



# **MASTER OF COMPUTER APPLICATIONS**

## **SEMESTER 1**

### **DATA VISUALIZATION**

# Unit 7

## Introduction to Python for Data Visualization

### Table of Contents

SL No	Topic	Fig No / Table / Graph	SAQ / Activity	Page No
1	<a href="#">Introduction</a>	-	-	3
1.1	<a href="#">Learning Objectives</a>	-	-	
2	<a href="#">Overview</a>	-	-	4-11
3	<a href="#">Setup</a>	-	-	12-22
4	<a href="#">Basics of Python</a>	-	-	22-27
5	<a href="#">Visualisation Libraries</a>	-	-	27-35
6	<a href="#">Summary</a>	-	-	36
7	<a href="#">Questions</a>	-	<a href="#">1</a>	36-37
8	<a href="#">Answers</a>	-	-	38

## 1. INTRODUCTION

In this chapter, we will journey through the fundamental concepts and tools that lay the foundation for data analysis and exploration using Python. Whether you are a beginner or an experienced programmer, this chapter will provide a comprehensive overview of the essential elements necessary to navigate the world of data science effectively.

We will start by offering you an overview of the chapter's contents, giving you a roadmap of the topics we will cover. From there, we will guide you through the initial setup, ensuring your Python environment is ready. This setup is crucial as it forms the backbone of your data analysis endeavours.

We will explore Python programming basics as we delve deeper into the chapter. For those new to Python, this section will provide you with a solid foundation to build. It will be a helpful refresher if you are already familiar with Python.

Visualisation is a powerful tool in data analysis, allowing you to gain insights and convey information effectively. In this chapter, we will introduce you to visualisation libraries in Python, showing you how to create compelling graphs and charts to represent your data.

By the end of this chapter, you will be equipped with the knowledge and skills necessary to begin your journey into data analysis with Python. Whether interested in data visualisation, analysis, or both, the concepts covered here will be invaluable as you explore and analyse real-world data sets in the coming chapters. So, let's dive in and get started on this exciting adventure in Python-based data science.

### 1.1 Learning objectives

*By the end of this chapter, you will be able to:*

- ❖ *Learn the fundamental concepts of Python.*
- ❖ *Explore the setup process of python.*
- ❖ *Interpret the various python visualisation libraries.*
- ❖ *Describe the concepts of python Ecosystem and Interactive python notebooks.*
- ❖ *Explain the best practices to follow for efficient data visualisation process.*

## 2. OVERVIEW

Data visualisation is the graphical representation of data to extract insights, patterns, and trends. Thanks to its versatile libraries and user-friendly syntax, Python has emerged as a prominent language for data visualisation. In this detailed exploration, we'll delve into data visualisation with Python, discussing its advantages, disadvantages, and applications across various fields.

Advantages of Python for data science and visualisation:

### 1. *Rich Ecosystem:*

- Python offers a vast and diverse ecosystem of libraries and tools that cater to a wide range of data visualisation needs. These libraries provide pre-built functions, classes, and modules for creating charts, plots, and graphs.
- For example, Matplotlib is a comprehensive library for creating static and interactive visualisations, Seaborn specialises in statistical plotting, Plotly excels in web-based interactive visualisations, and Bokeh provides tools for creating interactive web applications with data-driven visualisations.
- This rich library ecosystem allows data scientists and analysts to choose the most suitable tools for their specific visualisation tasks. Whether you need to create basic charts or complex interactive dashboards, Python has a library.

### 2. *Ease of Use:*

- Python is renowned for its readability and straightforward syntax. Its code is often described as "Pythonic," emphasising simplicity and clarity.
- This ease of use is particularly advantageous for beginners and experienced programmers. Thanks to its clean and intuitive syntax, beginners can quickly grasp the basics of Python. It's an excellent language for those new to programming and data analysis.
- On the other hand, experienced developers find Python's simplicity refreshing and conducive to rapid development. It allows them to focus on the logic and insights behind the data rather than wrestling with complex syntax.

### 3. *Integration:*

- Python seamlessly integrates with other data-related libraries and tools, making it a versatile choice for data manipulation and analysis.
- Libraries like NumPy and Pandas provide powerful data structures and functions for data cleaning, transformation, and manipulation. Python's native data types, such as lists and dictionaries, are also easy to work with, simplifying data preparation.
- Regarding visualisation, Python's integration with libraries like Matplotlib, Seaborn, and Plotly allows you to generate charts and plots from your data effortlessly. You can use these with data manipulation libraries to create end-to-end data analysis workflows.
- Additionally, Python's ability to interface with databases, web APIs, and other data sources further enhances its integration capabilities.

