

RISHI MS INSTITUTE OF ENGINEERING & TECHNOLOGY FOR WOMEN

(Approved by AICTE, New Delhi and Affiliated to JNTUH)

Nizampet Cross Road, JNTUH Kukatpally Hyderabad-500085



LABORATORY MANUAL

APPLIED PYTHON

PROGRAMMING

(R22 Regulations)

For

B. Tech

I Year II semester

DEPARTMENT OF

ELECTRONICS AND COMMUNICATION ENGINEERING

INDEX

S.NO	TOPIC	PAGENO
I	List of Experiments	iii
II	V/M/POs/PSOs/PEOs	vi
III	Syllabus	xi
IV	Course Objectives & Course Outcomes	xiv

EXP.NO	Experiment Name	Page No
1	Downloading and Installing Python and Modules	
	<p>a) Python 3 on Linux Follow the instructions given in the URL https://docs.pythonguide.org/starting/install3/linux/</p> <p>b) Python 3 on Windows Follow the instructions given in the URL https://docs.python.org/3/using/windows.html (Please remember that Windows installation of Python is harder!)</p> <p>c) pip3 on Windows and Linux Install the Python package installer by following the instructions given in the URL https://www.activestate.com/resources/quick-reads/how-to-install-and-use-pip3/</p> <p>d) Installing numpy and scipy You can install any python3 package using the command pip3 install</p> <p>e) Installing jupyter lab Install from pip using the command pip install jupyter lab</p>	
2	Introduction to Python3	
	<p>a) Printing your biodata on the screen</p> <p>b) Printing all the primes less than a given number</p> <p>c) Finding all the factors of a number and show whether it is a perfect number, i.e., the sum of all its factors (excluding the number itself) is equal to the number itself</p>	
3	Defining and Using Functions	
	<p>a) Write a function to read data from a file and display it on the screen</p> <p>b) Define a boolean function is palindrome(<input type="text"/>)</p> <p>c) Write a function collatz(x) which does the following: if x is odd, $x = 3x + 1$; if x is even, then $x = x/2$. Return the number of steps it takes for $x = 1$</p> <p>d) Write a function $N(m, s) = \exp(-(x-m)^2 / (2s^2)) / \sqrt{2\pi}s$ that computes the Normal distribution</p>	
4	The package numpy	

	<p>a) Creating a matrix of given order $m \times n$ containing random numbers in the range 1 to 99999</p> <p>b) Write a program that adds, subtracts and multiplies two matrices. Provide an interface such that, based on the prompt, the function (addition, subtraction, multiplication) should be performed</p> <p>c) Write a program to solve a system of n linear equations in n variables using matrix inverse</p>	
5	The package scipy and pyplot	
	<p>a) Finding if two sets of data have the same mean value</p> <p>b) Plotting data read from a file</p> <p>c) Fitting a function through a set of data points using polyfit function</p> <p>d) Plotting a histogram of a given data set</p>	
6	The strings package	
	<p>a) Read text from a file and print the number of lines, words and characters</p> <p>b) Read text from a file and return a list of all n letter words beginning with a vowel</p> <p>c) Finding a secret message hidden in a paragraph of text</p> <p>d) Plot a histogram of words according to their length from text read from a file</p>	
7	Installing OS on Raspberry Pi	
	<p>a) Installation using PiImager</p> <p>b) Installation using image file</p> <ul style="list-style-type: none"> • Downloading an Image • Writing the image to an SD card • using Linux • using Windows • Booting up Follow the instructions given in the URL https://www.raspberrypi.com/documentation/computers/getting-started.html 	
8	Accessing GPIO pins using Python	

	<p>a) Installing GPIO Zero library. First, update your repositories list: <code>sudo apt update</code> Then install the package for Python 3: <code>sudo apt install python3-gpiozero</code></p> <p>b) Blinking an LED connected to one of the GPIO pin</p> <p>c) Adjusting the brightness of an LED</p> <p>d) Adjust the brightness of an LED (0 to 100, where 100 means maximum brightness) using the in-built PWM wavelength.</p>	
9	Collecting Sensor Data	
	<p>a) DHT Sensor interface</p> <ul style="list-style-type: none"> ◦ Connect the terminals of DHT GPIO pins of Raspberry Pi. ◦ Import the DHT library using <code>import Adafruit_DHT</code> ◦ Read sensor data and display it on screen. 	



RISHI M.S. INSTITUTE OF ENGINEERING & TECHNOLOGY FOR WOMEN

(Affiliated to JNTUH University, Approved by AICTE)

Department of Electronics and Communication Engineering

Vision of the institution:

To be a center of excellence in producing women engineers and scientists who are professionally competent social leaders to face multi-disciplinary global environment by imparting quality technical education, values and ethics through innovation methods of teaching and learning.

Mission of the institution:

- To promote women technocrats capable enough to resolve the problems faced by the society using the knowledge imparted.
- To prepare self-reliant women engineering for technological growth of the nation and society by laying strong theoretical foundation accompanied by wide practical training.
- To equip the young women with creative thinking capabilities and empowering them towards innovation.



RISHI M.S. INSTITUTE OF ENGINEERING & TECHNOLOGY FOR WOMEN

(Affiliated to JNTUH University, Approved by AICTE)

Department of Electronics and Communication Engineering

Vision & Mission of Department

Vision of the department

To empower women by providing cutting-edge technology to female technocrats in the fields of Information Technology, allowing them to develop into competent engineers and entrepreneurs.

Mission of the department

- Adopting creative techniques to nurture and strengthen the core skill of Computer Science.
- Introduce students to the most recent technological advancements.
- Impart quality education; improve the research, entrepreneurial, and employability skills of women technocrats.
- Instill professional ethics and a sense of social responsibility in students.
- Strengthen the Industry-Academia interface, which will enable graduates to emerge as academic leaders or inspiring entrepreneurs



RISHI M.S. INSTITUTE OF ENGINEERING & TECHNOLOGY FOR WOMEN

(Affiliated to JNTUH University, Approved by AICTE)

Department of Electronics and Communication Engineering

A B.Tech –graduate should possess the following program outcomes.

- 1) **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
- 2) **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- 3) **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- 4) **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- 5) **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- 6) **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- 7) **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- 8) **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- 9) **Individual and teamwork:** Function effectively as an individual, and as a member or leader in diverse teams, and in multi disciplinary settings.
- 10) **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

Project management and finance: Demonstrate knowledge and understandi



**RISHI M.S. INSTITUTE OF ENGINEERING & TECHNOLOGY FOR
WOMEN**

(Affiliated to JNTUH University, Approved by AICTE)

Department of Electronics and Communication Engineering

Program specific outcomes (PSOs)

PSO 1: Improve the student's ability to decipher the basic principles and methodology of computer systems. Improve the student's ability to absorb facts and technical ideas in order to build and develop software.

PSO2: The capacity to create novel job routes as an entrepreneur using modern computer languages and evolving technologies like SDLC, Python, Machine Learning, Social Networks, Cyber Security, Mobile Apps etc.



**RISHI M.S. INSTITUTE OF ENGINEERING & TECHNOLOGY FOR
WOMEN**

(Affiliated to JNTUH University, Approved by AICTE)

Department of Electronics and Communication Engineering

Program educational objectives (PEOs)

PEO-1: Engineering graduates with excellent fundamental and technical skills will have successful careers in industry, meeting the needs of Indian and worldwide firms.

PEO-2: With determination, development, self-reliance, leadership, morality, and moral principles, engineering graduates will become successful entrepreneurs who will ever age employability.

