

Laboratory Instructions

1. In addition to the instructions in this laboratory manual, follow all verbal and written instructions provided by the instructional staff.
2. Before coming to the laboratory, students should carefully read the sections of text which are covered in the assigned experiment. If the topic has been covered in lectures, the lecture notes should be studied as well.
3. Students should wear a valid ID card before entering the lab.
4. Use of mobile phone in the lab is strictly prohibited. Each student must keep mobile phones in “Switched Off” mode while entering and working in the Lab.
5. If any problem arises, please bring the same to the notice of lab in-charge.
6. Handle computers softly and gently.
7. In case of theft/destruction of the instruments in the lab, double cost of the lost/destroyed instrument will be charged from the student/user.
8. Lab Assistants are available to assist basic computer handling problems.
9. Before leaving the laboratory, shut down the computer.
10. It is mandatory to get checked the experiments write-ups in the next practical turn.

S. Y. B. Tech (ECE)

Trimester: VI

Subject: Linux Based Python Laboratory (CET2005A)

Name:

Class:

Roll No.:

Batch:

Experiment – 01

Title: Installation of Linux Operating System

Performed on:

Submitted on:

Marks	Teacher's Signature with date

Aim: To install Linux Operating System

Objective:

1. To learn the steps to install Linux operating system (virtual box)
2. To learn the features of Linux operating system

Theory:

What is Linux?

Linux is an open-source Operating System (OS).

An operating system is a software that manages all of the hardware resources associated with your desktop or laptop. The operating system manages the communication between the software and the hardware. The software wouldn't work without the operating system (OS).

The Linux operating system comprises several different components:

1. **Bootloader** – The software that manages the boot process of your computer.
2. **Kernel** – The kernel is the core of the system and manages the CPU, memory, and peripheral devices. The kernel is the lowest level of the OS.
3. **Init system** – This is a sub-system that bootstraps the user space and is charged with controlling daemons. One of the most widely used init systems is systemd? which also happens to be one of the most controversial. It is the init system that manages the boot process, once the initial booting is handed over from the bootloader (i.e., GRUB or GRand Unified Bootloader).
4. **Daemons** – These are background services (printing, sound, scheduling, etc.) that either start up during boot or after you log into the desktop.
5. **Graphical server** – This is the sub-system that displays the graphics on your monitor. It is commonly referred to as the X server or just X.

6. **Desktop environment** – This is the piece that the users actually interact with. There are many desktop environments to choose from (GNOME, Cinnamon, Mate, Pantheon, Enlightenment, KDE, Xfce, etc.). Each desktop environment includes built-in applications (such as file managers, configuration tools, web browsers, and games).
7. **Applications** – Desktop environments do not offer the full array of apps. Just like Windows and macOS, Linux offers thousands upon thousands of high-quality software titles that can be easily found and installed. Most modern Linux distributions include App Store-like tools that centralize and simplify application installation. For example, Ubuntu Linux has the Ubuntu Software Center which allows you to quickly search among the thousands of apps and install them from one centralized location.

Why use Linux?

Linux combine the reliability with zero cost of entry and one has the perfect solution for a desktop platform. Linux can be installed on any number of computers without paying a cent for software or server licensing.

Linux is generally far less vulnerable to attacks like ransomware, malware, or viruses. Server reboots are only necessary if the kernel is updated. Linux server run for years without being rebooted. If you follow the regular recommended updates, stability and dependability are practically assured.

Open source

Linux is also distributed under an open-source license. Open source follows these key tenants:

- The freedom to run the program, for any purpose.
- The freedom to study how the program works, and change it to make it do what you wish.
- The freedom to redistribute copies so you can help your neighbor.
- The freedom to distribute copies of your modified versions to others.

What is a “distribution?”

Linux has a number of different versions to suit any type of user. From new users to hard-core users, you’ll find a “flavor” of Linux to match your needs. These versions are called distributions (or, in the short form, “distros”). Nearly every distribution of Linux can be downloaded for free, burned onto disk (or USB thumb drive), and installed (on as many machines as you like).

Popular Linux distributions include:

- LINUX MINT
- MANJARO
- DEBIAN

- UBUNTU
- ANTERGOS
- SOLUS
- FEDORA
- ELEMENTARY OS
- OPENSUSE

And the server distributions are:

1. Red Hat Enterprise Linux
2. Ubuntu Server
3. Centos
4. SUSE Enterprise Linux

Features of Linux Operating System:

1. Portable Environment

Linux software operates flawlessly on a variety of hardware platforms. Without the worry of incompatibility, individuals can use Linux operating system on any device. It runs the same way on both high-end and low-end hardware.

2. Free and Open-Source

Its source code is available for anybody to use and alter. Many developers collaborate in organizations to improve and strengthen Linux, and lots of developers constantly work on updating the Linux system.

3. Shell/ Command-line Interface

The Linux system includes essential programs that users can utilize in order to issue commands to the operating system for executing the design flawlessly. You may also direct it to carry out various forms of commands for effectively carrying out the applications.

4. End-to-end encryption/ Security

Authentication can help you keep your data protected. Before you may access some critical files, the Linux Operating System requires you to enter a password. Furthermore, the Linux environment allows users to encrypt their data.

5. Graphical User Interface (GUI)

Linux Operating System comes with Graphical User Interface (GUI) abilities in the same way you can with Windows. Similarly, users can install the programs, and the computer graphics will begin to work in the same way that Windows does.

6. Keyboard Support

Because Linux is available in various languages, it is simple to use it worldwide. As a result, you can change the language on your keyboard as per your preference.

7. **Multitasking:** More than one function can be performed simultaneously by dividing the CPU time intelligently.
8. **Live CD/USB:** Almost all Linux distros provide live CD/USB so that users can run/try it without installing it.
9. **Graphical User Interface (X Window system):** Linux is command line-based OS but it can be converted to GUI based by installing packages.
10. **Support's customized keyboard:** As it is used worldwide, hence supports different languages keyboards.
11. **Application support:** It has its own software repository from where users can download and install many applications.
12. **File System:** Provides hierarchical file system in which files and directories are arranged.
13. **Open Source:** Linux code is freely available to all and is a community-based development project

Installing Linux Operating System

We can install Linux OS in the three following ways-

1. Install Ubuntu inside a virtual Box in windows
2. Dual boot Ubuntu with windows.
3. Replace windows and install Ubuntu

Install Ubuntu inside a VirtualBox in windows

[VirtualBox](#) is free and open-source virtualization software from Oracle. It enables you to install other operating systems in virtual machines. system should have at least 4GB of RAM to get good performance from the virtual operating system.