### 4. *Community and Support:*

- Python benefits from a vast and active community of users, developers, and data enthusiasts worldwide. This community actively contributes to the Python ecosystem by creating libraries, sharing knowledge, and supporting fellow users.
- Resources: Python users can access many online resources, including forums, discussion boards (e.g., Stack Overflow), and dedicated Python communities. These resources are valuable for troubleshooting, seeking guidance, and sharing knowledge.
- Tutorials and Documentation: Python libraries and tools typically have extensive documentation and tutorials. These resources offer step-by-step guidance on using Python for data visualisation. They cater to users of all skill levels, from beginners to experts.
- Quick Problem-Solving: With a large and active user base, Python users can quickly find solutions to common problems or challenges during data visualisation. The community's collective knowledge and experience serve as a valuable resource.
- Open Source Collaboration: Python's open-source nature encourages collaboration among developers. Users and developers can contribute to libraries, report bugs,



and suggest improvements, fostering continuous development and enhancement of the ecosystem.

5. *Customisation:*

- Colors and Fonts: Users can select colour palettes, font styles, and font sizes to match their project's branding or to enhance readability. Customising colours also helps convey meaning or highlight specific data points.
- Plot Size and Layout: Python libraries typically provide control over the dimensions of the plot or chart. Users can adjust the width, height, and layout of visualisations to fit various media, such as reports, presentations, or web pages.
- Annotations and Labels: Customising labels, titles, annotations, and legends adds context to the visualisation. Users can explain data points, highlight trends, and provide additional information for clarity.
- Customisation empowers users to create aesthetically pleasing and informative visualisations that communicate their data-driven insights to a broader audience.

6. *Interactivity:*

- Python libraries like Plotly and Bokeh offer interactivity features beyond static visuals. Interactivity enhances the user experience and allows for dynamic exploration of data.
- Key interactive features include:
  - Zooming and Panning: Users can zoom in to view specific details or pan across large datasets. These features are beneficial when dealing with intricate charts or maps.
  - Tooltips: Interactive tooltips provide on-hover information, such as data values or additional context. Users can quickly access details without cluttering the visualisation.
  - Filters and Selections: Interactive visualisations often allow users to filter or select specific data subsets. This dynamic interaction helps users focus on areas of interest or investigate patterns.
  - Animations: Animations in data visualisation, often supported by Python libraries, help convey changes over time or highlight transitions in data.

- Interactivity turns static charts into dynamic exploration, analysis, and storytelling tools. It engages users and enables them to derive deeper insights from the data.

In summary, Python's strong community and support system, customisation capabilities, and interactivity features collectively make it a powerful and user-friendly platform for data visualisation. Users benefit from a wealth of resources, the ability to tailor visuals to their needs, and the means to create engaging and interactive data presentations. These advantages contribute to Python's popularity as a go-to choice for data professionals seeking to convey insights effectively through visualisation.

Disadvantages of Python for data science and visualisation:

*1. Performance Limitations:*

- Python, while versatile, may not be the most performant option for handling enormous datasets or real-time data streaming. This limitation primarily arises from Python's interpreted nature and the Global Interpreter Lock (GIL) in the CPython implementation, which can hinder multithreading performance.
- In scenarios where processing speed is critical, such as high-frequency trading, real-time sensor data analysis, or processing massive datasets, languages like C++ or specialised tools like Apache Spark or Apache Flink may be more suitable. These languages and tools offer lower-level control and optimisations that can significantly boost performance.

*2. Learning Curve:*

- Python is often lauded for its beginner-friendly syntax, but mastering data visualisation libraries and their intricacies can still be time-consuming, especially for newcomers.
- Using Python libraries like Matplotlib, Seaborn, Plotly, or Bokeh effectively requires understanding their specific syntax, options, and best practices. This learning curve can be steeper for individuals new to programming or data analysis.
- To mitigate this disadvantage, newcomers can leverage extensive online resources, tutorials, and courses tailored to Python data visualisation. These resources can help learners navigate the complexities and accelerate their proficiency.