PEO-3: To support personal and organizational progress, engineering graduates will pursue higher education and engage in lifelong learning

SYLLABUS
APPLIED PYTHON PROGRAMMING
B. TECH I Year II Sem

Week 1:

1. Downloading and Installing Python and Modules

- A) Python 3 on Linux
Follow the instructions given in the URL
<https://docs.python-guide.org/starting/install3/linux/>
- B) Python 3 on Windows
Follow the instructions given in the URL
<https://docs.python.org/3/using/windows.html> (Please remember that Windows installation of Python is harder!)
- C) pip3 on Windows and Linux
Install the Python package installer by following the instructions given in the URL <https://www.activestate.com/resources/quick-reads/how-to-install-and-use-pip3/>
- D) Installing numpy and scipy
You can install any python3 package using the command `pip3 install <packagename>`
- E) Installing jupyterlab
Install from pip using the command `pip install jupyterlab`

Week 2:

1. Introduction to Python3

- a) Printing your biodata on the screen
- b) Printing all the primes less than a given number
- c) Finding all the factors of a number and show whether it is a *perfect* number, i.e., the sum of all its factors (excluding the number itself) is equal to the number itself

Week 3:

1. Defining and Using Functions

- a) Write a function to read data from a file and display it on the screen
- b) Define a boolean function *is palindrome*(<input>)
- c) Write a function *collatz*(*x*) which does the following: if *x* is odd, $x = 3x + 1$; if *x* is even, then $x = x/2$. Return the number of steps it takes for $x = 1$
- d) Write a function $N(m, s) = \exp(-(x-m)^2/(2s^2))/\sqrt{2\pi}s$ that computes the Normal distribution.

Week 4:

1. The package numpy
 - a) Creating a matrix of given order $m \times n$ containing *random numbers* in the range 1 to 99999
 - b) Write a program that adds, subtracts and multiplies two matrices. Provide an interface such that, based on the prompt, the function (addition, subtraction, multiplication) should be performed
 - c) Write a program to solve a system of n linear equations in n variables using matrix inverse
2. The package scipy and pyplot
 - a) Finding if two sets of data have the same *mean* value.
 - b) Plotting data read from a file
 - c) Fitting a function through a set of data points using *polyfit* function
 - d) Plotting a histogram of a given data.

Week 5:

2. The strings package
 - A) Read text from a file and print the number of lines, words and characters
 - B) Read text from a file and return a list of all n letter words beginning with a vowel
 - C) Finding a secret message hidden in a paragraph of text
 - D) Plot a histogram of words according to their length from text read from a file.

Week 6:

Installing OS on Raspberry Pi

- a) Installation using PiImager
 - b) Installation using image file
- Downloading an Image
 - Writing the image to an SD card
 - using Linux
 - using Windows
 - Booting up

Follow the instructions given in the URL

<https://www.raspberrypi.com/documentation/computers/getting-started.html>

Week 7:

Accessing GPIO pins using Python

- a) Installing GPIO Zero library.

First, update your repositories list:

sudo apt update

Then install the package for Python 3:

b)Blinking an LED connected to one of the GPIO pin

c)Adjusting the brightness of an LED

d)Adjust the brightness of an LED (0 to 100, where 100 means maximum brightness) usingthe in-built PWM wavelength.

Week 8:

Collecting Sensor Data

DHT Sensor interface

- Connect the terminals of DHT GPIO pins of Raspberry Pi.
- Import the DHT library using *import Adafruit_DHT*
- Read sensor data and display it on screen.

Course Objectives:

1. To provide an understanding of the design aspects of operating system concepts through simulation.
2. Introduce basic Unix commands, system call interface for process management, inter-process communication and I/O in Unix.

Course Outcomes: After learning the contents of this course the student is able to

CO1. Simulate and implement operating system concepts such as scheduling, deadlock

management, file management and memory management.

CO2. Able to implement C programs using Unix system calls

CO-PO MAPPING

	CO	PO 1	PO 2	PO 3	PO 4	PO5	PO 6	PO7	PO8	PO 9	PO10	PO1 1	PO1 2
SKILL DEVEL OPM	CO1	2	3	3	2	2			2				2
ENT COURSE (NODE JS/ REACT JS/ DJANGO)	CO2	2	3	1	1	3		2			3		
	CO3	2	2	3	3	3			3		1		

CO-PSO MAPPING

	PSO-1	PSO-2
CO1	3	2
CO2	3	2
CO3	3	3

WEEK 1

1. Downloading and Installing Python and Modules

a. Python 3 on Linux

Follow the instructions given in the URL

<https://docs.python-guide.org/starting/install3/linux/>


Beginning Python installation

For almost every Linux system, the following command could be used to install Python directly:

```
$ sudo apt-get install python3.8
```

Getting Started

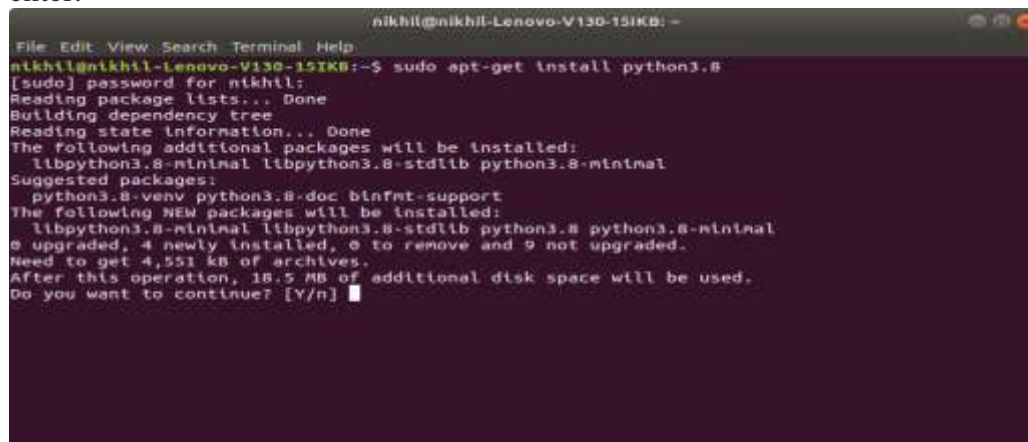
We will start our installation by writing the password as mentioned in the below picture that gives the permission to perform further operations.



```
nikhil@nikhil-Lenovo-V130-15IKB: ~$ sudo apt-get install python3.8
[sudo] password for nikhil: 
```

Assigning DiskSpace

Generally, it asks for additional disk space and we generally continue by typing Y and then enter.



```
nikhil@nikhil-Lenovo-V130-15IKB: ~$ sudo apt-get install python3.8
[sudo] password for nikhil: 
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libpython3.8-minimal libpython3.8-stdlib python3.8-minimal
Suggested packages:
  python3.8-venv python3.8-doc binfmt-support
The following NEW packages will be installed:
  libpython3.8-minimal libpython3.8-stdlib python3.8 python3.8-minimal
0 upgraded, 4 newly installed, 0 to remove and 9 not upgraded.
Need to get 4,551 kB of archives.
After this operation, 18.5 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Fetching and Installing Packages

It's a process in which required packages and modules are fetched and installed automatically. It happens on its own when we give the permission to assign the disk space.

```
nikhil@nikhil-Lenovo-V130-15IKB: ~  
File Edit View Search Terminal Help  
[sudo] password for nikhil:  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  libpython3.8-minimal libpython3.8-stdlib python3.8-minimal  
Suggested packages:  
  python3.8-venv python3.8-doc binfmt-support  
The following NEW packages will be installed:  
  libpython3.8-minimal libpython3.8-stdlib python3.8 python3.8-minimal  
0 upgraded, 4 newly installed, 0 to remove and 9 not upgraded.  
Need to get 4,551 kB of archives.  
After this operation, 18.5 MB of additional disk space will be used.  
Do you want to continue? [Y/n] y  
Get:1 http://ln.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 libpython3.8-minimal  
amd64 3.8.0-3-18.04 [704 kB]  
Get:2 http://ln.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 python3.8-minimal amd  
64 3.8.0-3-18.04 [1,016 kB]  
Get:3 http://ln.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 libpython3.8-stdlib a  
md64 3.8.0-3-18.04 [1,677 kB]  
Get:4 http://ln.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 python3.8 amd64 3.8.0  
-3-18.04 [355 kB]  
Fetched 4,551 kB in 3s (1,427 kB/s)
```

Getting through the installation process