Requirements

- Good internet connection to download software and Linux ISO.
- Windows system with at least 40 GB of free space.
- Windows system with 4GB of RAM.
- Make sure to enable virtualization in the BIOS

Steps for installation of Linux in Virtual Box

Step 1: Download and install VirtualBox

<https://www.virtualbox.org/wiki/Downloads>

Step 2: Download the Linux ISO

<https://releases.ubuntu.com/18.04/>

Step 3: Install Linux using VirtualBox

For Further steps click on the below icon-link and follow the steps given in the presentation-



Output: VirtualBox with Linux OS

Conclusion:

Post Lab Questions:

1. Discuss the various features of Linux.
2. List the different OS for PC and compare the Linux OS with Windows OS.

S. Y. B. Tech (ECE)

Trimester: VI

Subject: Linux Based Python Laboratory (CET2005A)

Name:

Class:

Roll No.:

Batch:

Experiment – 02

Title: Execution of Basic Linux Commands

Performed on:

Submitted on:

Marks	Teacher's Signature with date

Aim: To execute basic Linux commands

Objective:

3. To know the basic Linux commands.
4. To execute the basic Linux command from Linux terminal.

Theory:

The Linux command is a utility of the Linux operating system. All basic and advanced tasks can be done by executing commands. The commands are executed on the **Linux terminal**. The terminal is a command-line interface to interact with the system, which is similar to the command prompt in the Windows OS. *Commands in Linux are case-sensitive.*

Linux provides a powerful command-line interface compared to other operating systems such as **Windows** and **MacOS**. Some basic tasks such as creating a file, deleting a file, moving a file, and more can be easily implemented using basic Linux commands. In addition, some advanced tasks such as administrative tasks (including package installation, user management), networking tasks (ssh connection), security tasks, and many more can also be performed.

Linux terminal is a user-friendly terminal as it provides various support options. To open the Linux terminal, one has to press "**CTRL + ALT + T**" keys together, and execute a command by pressing the '**ENTER**' key.

The commonly used Linux commands are explained as below-

1. man

Man stands for manual which is a reference book of a [Linux operating system](#). It is similar to HELP file found in popular software. MAN command is used to get help on any command.

Syntax: man <Linux command>

Ex: man man, man mkdir etc.

2. pwd

pwd command is used to display the location of current working directory.

Ex:

```
[root@localhost ~]# pwd
/root
[root@localhost ~]#
```

3. mkdir Command

The **mkdir** command is used to create a new directory under any directory.

Syntax:

mkdir <directory name>

Ex:

```
[root@localhost ~]# mkdir test
[root@localhost ~]# ls
bench.py  hello.c  test
[root@localhost ~]#
```

4. ls Command

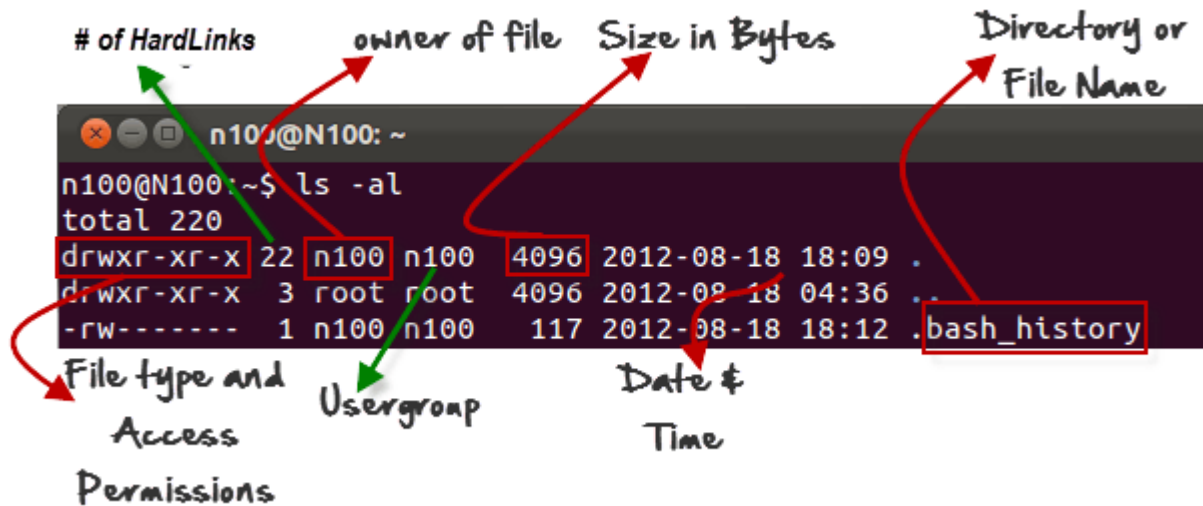
The **ls** command is used to display a list of content of a directory. It shows the files /directories in your current directory.

- Directories are denoted in blue color.
- Files are denoted in white.
- You will find similar color schemes in different flavors of Linux.

‘ls -R’ to shows all the files not only in directories but also subdirectories

‘ls -al’ gives detailed information of the files like the permissions, size, owner, etc.

Ex:



The first '-' implies that we have selected a file.

```
-rw-rw-r--
```

indicates
file

If it were a directory, **d** would have been shown.

d represents directory

```
drwxr-xr-x 2 ubuntu ubuntu 80 Sep 6 07:27 Desktop
```

The other characters are interpreted as-

r =	read	permission
w =	write	permission
x =	execute	permission
- =	no permission	

Ex:

The first part of the code is '**rw-**'. This suggests that the owner 'Home' can:

- Read the file
- Write or edit the file
- He cannot execute the file since the execute bit is set to '-'.

```
-rw-rw-r--
```

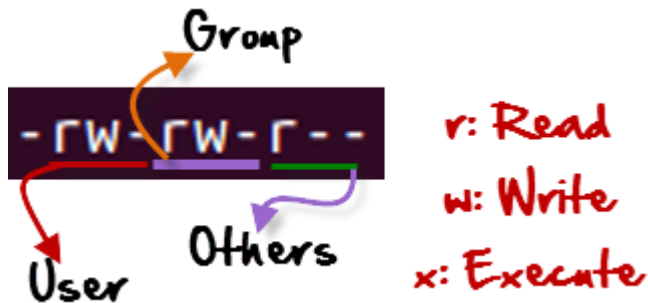
no execute
permission

The second part is '**rw-**'. It for the user group 'Home' and group-members can:

- Read the file
- Write or edit the file

The third part is for the world which means any user. It says 'r -'. This means the user can only:

- Read the file



Listing Hidden Files

Hidden items in UNIX/Linux begin with a '.' (period symbol) at the start of the file or directory.