### 3. *Limited 3D Visualisation:*

- Python's 3D visualisation capabilities are somewhat limited compared to specialised 3D visualisation software or languages like MATLAB, which excel in creating complex 3D plots and animations.
- While Python libraries like Matplotlib and Plotly support basic 3D plotting, creating intricate 3D visualisations with advanced features or scientific simulations may require additional effort and customisation.
- Users seeking advanced 3D visualisation may need to explore alternatives or extend Python libraries with custom code to achieve their desired results.

### 4. *Browser Compatibility:*

- Interactive visualisations created with Python libraries like Plotly or Bokeh, especially those deployed in web applications, may encounter compatibility issues with specific web browsers.
- These issues can arise due to variations in browser rendering engines, JavaScript compatibility, or specific browser settings.
- To address compatibility concerns, developers may need to test their interactive visualisations across different browsers and versions, implement browser-specific workarounds, or provide alternative non-interactive visualisation options for users experiencing problems.

In summary, while Python is a versatile and popular choice for data visualisation, it has some disadvantages, including potential performance limitations, a learning curve for mastering libraries, limited 3D visualisation capabilities, and the need to address browser compatibility issues in interactive web-based visualisations. Understanding these limitations allows users to make informed decisions and consider alternative approaches or tools when facing specific challenges in data visualisation projects.



---

**Applications of data visualisation with Python in various fields in more detail:****1. Business and Finance:**

- Python visualisations are invaluable for analysing financial data, tracking market trends, and making informed investment decisions in the business and finance sector. Financial analysts and traders use Python to create visualisations for:
- Portfolio Analysis: Visualising the performance of investment portfolios and assessing risk.
- Market Trends: Tracking stock prices, currency exchange rates, and commodity prices.
- Risk Assessment: Analysing risk factors and simulating risk scenarios.
- Financial Dashboards: Create interactive dashboards to monitor real-time financial data.

**2. Healthcare:**

- In healthcare, data visualisation with Python plays a pivotal role in managing and analysing vast medical data. Healthcare professionals leverage Python for:
- Patient Record Visualisation: Visualising patient records aids diagnosis and treatment planning.
- Clinical Trial Results: Presenting the outcomes of clinical trials through clear and informative graphics.
- Disease Outbreak Monitoring: Tracking disease outbreaks and epidemiological data.
- Medical Imaging: Visualising medical images such as MRI, X-rays, and CT scans.

**3. Marketing and Sales:**

- Python visualisations are essential for marketing and sales professionals to gain insights into consumer behaviour and campaign performance. This includes:
- Marketing Analytics: Analysing marketing campaign data to assess ROI and optimise strategies.
- Customer Segmentation: Visualising customer segments for targeted marketing efforts.
- Sales Forecasting: Predicting future sales trends and demand patterns.

- Sales Performance: Monitoring sales performance through charts and dashboards.

#### 4. *Education:*

- In the education sector, Python visualisations enhance the learning experience by simplifying complex concepts and making educational materials more engaging. Educational applications include:
  - Scientific Data Visualization: Helping students visualise scientific data, experiments, and simulations.
  - Statistical Concepts: Illustrating statistical concepts and probability distributions.
  - Interactive Educational Materials: Creating interactive educational modules and simulations.

#### 5. *Environmental Science:*

- Environmental scientists and researchers use Python visualisations to study, analyse, and communicate data related to the environment. Key applications include:
  - Weather Patterns: Visualising weather data, climate patterns, and forecasts.
  - Climate Change Analysis: Analysing and presenting climate change data and trends.
  - Ecological Trends: Visualising ecological data, habitat mapping, and biodiversity studies.

#### 6. *Social Sciences:*

- Researchers in the social sciences leverage Python visualisations to analyse survey data, study demographics, and visualise social trends. Applications encompass:
  - Survey Analysis: Visualising survey responses and conducting sentiment analysis.
  - Demographic Studies: Analysing population demographics and changes over time.
  - Social Trend Visualisation: Illustrating social and cultural trends through data.

#### 7. *Government and Public Policy:*

- Government agencies and policymakers use Python visualisations to inform decision-making and communicate data-driven insights to the public. This includes:
  - Public Health Data: Visualising public health statistics, disease spread, and vaccination coverage.

- Economic Data: Presenting economic indicators, GDP growth, and employment figures.
- Infrastructure Planning: Visualising infrastructure data for urban planning and development.