```
nikhil@nikhil-Lenovo-V130-15IKB: ~  
File Edit View Search Terminal Help  
-3-18.04 [355 kB]  
Fetched 4,551 kB in 3s (1,427 kB/s)  
Selecting previously unselected package libpython3.8-minimal:amd64.  
(Reading database ... 258857 files and directories currently installed.)  
Preparing to unpack .../libpython3.8-minimal_3.8.0-3-18.04_amd64.deb ...  
Unpacking libpython3.8-minimal:amd64 (3.8.0-3-18.04) ...  
Selecting previously unselected package python3.8-minimal.  
Preparing to unpack .../python3.8-minimal_3.8.0-3-18.04_amd64.deb ...  
Unpacking python3.8-minimal (3.8.0-3-18.04) ...  
Selecting previously unselected package libpython3.8-stdlib:amd64.  
Preparing to unpack .../libpython3.8-stdlib_3.8.0-3-18.04_amd64.deb ...  
Unpacking libpython3.8-stdlib:amd64 (3.8.0-3-18.04) ...  
Selecting previously unselected package python3.8.  
Preparing to unpack .../python3.8_3.8.0-3-18.04_amd64.deb ...  
Unpacking python3.8 (3.8.0-3-18.04) ...  
Setting up libpython3.8-minimal:amd64 (3.8.0-3-18.04) ...  
Setting up python3.8-minimal (3.8.0-3-18.04) ...  
Setting up libpython3.8-stdlib:amd64 (3.8.0-3-18.04) ...  
Setting up python3.8 (3.8.0-3-18.04) ...  
Processing triggers for gnome-menus (3.13.3-11ubuntu1) ...  
Processing triggers for mime-support (3.60ubuntu1) ...  
Processing triggers for desktop-file-utils (0.23-1ubuntu3.18.04.2) ...  
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
```

Finished Installation

Here, the required installation is completed.

```
nikhil@nikhil-Lenovo-V130-15IKB: ~  
File Edit View Search Terminal Help  
-3-18.04 [355 kB]  
Fetched 4,551 kB in 3s (1,427 kB/s)  
Selecting previously unselected package libpython3.8-minimal:amd64.  
(Reading database ... 258857 files and directories currently installed.)  
Preparing to unpack .../libpython3.8-minimal_3.8.0-3-18.04_amd64.deb ...  
Unpacking libpython3.8-minimal:amd64 (3.8.0-3-18.04) ...  
Selecting previously unselected package python3.8-minimal.  
Preparing to unpack .../python3.8-minimal_3.8.0-3-18.04_amd64.deb ...  
Unpacking python3.8-minimal (3.8.0-3-18.04) ...  
Selecting previously unselected package libpython3.8-stdlib:amd64.  
Preparing to unpack .../libpython3.8-stdlib_3.8.0-3-18.04_amd64.deb ...  
Unpacking libpython3.8-stdlib:amd64 (3.8.0-3-18.04) ...  
Selecting previously unselected package python3.8.  
Preparing to unpack .../python3.8_3.8.0-3-18.04_amd64.deb ...  
Unpacking python3.8 (3.8.0-3-18.04) ...  
Setting up libpython3.8-minimal:amd64 (3.8.0-3-18.04) ...  
Setting up python3.8-minimal (3.8.0-3-18.04) ...  
Setting up libpython3.8-stdlib:amd64 (3.8.0-3-18.04) ...  
Setting up python3.8 (3.8.0-3-18.04) ...  
Processing triggers for gnome-menus (3.13.3-11ubuntu1) ...  
Processing triggers for mime-support (3.60ubuntu1) ...  
Processing triggers for desktop-file-utils (0.23-1ubuntu3.18.04.2) ...  
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...  
nikhil@nikhil-Lenovo-V130-15IKB: ~$
```

To verify the installation enter the following commands in your Terminal.
python3.8

Run our First Python Program

Let's consider a simple Hello World Program

```
# Python program to print
```

```
# Hello World
```

```
print("Hello World")
```

Output:

A screenshot of a Linux terminal window. The title bar shows the user 'nikhil' on a machine named 'nikhil-Lenovo-V130-15IKB' in the directory '~/Desktop/gfg'. The terminal has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The prompt is 'nikhil@nikhil-Lenovo-V130-15IKB:~/Desktop/gfg\$'. The user has entered 'python3.8 HelloWorld.py', and the output 'Hello World' is displayed on the next line. The prompt is now 'nikhil@nikhil-Lenovo-V130-15IKB:~/Desktop/gfg\$' with a cursor at the end.

