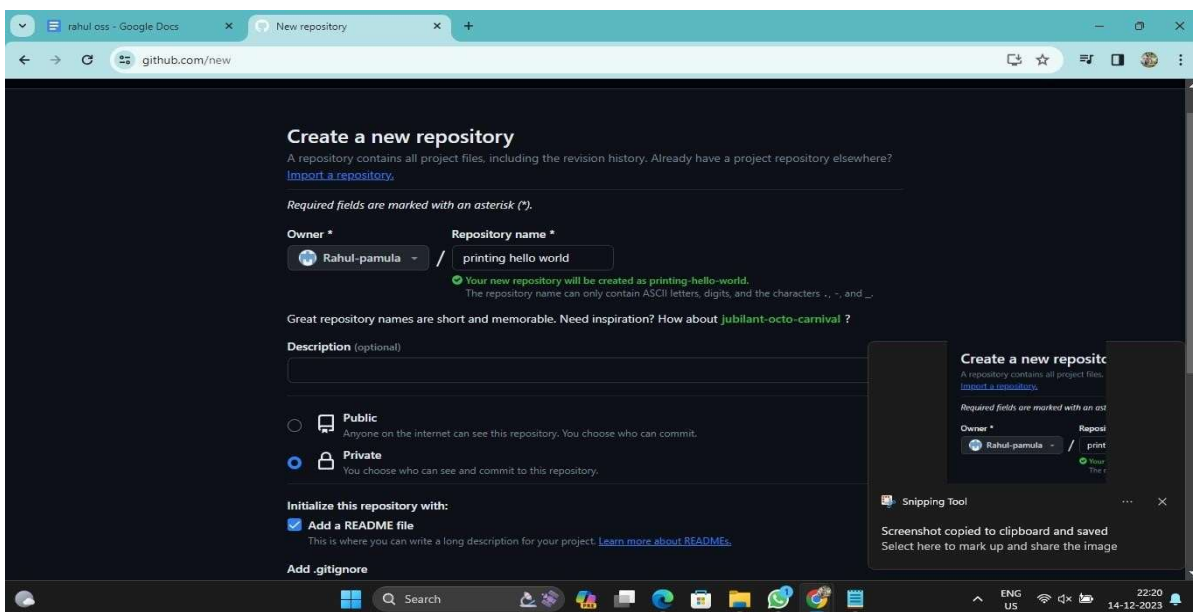
**Step 1: Create a GitHub Account** If you don't have a GitHub account, go to <https://github.com/> and sign up for a new account.



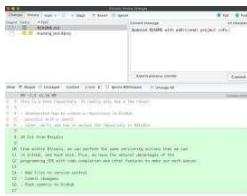
**Step 2: Create a New Repository** Once you have a GitHub account, log in, and click on the "+" sign in the top right corner. Then, select "New repository." Fill in the repository details, such as the repository name, description, visibility (public or private), and other options. Click "Create repository" when you're done.



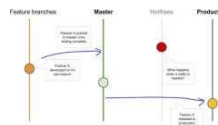
**Step 4 :Initializing the Repository:** After creating the repository, you have the option to initialize it with a README file, a .gitignore file (to specify which files to ignore), and an open-source license. The README file is useful for providing information about your project and instructions for anyone who visits the repository.



**Step 5:Adding Files:** You can demonstrate version control by adding files to your repository. Click the "Add file" button to create a new file or use the command-line interface to clone the repository to your local machine, make changes, and then push them back to the GitHub repository.



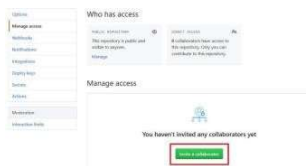
**Step 6: Branching and Pull Requests:** You can demonstrate the collaborative nature of GitHub by creating branches to work on specific features or fixes. When you're ready to merge your changes back into the main branch (usually "main" or "master"), you can create a pull request. Others can review the changes, provide feedback, and approve the merge.



**Step 7:Issues and Discussions:** GitHub also allows you to create issues to track tasks, bugs, or feature requests. You can use this feature to demonstrate how issues can be assigned to team members and resolved through code changes. Additionally, GitHub provides a "Discussions" feature where you can have more open-ended discussions with the community or contributors.



**Step 8:Collaboration:** For demonstration purposes, you can invite other GitHub users to collaborate on your repository. They can clone, branch, make changes, and push them back, just like you did earlier.



**Remember, GitHub is a powerful tool, and there are many features and workflows to explore. This basic outline should give you a starting point to demonstrate the essential functionalities of GitHub for version control and collaborative software development.**