#### 8. *Entertainment and Gaming:*

- The entertainment and gaming industry employs Python to create interactive visualisations within games and analyse player behaviour. Applications encompass:
- In-Game Visuals: Integrating data-driven visuals within video games to enhance the gaming experience.
- Player Behavior Analysis: Analysing player interactions, preferences, and gameplay patterns.

#### 9. *Research and Academia:*

- Researchers across various fields, including physics, biology, and engineering, use Python visualisations to present their findings effectively. This includes:
- Experimental Results: Visualising experimental data, laboratory findings, and simulations.
- Academic Publications: Creating informative and visually appealing figures and charts for research papers and presentations.

Python's versatility and powerful data visualisation capabilities have led to widespread adoption across many fields. It enables professionals and researchers to gain insights, communicate data-driven narratives, and make informed decisions across diverse domains, ultimately contributing to advancements in various industries.

### 3. SETUP

Installing Python on Windows takes a series of few easy steps.

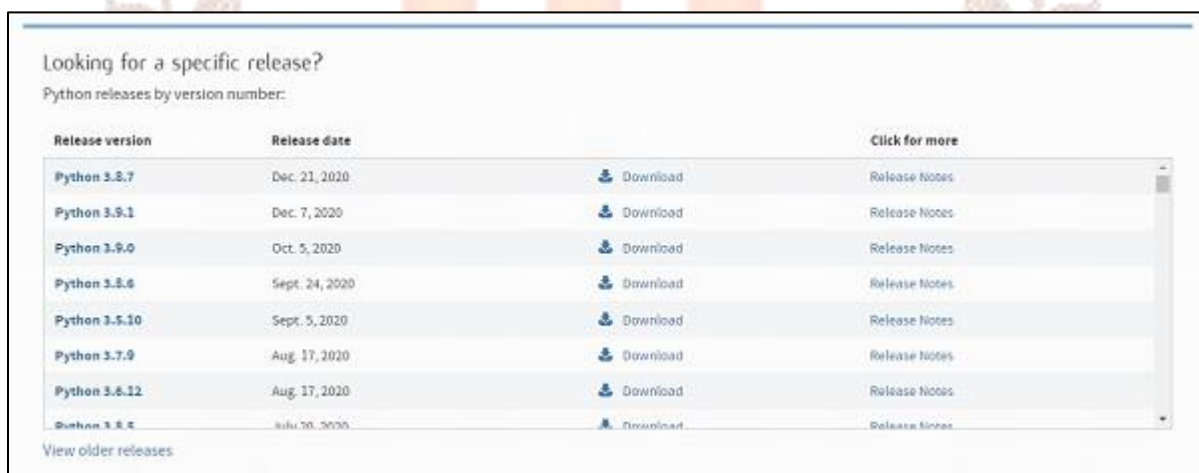
#### Step 1 – Select the Version of Python to Install

Python has various versions available, with differences between the syntax and the workings of different language versions. We need to choose the version which we want to use or need. Various versions of Python 2 and Python 3 are available.

#### Step 2 – Download Python Executable Installer

On the web browser, on the official site of Python ([www.python.org](http://www.python.org)), move to the Download for Windows section.

All the available versions of Python will be listed. Select the version you require and click on Download. Let's suppose we chose the Python 3.9.1 version.



Looking for a specific release?  
Python releases by version number:

Release version	Release date	Click for more	
Python 3.8.7	Dec. 21, 2020	Download	Release Notes
Python 3.9.1	Dec. 7, 2020	Download	Release Notes
Python 3.9.0	Oct. 5, 2020	Download	Release Notes
Python 3.8.6	Sept. 24, 2020	Download	Release Notes
Python 3.8.5	Sept. 5, 2020	Download	Release Notes
Python 3.7.9	Aug. 17, 2020	Download	Release Notes
Python 3.8.12	Aug. 17, 2020	Download	Release Notes
Python 3.8.2	July 20, 2020	Download	Release Notes

[View older releases](#)

Various executable installers with different operating system specifications will be visible when clicking download. Choose the installer that suits your operating system and download the installer. Let's suppose we select the Windows installer(64 bits).

The download size is less than 30MB.