```
nikhil@nikhil-Lenovo-V130-15IKB: ~/Desktop/gfg
File Edit View Search Terminal Help
nikhil@nikhil-Lenovo-V130-15IKB:~/Desktop/gfg$ python3.8 HelloWorld.py
Hello World
nikhil@nikhil-Lenovo-V130-15IKB:~/Desktop/gfg$
```

b)Python 3 on Windows

Follow the instructions given in the URL

<https://docs.python.org/3/using/windows.html> (Please remember that Windows installation of Python is harder!)

The installation requires downloading the official Python .exe installer and running it on your system. The sections below will explain several options and details during the installation process.

Step 1: Select Python Version

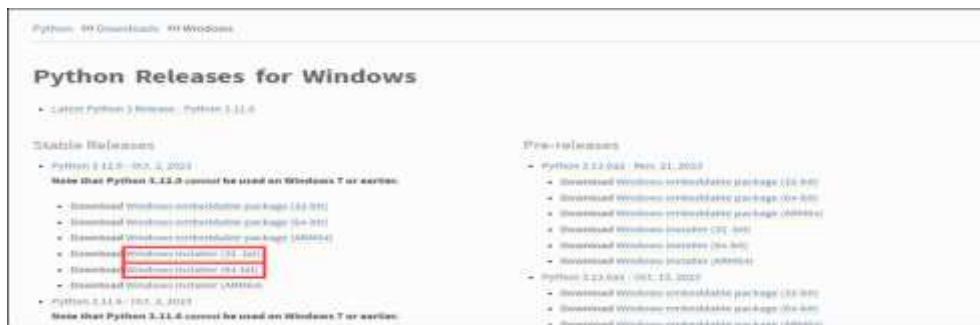
Deciding on a version depends on what you want to do in Python. The two major versions are Python 2 and Python 3. Choosing one over the other might be better depending on your project details. If there are no constraints, choose whichever one you prefer.

We recommend Python 3, as Python 2 reached its end of life in 2020. Download Python 2 only if you work with legacy scripts and older projects. Also, choose a stable release over the newest since the newest release may have bugs and issues.

Step 2: Download Python Executable Installer

Start by downloading the Python executable installer for Windows:

1. Open a web browser and navigate to the Downloads for Windows section of the official Python website.
2. Locate the desired Python version.



3. Click the link to download the file. Choose either the Windows 32-bit or 64-bit installer.

The download is approximately 25MB.

Step 3: Run Executable Installer

The steps below guide you through the installation process:

1. Run the downloaded **Python Installer**.

2. The installation window shows two checkboxes:

- **Admin privileges.** The parameter controls whether to install Python for the current or all system users. This option allows you to change the installation folder for Python.
- **Add Python to PATH.** The second option places the executable in the PATH variable after installation. You can also add Python to the PATH environment variable manually later.



For the most straightforward installation, we recommend ticking both checkboxes.

3. Select the **Install Now** option for the recommended installation (in that case, skip the next two steps).

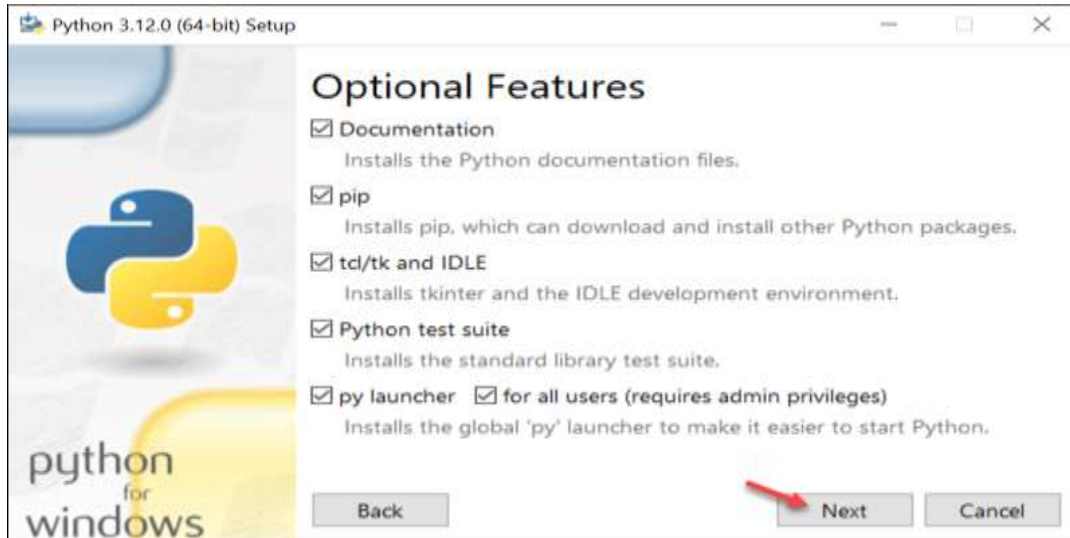
To adjust the default installation options, choose **Customize installation** instead and proceed to the following step.



The default installation installs Python to `C:\Users\[user]\AppData\Local\Programs\Python\Python[version]` for the current user. It includes IDLE (the default Python editor), the PIP package manager, and additional documentation. The installer also creates necessary shortcuts and file associations.

Customizing the installation allows changing these installation options and parameters.

4. Choose the optional installation features. Python works without these features, but adding them improves the program's usability.



Click **Next** to proceed to the Advanced Options screen.

5. The second part of customizing the installation includes advanced options.

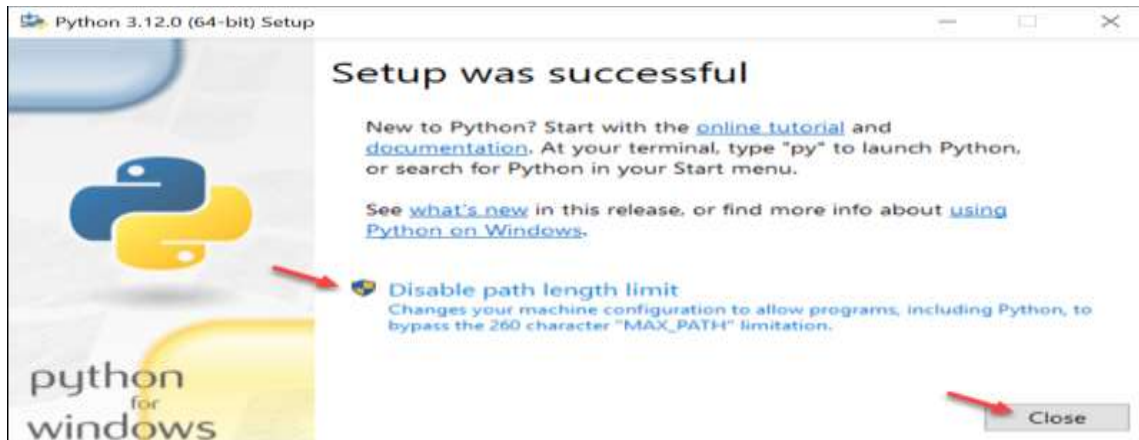
Choose whether to install Python for all users. The option changes the install location to `C:\Program Files\Python[version]`. If selecting the location manually, a common choice is `C:\Python[version]` because it avoids spaces in the path, and all users can access it. Due to administrative rights, both paths may cause issues during package installation.

Other advanced options include creating shortcuts, file associations, and adding Python to PATH.



After picking the appropriate options, click **Install** to start the installation.

6. Select whether to disable the path length limit. Choosing this option will allow Python to bypass the 260-character **MAX_PATH** limit.



The option will not affect any other system settings, and disabling it resolves potential name-length issues. We recommend selecting the option and closing the setup.

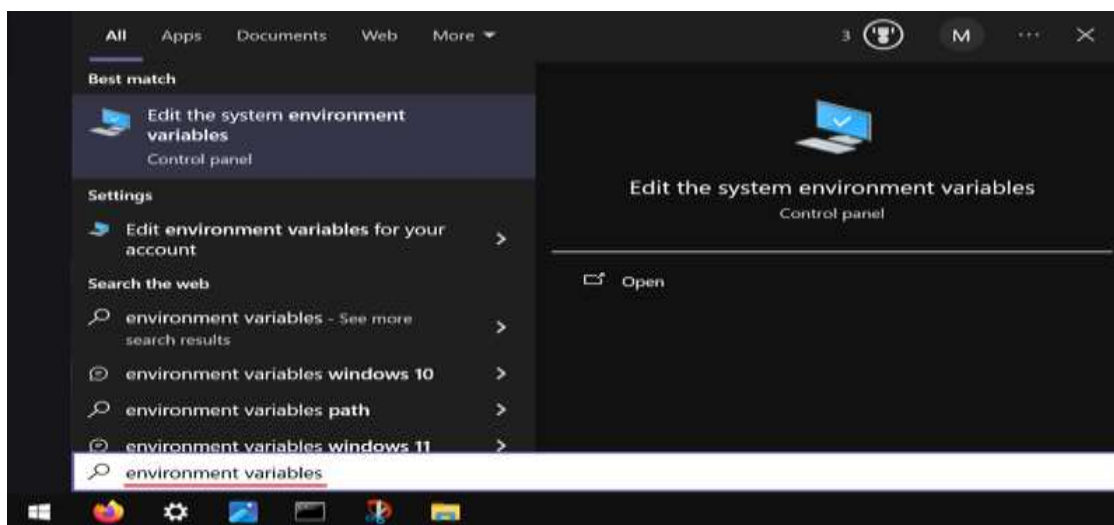
Step 4: Add Python to Path (Optional)

If the Python installer does not include the **Add Python to PATH** checkbox or you have not selected that option, continue in this step. Otherwise, skip to the next step.

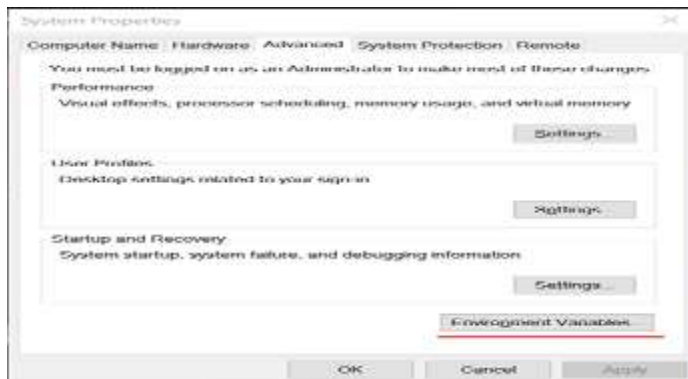
Adding the Python path to the PATH variable alleviates the need to use the full path to access the Python program in the command line. It instructs Windows to review all the folders added to the PATH environment variable and to look for the *python.exe* program in those folders.

To add Python to PATH, do the following:

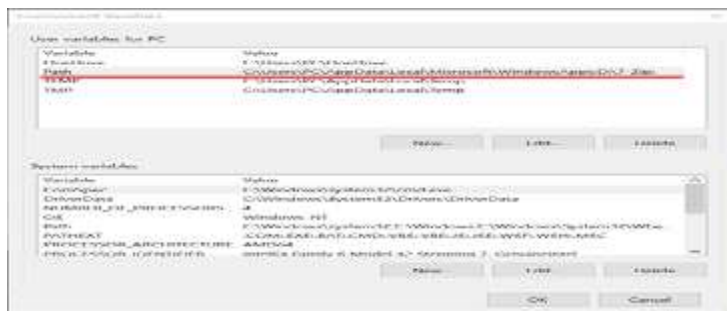
1. In the **Start** menu, search for **Environment Variables** and press **Enter**.



2. Click **Environment Variables** to open the overview screen.

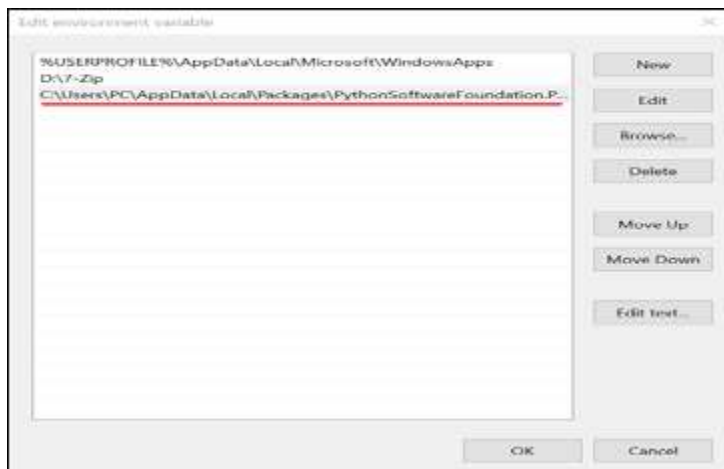


3. Double-click **Path** on the list to edit it.



Alternatively, select the variable and click the **Edit** button.

4. Double-click the first empty field and paste the Python installation folder path.



Alternatively, click the **New** button instead and paste the path.

3. Click **OK** to save the changes. If the command prompt is open, restart it for the following step.

c) pip3 on Windows and Linux

Install the Python package installer by following the instructions given in the URL <https://www.activestate.com/resources/quick-reads/how-to-install-and-use-pip3/>

Check if pip is already installed by running the following on the command line:

```
pip --version
```

or

```
pip -V
```

If pip is installed, you'll see something similar to the following output:

```
C:\users\jdoe>pip --version
```

```
Pip 19.2.3 from c:\users\jdoe\appdata\local\programs\python\python38-32\lib\site-packages\pip (python 3.8)
```

2. Verify that Python is installed. If pip is not installed, you can confirm that Python is available on your local machine and determine the version by running the following command:

```
python --version
```

You should see something like the following if Python is installed:

```
C:\users\jdoe>python --versionPython 3.8.0
```

If you do not have a version of Python installed, you can quickly download and install a recent version of [ActivePython](#).

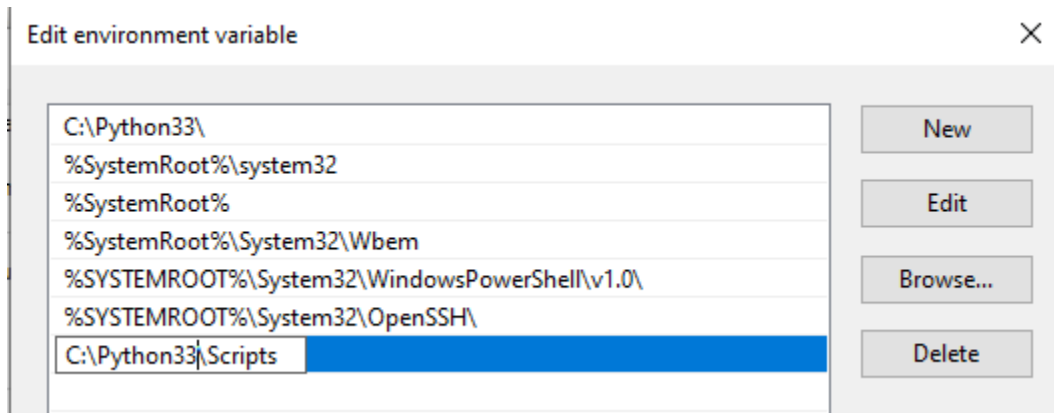
Adding PIP to Windows Environment Variables

One of the most common problems with running Python tools like pip is the “not on PATH” error. This means that Python cannot find the tool you’re trying to run in your current directory. In most cases, you’ll need to navigate to the directory in which the tool is installed before you can run the command to launch it.

If you’d rather run pip (or other tools) from any location, you’ll need to add the directory in which it’s installed as a PATH environment variable by doing the following:

1. Open up the **Control Panel** and navigate to **System and Security > System**
2. Click on the **Advanced system settings** link on the left panel

3. Click **Environment Variables**.
4. Under **System Variables**, double-click the variable PATH.
5. Click **New**, and add the directory where pip is installed, e.g. C:\Python33Scripts, and select **OK**.



Upgrading Pip on Windows

Pip is a key tool in the Python ecosystem, and as such is updated on a regular basis. Changes can always be found in the [release notes](#) for each version. In order to keep your version of pip up to date, you can run the following on the command line:

```
python -m pip install -U pip
```

This command will uninstall the outdated version of pip first, and then download the most current iteration.

It's also possible to downgrade to a previous version of pip, in case a newer version is causing unexpected compatibility errors. For example, to downgrade to pip v18.0 run the following command:

```
python -m pip install pip==18.0
```


d)Installing numpy and scipy

You can install any python3 package using the command `pip3 install <packagename>`
Steps for Installing NumPy

1. Download and Install Python.
 - Python should be installed before installing NumPy.
 - Download Python from [Python's official website](#) according to the operating system. After installing Python, Numpy is installed using different commands in different operating systems.

Let's see how NumPy is installed on different operating systems.

Installing Numpy

1. Installing NumPy on Mac Operating System

1. Check if pip3 and python3 are installed correctly by running the following command:

```
python3 --version  
pip3 --version
```

2. Install numpy by using the pip command:

```
pip3 install numpy
```

3. Verify installation by typing the following command:

```
import numpy
```

2. Installing NumPy on Windows Operating System

Steps for installing NumPy on Windows:

1. Install NumPy using the following PIP command in the command prompt terminal:

```
pip install numpy
```

e)Installing jupyterlab

Install from pip using the command `pip install jupyterlab`

JupyterLab

Install JupyterLab with **pip**:

```
pip install jupyterlab
```

Note: If you install JupyterLab with conda or mamba, we recommend using [the conda-forge channel](#).

Once installed, launch JupyterLab with:

```
jupyter lab
```

WEEK 2

2.Introduction to Python3

a)Printing your biodata on the screen

```
# Define a function 'personal_details'.  
  
def personal_details():  
  
    # Define variables 'name' and 'age' and assign values to them.  
  
    name, age = "Simon", 19  
  
    # Define a variable 'address' and assign a value to it.  
  
    address = "Bangalore, Karnataka, India"  
  
    # Print the personal details using string formatting.  
  
    print("Name: {}\nAge: {}\nAddress: {}".format(name, age, address))  
  
  
# Call the 'personal_details' function to display the details.  
  
personal_details()
```

Sample Output:

```
Name: Simon  
Age: 19  
Address: Bangalore, Karnataka, India
```

b) Printing all the primes less than a given number

```
def primenumber(MyNum):  
    n = 0  
    i = 2  
    for i in range(2, MyNum//2+1):  
        if MyNum % i == 0:  
            n = n + 1  
            break  
    if n == 0:  
        print(MyNum, end=" ")  
  
x = 50  
print("Prime numbers less than", x, "are:")  
for i in range(2, x+1):  
    primenumber(i)
```

The above code will give the following output:

```
Prime numbers less than 50 are:  
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47
```

c)Finding all the factors of a number and show whether it is a *perfect number*, i.e., the sumof all its factors (excluding the number itself) is equal to the number itself.