Any directory/file starting with a '.' (period symbol) will not be seen unless you request for it. To view hidden files, use the command.

ls -a

ls has many other options as follows

- -l long list (displays lots of info)
- -t sort by modification time
- -S sort by size
- -h list file sizes in human readable format
- -r reverse the order

Options can also be combined as: "ls -ltr"

5. rmdir Command

The rmdir command is used to delete a directory.

Syntax:

rmdir <directory name>

Ex:

```
[root@localhost ~]# mkdir test
[root@localhost ~]# ls
bench.py  hello.c  test
[root@localhost ~]# ls -l
total 12
-rw-r--r-- 1 root root 114 Dec 26 15:49 bench.py
-rw-r--r-- 1 root root 185 Sep  9 2018 hello.c
drwxr-xr-x 2 root root  37 Jun 16 16:05 test
[root@localhost ~]# rmdir test
[root@localhost ~]#
```

6. cd Command

The `cd` command is used to change the current directory.

Syntax:

`cd <directory name>`

```
[root@localhost ~]# mkdir test
[root@localhost ~]# cd test
[root@localhost test]#
```

The `cd ..` command is used to switch to the parent directory.

The `cd ~` command is used to jump to root directory from any current working directory.

Ex.:

```
[root@localhost newfolder]# cd..
sh: cd..: command not found
[root@localhost newfolder]# cd ..
[root@localhost test]# cd newfolder
[root@localhost newfolder]# cd ~
[root@localhost ~]#
```

Creating files in Linux:

To create and edit a file in Linux requires an editor. The various editors that can be used with linux as follows-

- gedit
- nano/pico
- vi
- emacs

To create / Edit the file using “nano” or “vi” type “nano” followed by the filename on command prompt that opens the following editor.



- Edit the file (add content to the file)
- To save the file use “ctrl+O” and press enter
- To exit from the editor, press “ctrl+X”

Displaying file contents:

cat command is used to dump an entire file to standard output. It is good for displaying short, simple files.

Ex:

```
[root@localhost ~]# cat firstpyprogram.py
a=int(input('Enter a number'))
b=int(input('Enter a second number'))
c=a+b
print("addition of numbers=",c)

[root@localhost ~]#
```

- head - displays the top part of a file, by default 10 lines
- tail - displays the bottom part of a file, by default 10 lines

File Commands used to perform various operations on file are as follows:

- To copy a file: cp
- To move or rename a file: mv
- To remove a file: rm
- To change file access permissions: chmod

Examples:

1. cp command

```
[root@localhost ~]# cp test.py test1.py
[root@localhost ~]# ls
bench.py hello.c test test1.py test.py
[root@localhost ~]#
```

2. mv command

2.1 mv used to move the file

A new directory named 'abc' is created and moved the test from current directory to 'abc' directory. Changed directory to 'abc' and listed the content of directory that shows the moved file name.

```
[root@localhost ~]# cp test.py test1.py
[root@localhost ~]# ls
bench.py hello.c test test1.py test.py
[root@localhost ~]# mkdir abc
[root@localhost ~]# mv test.py abc
[root@localhost ~]# cd abc
[root@localhost abc]# ls
test.py
[root@localhost abc]#
```

2.2 mv can also be used to rename a file

```
[root@localhost ~]# ls
abc bench.py hello.c test test1.py
[root@localhost ~]# mv test1.py test2.py
[root@localhost ~]# ls
abc bench.py hello.c test test2.py
[root@localhost ~]#
```

3. rm command

To remove a file “recursively” the command used is **rm -r**. It is used to remove all files and directories. The deletions are permanent in Linux.

```
[root@localhost abc]# rm test.py
[root@localhost abc]# ls
[root@localhost abc]#
```

4. chmod command

one can change file permissions (read, write, execute) on a file/directory for the owner, group and the world using “chmod”.

Syntax:

chmod [user/group/others/all]+[permission] [file(s)]

```
[root@localhost ~]# ls -l
total 20
drwxr-xr-x 2 root root 37 Jun 16 17:10 abc
-rw-r--r-- 1 root root 114 Dec 26 15:49 bench.py
-rw-r--r-- 1 root root 185 Sep 9 2018 hello.c
drwxr-xr-x 2 root root 37 Jun 16 16:36 test
-rw-r--r-- 1 root root 20 Jun 16 17:08 test2.py
[root@localhost ~]# chmod g+x test2.py
[root@localhost ~]# ls -l
total 20
drwxr-xr-x 2 root root 37 Jun 16 17:10 abc
-rw-r--r-- 1 root root 114 Dec 26 15:49 bench.py
-rw-r--r-- 1 root root 185 Sep 9 2018 hello.c
drwxr-xr-x 2 root root 37 Jun 16 16:36 test
-rw-r-xr-- 1 root root 20 Jun 16 17:08 test2.py
[root@localhost ~]#
```

Steps to create a python file, execute a file, download/upload a file, view a file:

1. Create a python file

```
[root@localhost ~]# nano hello.py
```

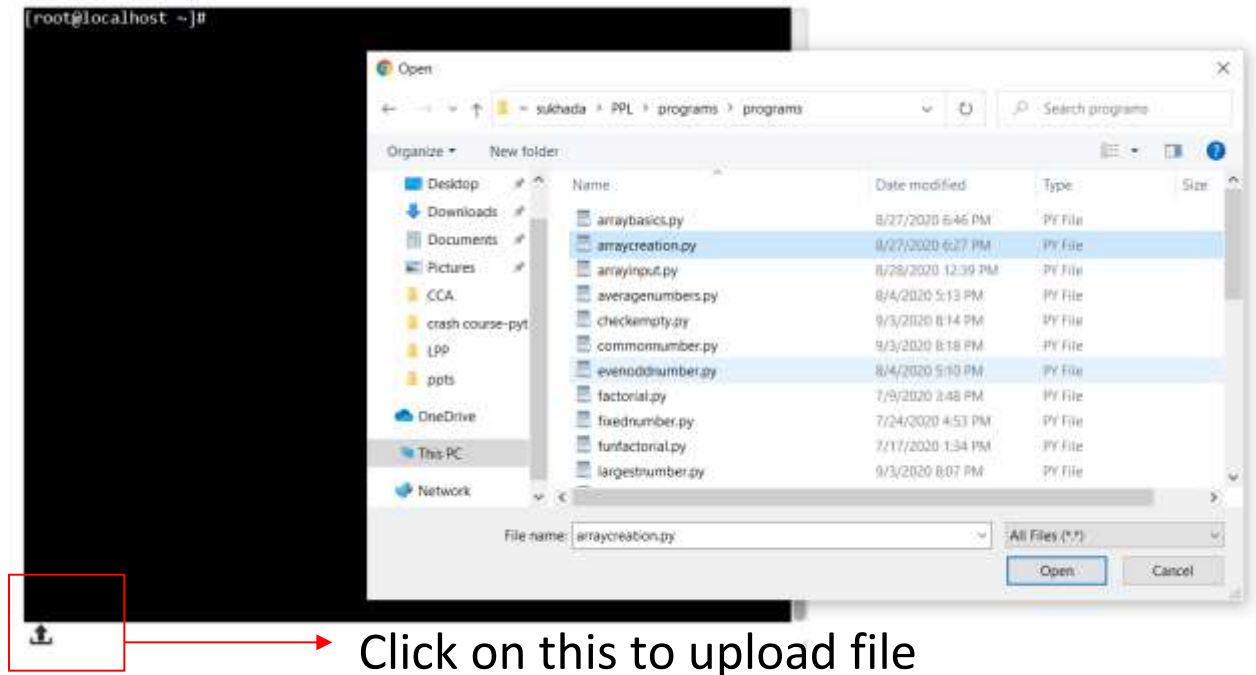
2. Execute a python file

```
[root@localhost ~]# python hello.py
Hello World
[root@localhost ~]#
```

3. Download a python file

```
[root@localhost ~]# export_file hello.py
[root@localhost ~]#
```

4. Upload a file



5. Uploaded file can be viewed using ls command

Input: Linux Commands

Output: Output of each command

Conclusion:

Post Lab Questions:

3. What are basic components of Linux?
4. How the permissions are granted in Linux?
5. Explain various options used using ls command.

Additional Reference Links:

1. <https://www.guru99.com/must-know-linux-commands.html>
2. <https://www.javatpoint.com/linux-commands>
3. <https://www.softwaretestinghelp.com/linux-interview-questions-answers/>

S. Y. B. Tech (ECE)

Trimester: VI

Subject: Linux Based Python Laboratory (CET2005A)

Name:

Class:

Roll No.:

Batch:

Experiment – 03

Title: Introduction to Fundamentals of Python

Performed on:

Submitted on:

Marks	Teacher's Signature with date

Aim: Introduction to Fundamentals of Python

Objective:

5. To know the Fundamentals of Python.
6. To implement the basic Python programs.

Theory:

Python has the following pedagogical benefits:

- Python has simple, conventional syntax. Python statements are very close to those of pseudocode algorithms, and Python expressions use the conventional notation found in algebra. Thus, you can spend less time dealing with the syntax of a programming language and more time learning to solve interesting problems.
- Python has safe semantics. Any expression or statement whose meaning violates the definition of the language produces an error message.
- Python scales well. It is easy for beginners to write simple programs in Python. Python also includes all the advanced features of a modern programming language, such as support for data structures and object-oriented software development, for use when they become necessary.
- Python is highly interactive. You can enter expressions and statements at an interpreter's prompts to try out experimental code and receive immediate feedback. You can also compose longer code segments and save them in script files to be loaded and run as modules or standalone applications.
- Python is general purpose. In today's context, this means that the language includes resources for contemporary applications, including media computing and web services.

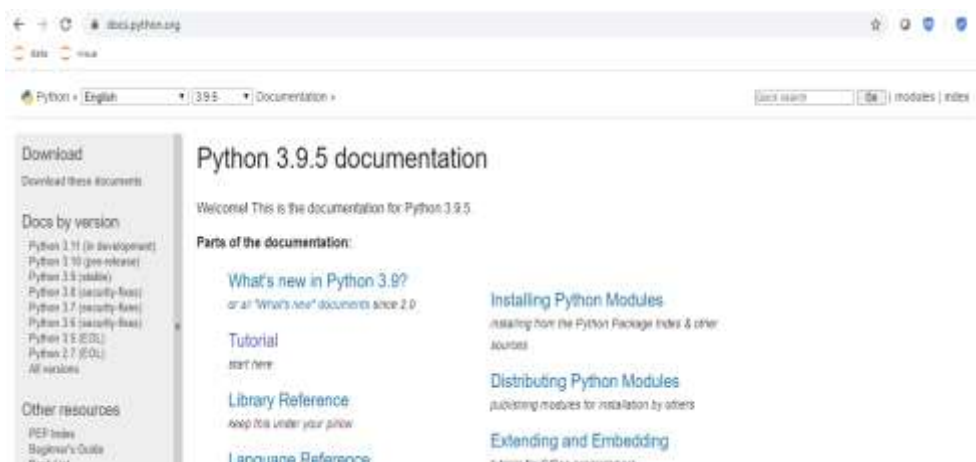
- Python is free and is in widespread use in the industry. You can download Python to run on a variety of devices. There is a large Python user community, and expertise in Python programming has great resume value.

To summarize these benefits, Python is a comfortable and flexible vehicle for expressing ideas about computation, both for beginners and for experts.