Version	Operating System	Description	MDS Sum	File Size	GPG
Gzipped source tarball	Source release		429ae95d24227f8fa1560684fad61ca7	25372998	SIG
XZ compressed source tarball	Source release		61981498e75ac8f00adcb906281fadb6	18897104	SIG
macOS 64-bit intel installer	Mac OS X	for macOS 10.9 and later	74f5cc5b5783ce8fb2ca55f11f3f0699	29795899	SIG
macOS 64-bit universal2 installer	Mac OS X	for macOS 10.9 and later, including macOS 11 Big Sur on Apple Silicon (experimental)	8b19748473609241e60aa3618bbaf3ed	37451735	SIG
Windows embeddable package (32-bit)	Windows		96c6fa81fe8b650e68c3dd41258ae317	7571141	SIG
Windows embeddable package (64-bit)	Windows		e70e5c22432d8f57a497cde5ec2e5ce2	8402333	SIG
Windows help file	Windows		c49d9b6ef88c0831ed0e2d39bc42b316	8787443	SIG
Windows installer (32-bit)	Windows		dde210ea04a31c27488605a9e7cd297a	27126136	SIG
Windows installer (64-bit)	Windows	Recommended	b3fce2ed8bc315ad22bc49eae48a94457	28204528	SIG

### Step 3 – Run Executable Installer

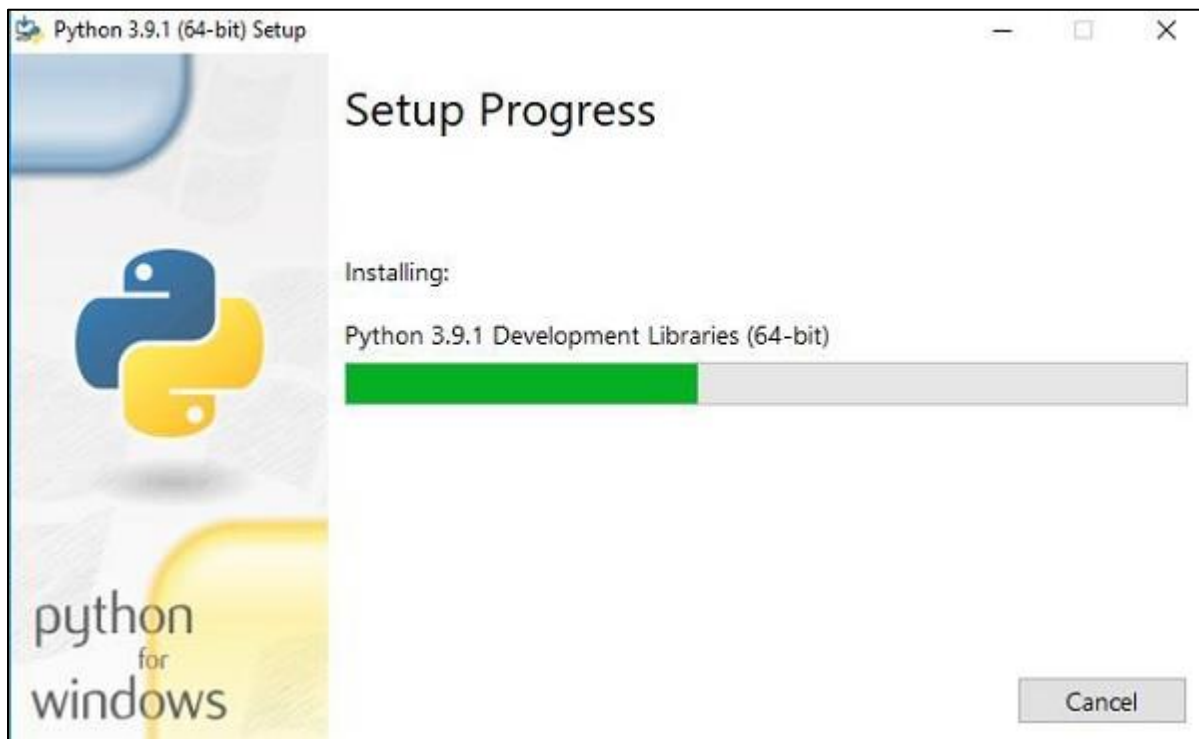
We downloaded the Python 3.9.1 Windows 64-bit installer.

Run the installer. Make sure to select both the checkboxes at the bottom and then click Install New.