```
n = int(input("Enter any number: "))
sum1 = 0
for i in range(1, n):
    if(n % i == 0):
        sum1 = sum1 + i
if (sum1 == n):
    print("The number is a Perfect number!")
else:
    print("The number is not a Perfect number!")
```

output:

case 1:

Enter any number: 6

The number is a Perfect number!

Case 2:

Enter any number: 25

The number is not a Perfect number!

WEEK 3

3. Defining and Using Functions

a) Write a function to read data from a file and display it on the screen

```
a=str(input("Enter the name of the file with .txt extension:"))
file2=open(a,'r')
line=file2.readline()
while(line!=""):
    print(line)
    line=file2.readline()
file2.close()
```

output:

case 1:

Contents of file:

Hello world

Output:

Enter the name of the file with .txt extension: data1.txt

Case 2:

Contents of file:

This programming language is
Python.

Output:

Enter the name of the file with .txt extension: data2.txt

This programming language is
Python.

b) Define a boolean function *is palindrome*(<input>)

```
#Define a function
def isPalindrome(string):
    if (string == string[::-1]):
        return "The string is a palindrome."
    else:
        return "The string is not a palindrome."

#Enter input string
string = input ("Enter string: ")

print(isPalindrome(string))
```

Let's look at the output when we enter the string "rodent" :

```
Enter string: rodent
The string is not a palindrome.
```

c) Write a function *collatz(x)* which does the following: if x is odd $x = 3x + 1$; if x is even, then $x = x/2$. Return the number of steps it takes for $x = 1$

```
# Python 3 program to print  
# Collatz sequence
```

```
def printCollatz(n):
```

```
    # We simply follow steps  
    # while we do not reach 1  
    while n != 1:  
        print(n, end = ' ')
```

```
    # If n is odd  
    if n & 1:  
        n = 3 * n + 1
```

```
    # If even  
    else:  
        n = n // 2
```

```
    # Print 1 at the end  
    print(n)
```

```
# Driver code  
printCollatz(6)
```

output:

```
6 3 10 5 16 8 4 2 1
```


d) Write a function $N(m, s) = \exp(-(x-m)^2/(2s^2))/\sqrt{2\pi}s$ that computes the Normal distribution.

```
import numpy as np

def normal_dist(x, mean, sd):

    prob_density = (np.pi*sd) * np.exp(-0.5*((x-mean)/sd)**2)

    return prob_density

mean = 0

sd = 1

x = 1

result = normal_dist(x, mean, sd)

print(result)
```

Output:
1.9054722647301798

WEEK 4

1.The package numpy

a)Creating a matrix of given order m x n containing random numbers in the range 1 to 99999

```
# importing numpy library
```

```
import numpy as np
```

```
# random is a function, doing random sampling in numpy.
```

```
array = np.random.randint(10, size=(20))
```

```
# the array will be having 20 elements.
```

```
print(array)
```

OUTPUT:

```
[2 6 1 4 3 3 6 5 0 3 6 8 9 1 6 4 0 5 4 1]
```

B)Write a program that adds, subtracts and multiplies two matrices. Provide an interfacesuch that, based on the prompt, the function (addition, subtraction, multiplication) should be performed.

ADDITION

```
# importing numpy as np
import numpy as np

# creating first matrix
A = np.array([[1, 2], [3, 4]])

# creating second matrix
B = np.array([[4, 5], [6, 7]])

print("Printing elements of first matrix")
print(A)
print("Printing elements of second matrix")
print(B)

# adding two matrix
print("Addition of two matrix")
print(np.add(A, B))
```

SUBTRACTION

```
# importing numpy as np
import numpy as np

# creating first matrix
A = np.array([[1, 2], [3, 4]])

# creating second matrix
```

```

B = np.array([[4, 5], [6, 7]])
print("Printing elements of first matrix")
print(A)
print("Printing elements of second matrix")
print(B)
# subtracting two matrix
print("Subtraction of two matrix")
print(np.subtract(A, B))

```

MULTIPLICATION

```

# Program to multiply two matrices using nested loops
# 3x3 matrix
X = [[12,7,3],
     [4 ,5,6],
     [7 ,8,9]]
# 3x4 matrix
Y = [[5,8,1,2],
     [6,7,3,0],
     [4,5,9,1]]
# result is 3x4
result = [[0,0,0,0],
          [0,0,0,0],
          [0,0,0,0]]
# iterate through rows of X
for i in range(len(X)):
    # iterate through columns of Y
    for j in range(len(Y[0]))

```

```
        for k in range(len(Y)):
            result[i][j] += X[i][k] * Y[k][j]
for r in result:
    print(r)
```

c)Write a program to solve a system of n linear equations in n variables using matrixinverse

```
x1 = 0
x2 = 0
x3 = 0
epsilon = 0.01
converged = False
x_old = np.array([x1, x2, x3])
print('Iteration results')
print(' k,  x1,  x2,  x3 ')
for k in range(1, 50):
    x1 = (14-3*x2+3*x3)/8
    x2 = (5+2*x1-5*x3)/(-8)
    x3 = (-8-3*x1-5*x2)/(-5)
    x = np.array([x1, x2, x3])
    # check if it is smaller than threshold
    dx = np.sqrt(np.dot(x-x_old, x-x_old))
    print("%d, %.4f, %.4f, %.4f"%(k, x1, x2, x3))
    if dx < epsilon:
        converged = True
        print('Converged!')
        break
    # assign the latest x value to the old value
    x_old = x
if not converged:
    print('Not converge, increase the # of iterations')
```

OUTPUT:

```
Iteration results
k,  x1,  x2,  x3
1, 1.7500, -1.0625, 1.5875
2, 2.7437, -0.3188, 2.9275
3, 2.9673, 0.4629, 3.8433
4, 3.0177, 1.0226, 4.4332
5, 3.0290, 1.3885, 4.8059
6, 3.0315, 1.6208, 5.0397
7, 3.0321, 1.7668, 5.1861
8, 3.0322, 1.8582, 5.2776
```

a) Finding if two sets of data have the same *mean* value.

By using the `==` operator:

We can check if two *sets* are equal or not by using the `==` operator. It will check if two *sets* are equal or not and return one *boolean* value. It returns `True` if the elements of the sets are equal. Else, it returns `False`. The order of the elements in the set is not considered for the comparison.

For example:

```
First_set = {'one', 'two', 'three'}
second_set = {'one', 'two', 'three'}

print(first_set == second_set)
```

B) Plotting data read from a file

```
import numpy as np
import matplotlib.pyplot as plt
with open("Alpha_Particle.txt") as f:
    data = f.read()
    data = data.split('\n')
    x = [row.split(' ')[0] for row in data]
    y = [row.split(' ')[1] for row in data]
    fig = plt.figure()
    ax1 = fig.add_subplot(111)
    ax1.set_title("Plot title")
    ax1.set_xlabel('x label')
    ax1.set_ylabel('y label')
    ax1.plot(x,y, c='r', label='the data')
    leg = ax1.legend()
    plt.show()
```

C) Fitting a function through a set of data points using *polyfit* function

Let's dive into a basic example of polynomial fitting with NumPy. Suppose we have some experimental data and we believe that it can be approximated by a quadratic polynomial. Here's how you could do it:

```
x = np.linspace(0, 10, 10)
y = np.random.normal(loc=1, scale=2, size=10) + 1 * x ** 2
c = np.polyfit(x, y, 2)
p = np.poly1d(c)
```

In the above code, we:

1. Created an array of x-values using `linspace`.
2. Generated corresponding y-values using a quadratic function with some added random noise.
3. Used `polyfit` to fit a second-degree polynomial to our x and y data, storing the coefficients in `c`.
4. Created a polynomial object `p` using `poly1d` that represents the fitted polynomial.

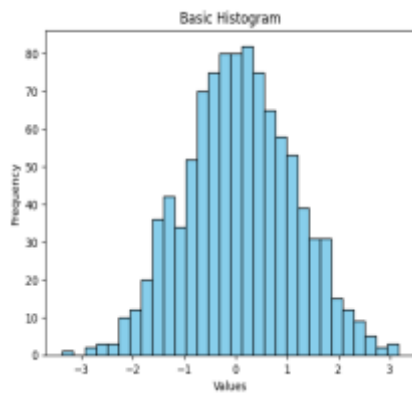
We can also visualize our results with the help of the `matplotlib` library. Here's how:

```
import matplotlib.pyplot as plt
plt.scatter(x, y, label='Data')
x_line = np.linspace(min(x), max(x), 100)
plt.plot(x_line, p(x_line), label='Fitted Polynomial', color='red')
plt.legend()
plt.show()
```


D)Plotting a histogram of a given data .