Python Applications:

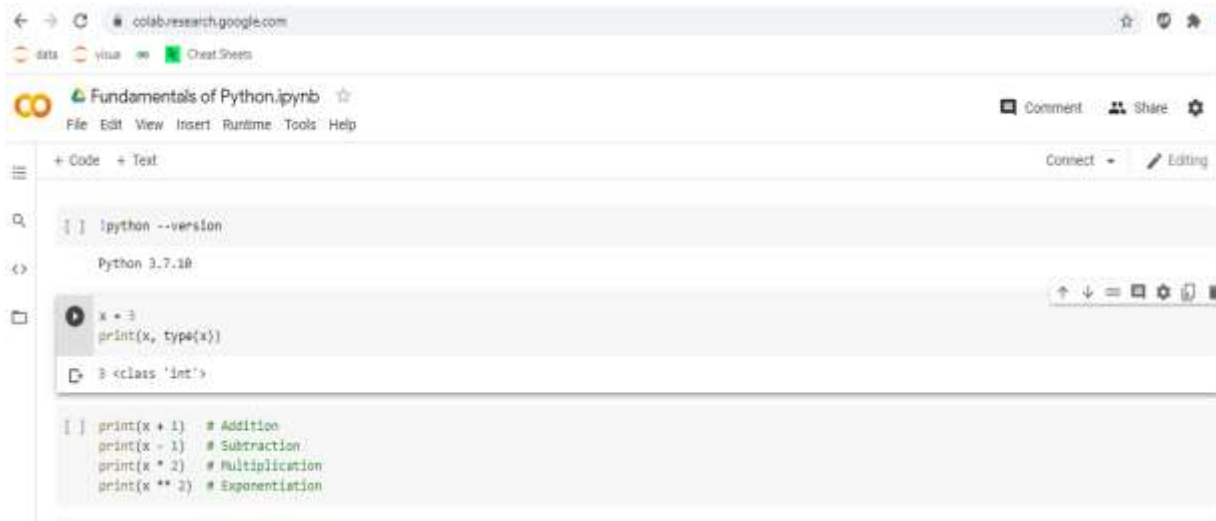
- Data Science
- Data Mining
- Desktop Applications
- Console-based Applications
- Mobile Applications
- Software Development
- Artificial Intelligence
- Web Applications
- Enterprise Applications
- 3D CAD Applications
- Machine Learning
- Computer Vision or Image Processing Applications.
- Speech Recognitions

Python Documentation:



Execution of Python code using Google Colaboratory:

- Colaboratory, or “Colab” for short, is a product from Google Research.
- Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.
- More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources
- including GPUs.

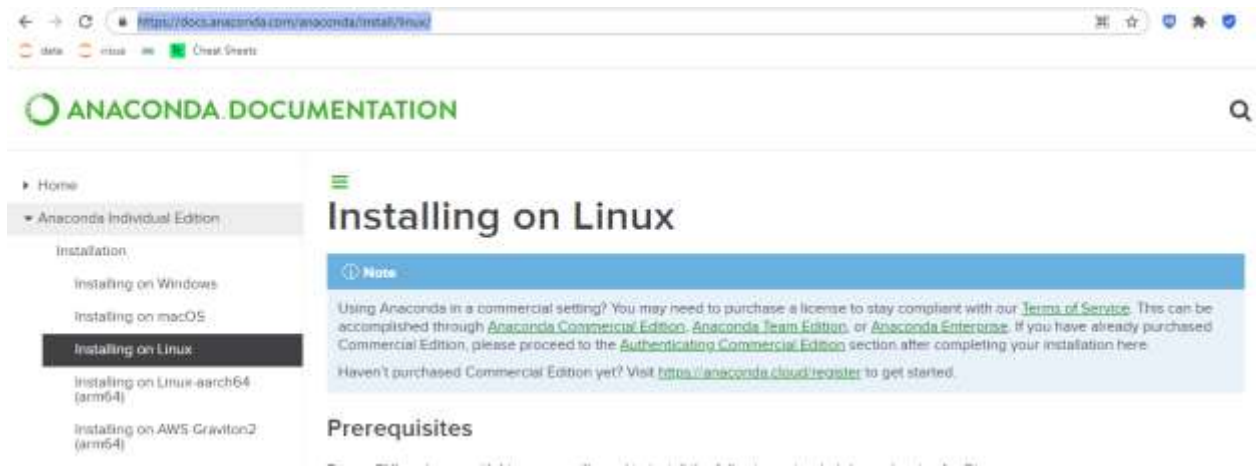


Execution of Python code using Anaconda Navigator:

- Anaconda Navigator is a desktop GUI that comes with Anaconda Individual Edition.
- It makes it easy to launch applications and manage packages and environments without using command-line commands

Install Anaconda by following steps given in the link below:

https://docs.google.com/document/d/16eQKQBbaI_XIEHqHV9N7UKOEaMEQYpRoXOykYU-ZJQA/edit?usp=sharing



Python Data Types:

Text Type:	str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Mapping Type:	dict
Set Types:	set, frozenset
Boolean Type:	bool
Binary Types:	bytes, bytearray, memoryview

Basic Python Commands:

Checking Version:

```
!python --version
```

Printing:

```
x = 4
print(x, type(x))
```

Arithmetic Operations:

```
x=5
print(x + 1) # Addition
print(x - 1) # Subtraction
print(x * 2) # Multiplication
print(x ** 2) # Exponentiation
```

String:

```
hello = 'hello' # String literals can use single quotes
world = "world" # or double quotes; it does not matter
print(hello, len(hello))
```

Input: Python Commands

Output: Output of each command

Conclusion:

Post Lab Questions:

6. What are basic Python program elements?
7. Write a Python Program to Swap Two Variables.
8. Write a Python Program using if-elif-else.
9. Write a Python Program to Find the Sum of Natural Numbers

Additional Reference Links:

4. <https://www.python.org>
5. <https://www.tutorialspoint.com>
6. <https://www.programiz.com/python-programming>

S. Y. B. Tech (ECE)

Trimester: VI

Subject: Linux Based Python Laboratory (CET2005A)

Name:

Class:

Roll No.:

Batch:

Experiment – 04

Title: Introduction to Basic Data Structures of Python

Performed on:

Submitted on:

Marks

Teacher's Signature with date

Aim: Introduction to Basic Data Structures of Python.

Objective:

7. To know the Basic Data Structures of Python.
8. To perform different operations on List and Set data structure.

Theory:

The Data Structures in the Python Programming Language and how they are related to some specific Python Data Types. We will discuss all the in-built data structures like list, set, tuples, dictionaries, etc.