```
import matplotlib.pyplot as plt
import numpy as np
# Generate random data for the histogram
data = np.random.randn(1000)
# Plotting a basic histogram
plt.hist(data, bins=30, color='skyblue', edgecolor='black')
# Adding labels and title
plt.xlabel('Values')
plt.ylabel('Frequency')
plt.title('Basic Histogram')
# Display the plot
plt.show()
```

OUTPUT:



WEEK 5

1)The strings package

A) Read text from a file and print the number of lines, words and characters

Brute Force method

In this method, we will develop our own logic in a brute-force manner and take a text file as input and count the number of characters, words, spaces, and lines in the file. In this method, we will not use any inbuilt methods.

Algorithm

- Open the file in read mode using the open() function.
- Initialize variables to keep track of the character count, word count, space count, and line count.
- Read the file line by line using a loop.
- For each line, increment the line count.
- Increment the character count by the length of the line.
- Split the line into words using the split() method.
- Increment the word count by the number of words in the line.
- Calculate the space count by subtracting the number of words minus one from the length of the line.
- Close the file.
- Print the results.

b)Read text from a file and return a list of all n letter words beginning with a vowel

```
def read_words(test.txt):  
    open_file = open(words_file, 'r')  
    words_list = []  
    contents = open_file.readlines()  
    for i in range(len(contents)):  
        words_list.append(contents[i].strip('\n'))  
    return words_list  
    open_file.close()
```

Running this code produces this list:

```
['hello there how is everything ', 'thank you all', 'again', 'thanks a lot']
```

c)Finding a secret message hidden in a paragraph of text

I would like to open a file with an encrypted message in it, e.g. a passage from a book.

1. Load the passage as a single string.
2. Remove any characters that are not letters.
3. Break the string into a table, with at least 5 characters in each row.
4. Use a sequence of numbers 4,5,4,3,1,1,2,3,4,5 to index each row.

The result should be something like:

```
was h er
thise e
fret l
rj l
d o
```

This is what I have so far:

```
def cipher():
    f = open("cipher.txt", "r")
    fileString = f.readline()
    for line in fileString:
        lineSplit = line.split()
```

d)Plot a histogram of words according to their length from text read from a file.

```
import sys
import time
from string import punctuation as punc
from string import maketrans as trans
import operator
lengthDict = {}
def insert_Dict(word):
    lengthDict[word] = lengthDict.get(word, 0) + 1
def timing(f):
    def wrap(*args):
        time1 = time.time()
        ret = f(*args)
        time2 = time.time()
        print '%s function took %0.3f ms' % (f.func_name, (time2-time1)*1000.0)
    return return
```

```

return wrap
def roundup(x):
    return x if x % 100 == 0 else x + 100 - x % 100
@timing
def main():
    fileName = sys.argv[1]
    openFile = open(str(sys.argv[1]))
    readString = openFile.read()
    out = readString.translate(trans("", ""), punc)
    for word in readString.split():
        if word in punc:
            #print(word)
            insert_Dict(0)
    for word in out.split():
        #print(word)
        insert_Dict(len(word))
    maxDomain = max(lengthDict.keys())
    maxRange = max(lengthDict.values())
    roundedMaxRange = roundup(maxRange)
    for x in range(roundedMaxRange, 0, -20):
        newStr = ""
        if x % 100 == 0:
            newStr = newStr + str(x) + " -|"
        elif x % 20 == 0:
            newStr = newStr + "    |"
        else:
            newStr = newStr + "      "
    for domain in range(maxDomain + 1):
        if lengthDict.get(domain) >= x and lengthDict.has_key(domain) :
            newStr = newStr + ('***')
        else:
            newStr = newStr + ' '

    print(newStr)

    print('  -|--|--|--|--|--|--|--|--|--|--|--|--|--|')
    print '    0',
    for x in xrange(1, 16):
        print "",
        print x,
    print
if __name__ == '__main__':
    print("This program is being run by itself")
    main()
else:
    print('I am being imported from another module')

```

Week 6:

Installing OS on Raspberry Pi

- a) **Installation using PiImager**
- b) **Installation using image file**

- Downloading an Image
- Writing the image to an SD card
- using Linux
- using Windows
- Booting up

Follow the instructions given in the URL

<https://www.raspberrypi.com/documentation/computers/getting-started.html>

If you're new to the Raspberry Pi world, installing (or reinstalling) the operating system can be a daunting challenge. Hopefully, the Raspberry Pi Foundation does its best to make it as easy as possible for you. This article will show you the easiest and safest method to do it on your own.

Raspberry Pi OS can be installed on a new SD card from any computer, by using an application named “Raspberry Pi Imager”, created by the Raspberry Pi manufacturer. It includes all versions of Raspberry Pi OS, to flash it to the SD card in a few clicks.

Install Raspberry Pi Imager

The only tool you need on your computer to install Raspberry Pi OS on your SD card is Raspberry Pi Imager. It has been developed by the Raspberry Pi Foundation to make this process easier. No manual downloads or random apps are required anymore.

Here is how to get it on your computer:

- Go to the official Raspberry Pi website.
 - Visit [the “Software” page](#).
 - Download Raspberry Pi Imager for your system:
 - **It's available for Windows, macOS and Ubuntu. It's even possible to install it on an existing Raspberry Pi OS system if you already have one.**
 - It's even included in some RPI OS editions, so you may already have it.
 - Once downloaded, follow the installation steps you'd use for any application on your computer.
- On Windows, it's a double click on the file, and the next, next, next.

Week 7

Accessing GPIO pins using Python

a) Installing GPIO Zero library.

First, update your repositories list:

```
sudo apt update
```

Then install the package for Python 3:

GPIO Zero is installed by default in the [Raspberry Pi OS](#) desktop image, [Raspberry Pi OS](#) Lite image, and the [Raspberry Pi Desktop](#) image for PC/Mac, all available from raspberrypi.org. Follow these guides to installing on other operating systems, including for PCs using the [remote GPIO](#) feature.

1.1. Raspberry Pi

GPIO Zero is packaged in the apt repositories of Raspberry Pi OS, [Debian](#) and [Ubuntu](#). It is also available on [PyPI](#).

1.1. *apt*

First, update your repositories list:

```
pi@raspberrypi:~$ sudo apt update
```

Then install the package for Python 3:

```
pi@raspberrypi:~$ sudo apt install python3-gpiozero
```

or Python 2:

```
pi@raspberrypi:~$ sudo apt install python-gpiozero
```

1.1.2. *pip*

If you're using another operating system on your Raspberry Pi, you may need to use pip to install GPIO Zero instead. Install pip using [get-pip](#) and then type:

```
pi@raspberrypi:~$ sudo pip3 install gpiozero
```

or for Python 2:

```
pi@raspberrypi:~$ sudo pip install gpiozero
```

To install GPIO Zero in a virtual environment, see the [Development](#) page.

1.2. PC/Mac

In order to use GPIO Zero's remote GPIO feature from a PC or Mac, you'll need to install GPIO Zero on that computer using pip. See the [Configuring Remote GPIO](#) page for more information.

b)Blinking an LED connected to one of the GPIO pin

With the circuit created we need to write the Python script to blink the LED. Before we start writing the software we first need to install the Raspberry Pi GPIO Python module. This is a library that allows us to access the GPIO port directly from Python.

To install the Python library open a terminal and execute the following

```
$ sudo apt-get install python-rpi.gpio python3-rpi.gpio
```

With the library installed now open your favorite Python IDE (I recommend Thonny Python IDE more information about using it [here](#)).

Our script needs to do the following:

- Initialize the GPIO ports
- Turn the LED on and off in 1 second intervals

To initialize the GPIO ports on the Raspberry Pi we need to first import the Python library, the initialize the library and setup pin 8 as an output pin.