Lists:

Python Lists are just like the arrays, declared in other languages which is an ordered collection of data. It is very flexible as the items in a list do not need to be of the same type.

The implementation of Python List is similar to Vectors in C++ or ArrayList in JAVA. The costly operation is inserting or deleting the element from the beginning of the List as all the elements are needed to be shifted. Insertion and deletion at the end of the list can also become costly in the case where the preallocated memory becomes full.

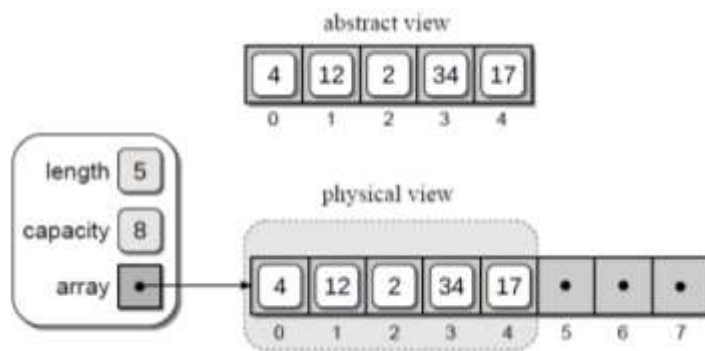
Python's list structure is a mutable sequence container that can change size as items are added or removed. It is an abstract data type that is implemented using an array structure to store the items contained in the list.

Creating a Python List:

Suppose we create a list containing several values:

```
pyList = [ 4, 12, 2, 34, 17 ]
```

Figure 2.2 illustrates the abstract and physical views of our sample list. In the physical view, the elements of the array structure used to store the actual contents of the list are enclosed inside the dashed gray box. The elements with null references shown outside the dashed gray box are the remaining elements of the underlying array structure that are still available for use. This notation will be used throughout the section to illustrate the contents of the list and the underlying array used to implement it.



The length of the list, obtained using `len()`, is the number of items currently in the subarray and not the size of the underlying array.

The list data type has some methods. Here are some of the methods of list objects:

`list.append(x)`

Add an item to the end of the list. Equivalent to `a[len(a):] = [x]`.

`list.extend(iterable)`

Extend the list by appending all the items from the iterable. Equivalent to `a[len(a):] = iterable`.

`list.insert(i, x)`

Insert an item at a given position. The first argument is the index of the element before which to insert, so `a.insert(0, x)` inserts at the front of the list, and `a.insert(len(a), x)` is equivalent to **`a.append(x)`**.

`list.remove(x)`

Remove the first item from the list whose value is equal to `x`. It raises a `ValueError` if there is no such item.

list.pop([i])

Remove the item at the given position in the list, and return it. If no index is specified, a.pop() removes and returns the last item in the list. (The square brackets around the i in the method signature denote that the parameter is optional, not that you should type square brackets at that position. You will see this notation frequently in the Python Library Reference.)

Sets in Python:

A Set is an unordered collection data type that is iterable, mutable and has no duplicate elements. Python's set class represents the mathematical notion of a set. The major advantage of using a set, as opposed to a list, is that it has a highly optimized method for checking whether a specific element is contained in the set. This is based on a data structure known as a hash table. Since sets are unordered, we cannot access items using indexes like we do in lists.

Creation of Sets:

There are two main methods of creating a set. One method is to use the set function that is available in Python. The other method is to use the curly braces '{ }' and type the list of various elements.

The sets that are created cannot be indexed because they are unordered elements. If any element is repeated in the set, it is not accounted for and will be disregarded. The elements in a set are always arranged in ascending order.

Python program to demonstrate sets

```
myset = set(["a", "b", "c"])
print(myset)

# Adding element to the set
myset.add("d")
print(myset)
```

Methods for Sets:

Adding elements- set.add()

Insertion in set is done through **set.add()** function, where an appropriate record value is created to store in the hash table.

Union

Two sets can be merged using union() function or | operator. Both Hash Table values are accessed and traversed with merge operation perform on them to combine the elements, at the same time duplicates are removed.

Intersection

This can be done through intersection() or & operator. Common Elements are selected. They are similar to iteration over the Hash lists and combining the same values on both the Table.

Difference

To find difference in between sets. Similar to find difference in linked list. This is done through difference() or – operator.

Input: Data in List and Set

Output: Different operations on List and Set

Conclusion:

Post Lab Questions:

1. Describe basic data structures in Python
2. Write a python program to check even numbers from a set of numbers from 1 to 50.
3. Write a Python Program to Concatenate Two Lists
4. Write a Python program to perform any four operations on set.

Additional Reference Links:

7. <https://www.python.org>
8. <https://www.tutorialspoint.com>
9. <https://www.programiz.com/python-programming>

S. Y. B. Tech (ECE)

Trimester: VI

Subject: Linux Based Python Laboratory (CET2005A)

Name:

Class:

Roll No.:

Batch:

Experiment – 05

Title: Introduction to Advanced Data Structures of Python (Any two)

Performed on:

Submitted on:

Marks

Teacher's Signature with date

Aim: Introduction to Advanced Data Structures of Python (Any two)

Objective:

9. To know the Advanced Data Structures of Python.
10. To implement at least two Advanced Data Structures of Python programs.

Theory:

- **Advanced Data Structures in Python:**
- **Tuple**
- **Functions**
- **List and set**
- **Sorting**
- **Dictionary**

Tuple in python

A tuple is another sequence data type that is similar to the list.

The main differences between lists and tuples are:

Lists are enclosed in brackets ([]) and their elements and size can be changed, while tuples are enclosed in parentheses (()) and cannot be updated.

Tuple example

TupSub = ('ADC', 'MC' , 'EE322')

```
TupMob = ('Iphone6','Sony','Appo')  
Print (TupSub)      # Prints complete list  
Print (TupSub[0])    # Prints first element of the list  
print (TupSub[1:3])  # Prints elements starting from 2nd till 3rd  
Print (TupSub[2:])   # Prints elements starting from 3rd element  
Print (TupMob * 2 )  # Prints list two times  
print (TupSub + TupMob) # Prints concatenated lists
```

Example : Program to merge two unsorted lists

```
A = [100,50,150]  
B = [9,51,20,3]  
Merged sorted list C=[3,9,20,50,51,100,150]  
Step 1: Create two user input list.  
Step 2: Final merge list size is (size of the first list + the size of the second list).  
Step 3: Sort two lists using sort() method.  
Step 4: Merge two sorted list and store it into a third list.  
Step 5: Merging remaining elements of a[] (if any).Merging remaining elements of b[]  
(if any).  
Step 6: Display merged sorted list.  
Step 1: A = [100,50,150]   B = [9,51,20,3]  
Step 2: Final merged list is C and its size is (3+4).  
Step 3: Sort two lists using sort() method.  
        A = [50,100,150]   B = [3,9,20,51]  
Step 4: Merge two sorted list and store it into a third list.  
        C=[3,9,20,50,51]  
Step 5: Merging remaining elements of A or B (if any).  
        C=[3,9,20,50,51,100,150]  
Step 6: Display merged sorted list.
```

Dictionaries

Lists, tuples, and strings hold elements with only integer indices

45	"Coding"	4.5	7	89
----	----------	-----	---	----

**Integer
Indices**

0 1 2 3 4

In essence, each element has an index (or a key) which can only be an integer

- What if we want to store elements with non-integer indices (or keys)?

Python Dictionary

Python 's dictionaries are hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs.

Keys can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

Dictionaries

Lists index their entries based on the position in the list

Dictionaries are like bags - no order

So we index the things we put in the dictionary with a “lookup tag”

```
>>> purse = dict()
>>> purse['money'] = 12
>>> purse['candy'] = 3
>>> purse['tissues'] = 75
>>> print purse
{'money': 12, 'tissues': 75, 'candy': 3}
>>> print purse['candy']
3
>>> purse['candy'] = purse['candy'] + 2
>>> print purse
{'money': 12, 'tissues': 75, 'candy': 5}
```

Example of dictionary

```
dict = {}

dict['one'] = "This is one"

dict[2]    = "This is two"

tinydict = {'name': 'john', 'code': 6734, 'dept': 'sales'}


print dict['one']    # Prints value for 'one' key

print dict[2]        # Prints value for 2 key

print tinydict       # Prints complete dictionary

print tinydict.keys() # Prints all the keys

print tinydict.values() # Prints all the values
```

OUTPUT:

This is one

This is two

{'dept': 'sales', 'code': 6734, 'name': 'john'}

['dept', 'code', 'name']

['sales', 6734, 'john']

Input: Python Programs.

Output: Output of each Program.

Conclusion:

Post Lab Questions:

1. What are advanced data structures in python? Explain in detail.
2. Explain in detail Queues in Python
3. Write a program in python to Print the middle element of a given linked list
4. Remove duplicate elements from a sorted linked list

Additional Reference Links:

10. <https://www.python.org>
11. <https://www.tutorialspoint.com>
12. <https://www.programiz.com/python-programming>

S. Y. B. Tech (ECE)

Trimester: VI

Subject: Linux Based Python Laboratory (CET2005A)

Name:

Class:

Roll No.:

Batch:

Experiment – 06

Title: File Handling Concepts of Python

Performed on:

Submitted on:

Marks

Teacher's Signature with date

Aim: Introduction to File Handling Concepts of Python

Objective: Write a python program to perform following file handling operations: Create, Open, Append, Read, Write

File Handling Concepts (Attempt any one)

- Write a python program to perform following file handling operations: Create, open, append, Read, Write
- Write a python program to count occurrences of characters, numbers, newlines, special characters, spaces from a file and write it in a new file
- Create a database of (Bank/library/students) and write it in a file. Perform different operations like read, write, search, update, delete

Theory:

- A File is an object on a computer that stores data, information, settings, or commands used with a computer
- Advantages of Files
- Data is stored permanently
- Updating data becomes easy
- Data can be shared amongst various programs
- Huge amount of data can be stored

File Types

Text Files	Binary Files
Stores the data in the form of String	Stores data in the form of bytes
Example: “Hello” is stored as 5 characters	“Hello” is stored as 5 bytes
Example: .txt, .c , .cpp	.jpg, .gif, .png

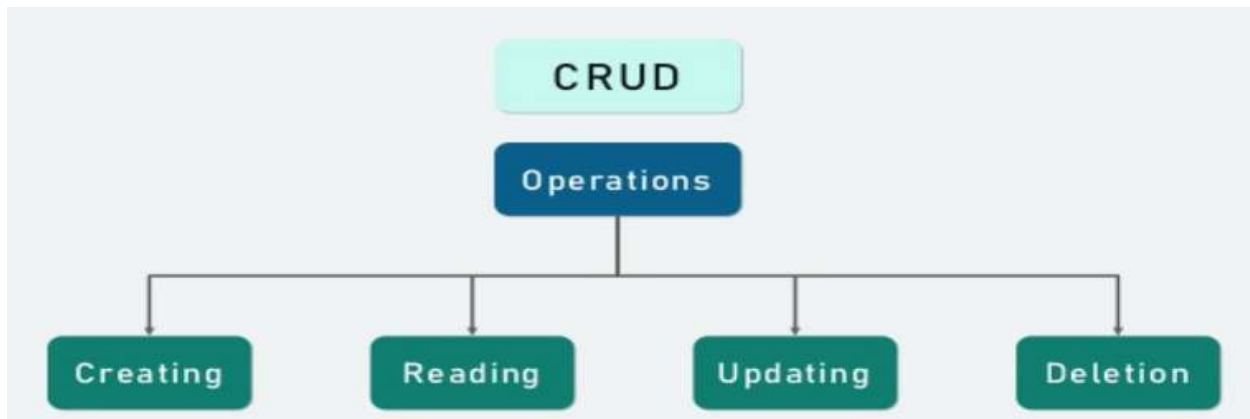
File Types : Text Files

- Text file stores information in **ASCII OR UNICODE** character. In text file everything will be stored as a character for example if data is “computer” then it will take 8 bytes and if the data is floating value like 11237.9876 it will take 10 bytes.
- In text file each line is terminated by special character called **EOL**. In text file some translation takes place when this EOL character is read or written. In python EOL is '\n' or '\r' or combination of both

File Types : Binary Files

- It stores the information in the same format as in the memory i.e. data is stored according to its data type so no translation occurs.
- In binary file there is no delimiter for a new line
- Binary files are faster and easier for a program to read and write than text files.
- Data in binary files cannot be directly read, it can be read only through python program for the same.