```
import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library
from time import sleep # Import the sleep function from the time module
GPIO.setwarnings(False) # Ignore warning for now
GPIO.setmode(GPIO.BOARD) # Use physical pin numbering
GPIO.setup(8, GPIO.OUT, initial=GPIO.LOW) # Set pin 8 to be an output pin and set initial value to low (off)
```

Next we need to turn the LED on and off in 1 second intervals by setting the output pin to either high (on) or low (off). We do this inside a infinite loop so our program keep executing until we manually stop it.

```
while True: # Run forever
    GPIO.output(8, GPIO.HIGH) # Turn on
    sleep(1) # Sleep for 1 second
```

```
GPIO.output(8, GPIO.LOW) # Turn off
```

```
sleep(1) # Sleep for 1 second
```

Combining the initialization and the blink code should give you the following full Python program:

```
import RPi.GPIO as GPIO # Import Raspberry Pi GPIO library
```

```
from time import sleep # Import the sleep function from the time module
```

```
GPIO.setwarnings(False) # Ignore warning for now
```

```
GPIO.setmode(GPIO.BOARD) # Use physical pin numbering
```

```
GPIO.setup(8, GPIO.OUT, initial=GPIO.LOW) # Set pin 8 to be an output pin and set initial value to low (off)
```

```
while True: # Run forever
```

```
    GPIO.output(8, GPIO.HIGH) # Turn on
```

```
    sleep(1) # Sleep for 1 second
```

```
    GPIO.output(8, GPIO.LOW) # Turn off
```

```
    sleep(1) # Sleep for 1 second
```

With our program finished, save it as `blinking_led.py` and run it either inside your IDE or in the console with:

```
$ python blinking_led.py
```

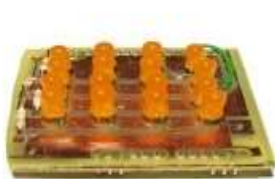
c)Adjusting the brightness of an LED

d)Adjust the brightness of an LED (0 to 100, where 100 means maximum brightness) using the in-built PWM wavelength.

Do you want to adjust the brightness of your Light-Emitting Diode (LED)? If we break it down to the most basic, there are two ways to change the brightness of an LED:

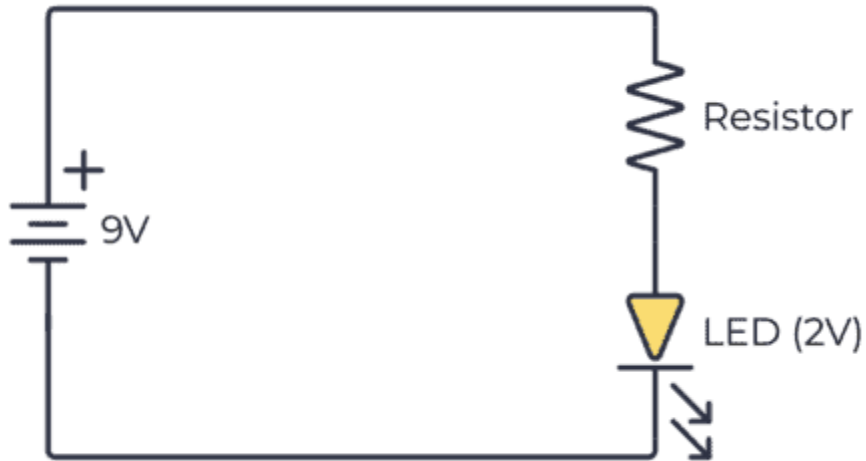
- Changing the resistance value.
- Turning it on and off fast (Using PWM).

Below I'll explain the two options and show you circuits you can build.

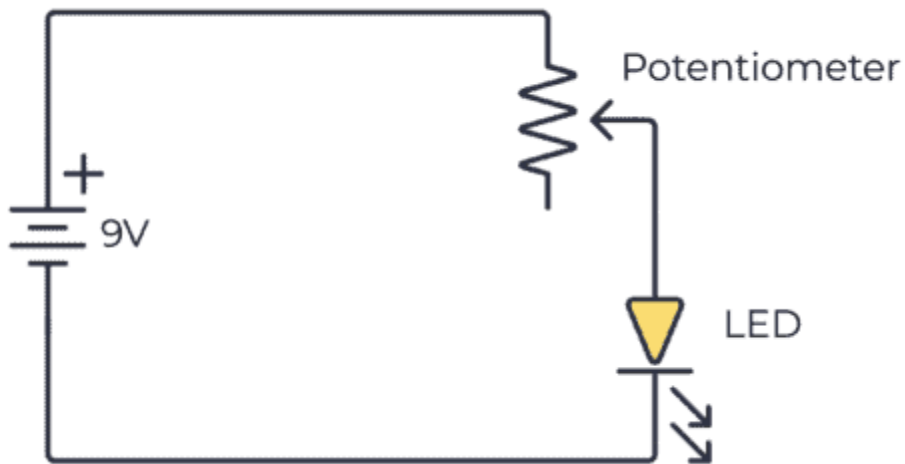


Option 1: Change the resistor value

The brightness of an LED depends on the current through it. A resistor in series with the LED sets the current. In the circuit below, you can see the basic setup for controlling an LED:



A [potentiometer](#) is a variable resistor that you can use to change the brightness of the LED. Just add the potentiometer in series with the LED. When you adjust the knob of the potentiometer, the brightness of the LED will change.



Week 8

Collecting Sensor Data DHT Sensor interface

- Connect the terminals of DHT GPIO pins of Raspberry Pi.
- Import the DHT library using `import Adafruit_DHT`
- Read sensor data and display it on screen.

We're going to use a special library called [adafruit blinka](#) (named after Blinka, the CircuitPython mascot) to provide the layer that translates the CircuitPython hardware API to whatever library the Linux board provides.

Installing CircuitPython Libraries on Raspberry Pi or BeagleBone Black
If you haven't set up your Raspberry Pi or BeagleBone Black for running CircuitPython libraries yet, follow our guide and come back to this page when you've completed the steps listed on the page and verified that your setup is working:

- [Setup your Linux Board for using CircuitPython Libraries](#)

Installing the CircuitPython-DHT Library

You'll also need to install a library to communicate with the DHT sensor. Since we're using Adafruit Blinka (CircuitPython), we can install CircuitPython libraries straight to our small linux board. In this case, we're going to install the CircuitPython_DHT library. This library works with both the DHT22 and DHT11 sensors.

Run the following command to **install the [CircuitPython-DHT library](#)**:

```
pip3 install adafruit-circuitpython-dht
```

```
sudo apt-get install libgpiod2
```

Testing the CircuitPython DHT Library

To make sure you've installed everything correctly, we're going to test that we can read values from the DHT sensor connected to your device.

Create a new file called **dht_simpletest.py** with **nano** or your favorite text editor and put the following in:

[Download Project Bundle](#)

[Copy Code](#)

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries  
# SPDX-License-Identifier: MIT
```

```
import time  
import board  
import adafruit_dht
```

```

# Initial the dht device, with data pin connected to:
dhtDevice = adafruit_dht.DHT22(board.D18)

# you can pass DHT22 use_pulseio=False if you wouldn't like to use pulseio.
# This may be necessary on a Linux single board computer like the Raspberry Pi,
# but it will not work in CircuitPython.
# dhtDevice = adafruit_dht.DHT22(board.D18, use_pulseio=False)

while True:
    try:
        # Print the values to the serial port
        temperature_c = dhtDevice.temperature
        temperature_f = temperature_c * (9 / 5) + 32
        humidity = dhtDevice.humidity
        print(
            "Temp: {:.1f} F / {:.1f} C  Humidity: {}% ".format(
                temperature_f, temperature_c, humidity
            )
        )

    except RuntimeError as error:
        # Errors happen fairly often, DHT's are hard to read, just keep going
        print(error.args[0])
        time.sleep(2.0)
        continue
    except Exception as error:
        dhtDevice.exit()
        raise error

    time.sleep(2.0)

```