File Handling



Opening a File



```
myfile = open("d:\\mydata\\poem.txt","r")
```

here we are accessing "poem.txt" file stored in separate location i.e. d:\mydata folder.

at the time of giving path of file we must use double backslash(\\) in place of single backslash because in python single slash is used for escape character and it may cause problem like if the folder name is "nitin" and we provide path as d:\nitin\poem.txt then in \nitin "\n" will become escape character for new line, SO ALWAYS USE DOUBLE BACKSLASH IN PATH

another solution of double backslash is using "r" before the path making the string as raw string i.e. no special meaning attached to any character as:

```
myfile = open(r"d:\mydata\poem.txt","r")
```

Text File Mode	Binary File Mode	Description	Notes
'r'	'rb'	Read only	File must exists, otherwise Python raises I/O errors
'w'	'wb'	Write only	If file not exists, file is created If file exists, python will truncate existing data and overwrite the file.
'a'	'ab'	Append	File is in write mode only, new data will be added to the end of existing data i.e. no overwriting. If file not exists it is created
'r+'	'r+b' or 'rb+'	Read and write	File must exists otherwise error is raised Both reading and writing can take place
'w+'	'w+b' or 'wb+'	Write and read	File is created if not exists, if exists data will be truncated, both read and write allowed
'a+'	'a+b' or 'ab+'	Write and read	Same as above but previous content will be retained and both read and write.

Reading from File

- To read from file python provide many functions like :
- **Filehandle.read([n])** : reads and return n bytes, if n is not specified it reads entire file.
- **Filehandle.readline([n])** : reads a line of input. If n is specified reads at most n bytes. Read bytes in the form of string ending with line character or blank string if no more bytes are left for reading.
- **Filehandle.readlines():** reads all lines and returns them in a list

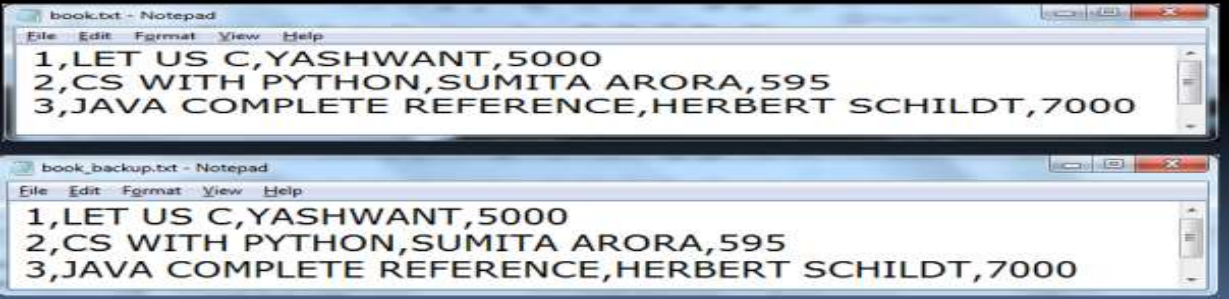
Writing onto Files

Name	Syntax	Description
write()	Filehandle.write(str1)	Writes string str1 to file referenced by filehandle
Writelines()	Filehandle.writelines(L)	Writes all string in List L as lines to file referenced by filehandle.

Copy Content of one file to other

```
myfile1 = open("book.txt","r")
myfile2 = open("book_backup.txt","w")
str1=""
while str1:
    str1 = myfile1.readline()
    myfile2.write(str1)
myfile1.close()
myfile2.close()
print("Copied Successfully...")
```

Copied Successfully...



flush() function

- When we write any data to file, python hold everything in buffer (temporary memory) and pushes it onto actual file later. If you want to force Python to write the content of buffer onto storage, you can use flush() function.
- Python automatically flushes the files when closing them i.e. it will be implicitly called by the close(), BUT if you want to flush before closing any file you can use flush()

Files

The With Statement

- Python's "with" statement for file handling is very handy when you have two related operations which you would like to execute as a pair, with a block of code in between:
**with open(filename[, mode]) as filehandle:
file_manipulation_statement**
- The advantage of "with" is it will automatically close the file after nested block of code. It guarantees to close the file how nested block exits even if any run time error occurs

Binary File Operations

- If we want to write a structure such as list or dictionary to a file and read it subsequently we need to use the Python module **pickle**.

Pickling is the process of converting structure to a byte stream before writing to a file and while reading the content of file a reverse process called **Unpickling** is used to convert the byte stream back to the original format.

Steps to perform binary file operations

- First we need to import the module called pickle.
- This module provides 2 main functions:
 - dump()** : to write the object in file which is loaded in binary mode
 - Syntax: `dump(object_to_write, filehandle)`
 - load()** : dumped data can be read from file using load() i.e. it is used to read object from pickle file.
 - Syntax: `object = load(filehandle)`

References:

https://www.w3schools.com/python/python_file_handling.asp

<https://stackoverflow.com/questions/39606186/python-class-employee-management-system>

<https://www.tutorialaicsip.com/cs-xii/file-handling-binary-file-operations-in-python-search-append-update-and-delete-records/>

Input:

Output:

Conclusion:

Post Lab Questions:

1. Write a function in Python that counts the number of “Me” or “My” (in smaller case also) words present in a text file “STORY.TXT”. If the “STORY.TXT” contents are as follows:
My first book was Me and My Family. It gave me chance to be Known to the world.

The output of the function should be: Count of Me/My in file: 4

2. Write a function AMCount() in Python, which should read each character of a text file STORY.TXT, should count and display the occurrence of alphabets ‘A’ and ‘M’ (including small cases ‘a’ and ‘m’ too). Example: If the file content is as follows:

The AMCount() function should display the output as: A or a = 4, M or m =2

3. Write a function in python to count the number of lines in a text file
4. Write a user-defined function named count() that will read the contents of text file named “Story.txt” and count the number of lines which starts with either “I” or “M”. E.g. In the following paragraph, there are 2 lines starting with “I” or “M”:

“India is the fastest growing economy.

India is looking for more investments around the globe.

The whole world is looking at India as a great market.

Most of the Indians can foresee the heights that India is capable of reaching.”