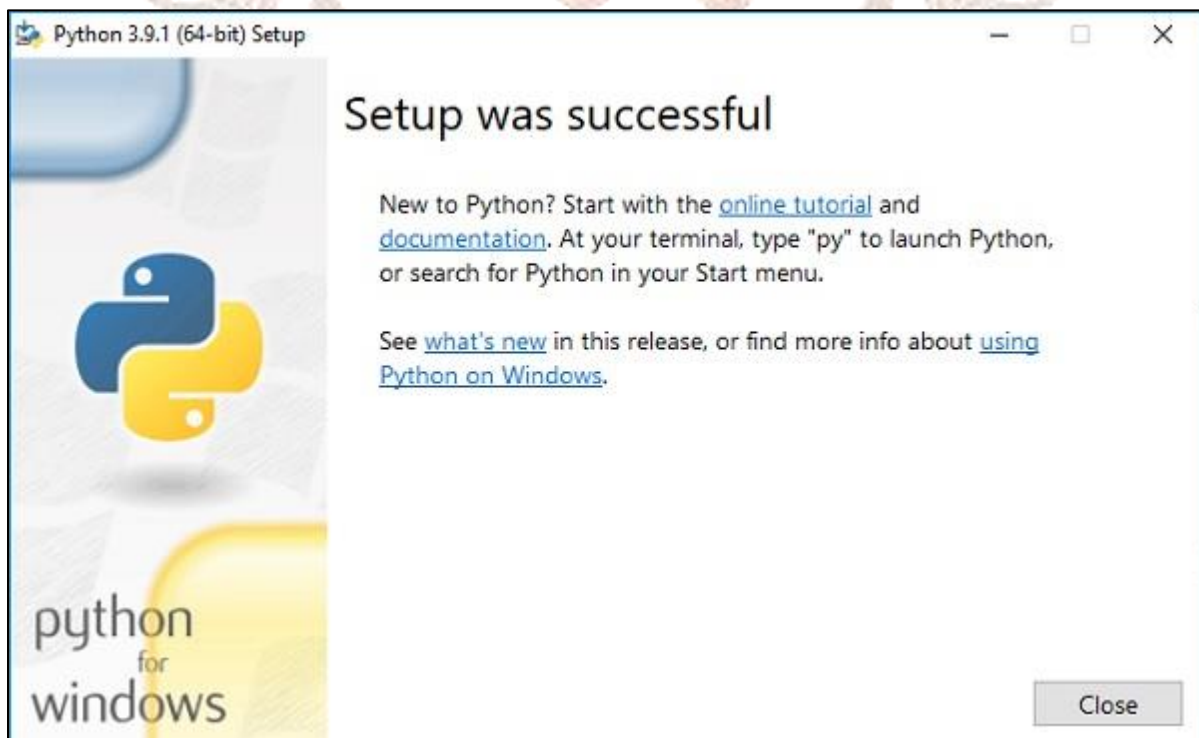


On clicking Install Now, The installation process starts.





The installation process will take a few minutes to complete, and once the installation is successful, the following screen will be displayed.

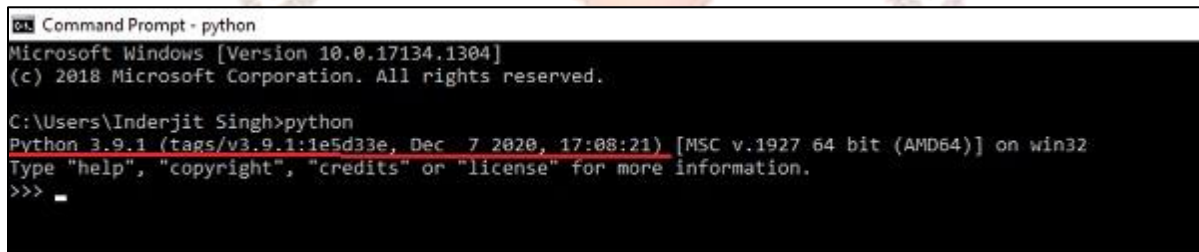


#### Step 4 – Verify Python is installed on Windows

This is to ensure that Python is successfully installed on your system. Follow the given steps

–

- Open the command prompt.
- Type 'python' and press enter.
- The version of the Python you have installed will be displayed if the Python is successfully installed on your Windows.



```
Command Prompt - python
Microsoft Windows [Version 10.0.17134.1304]
(c) 2018 Microsoft Corporation. All rights reserved.

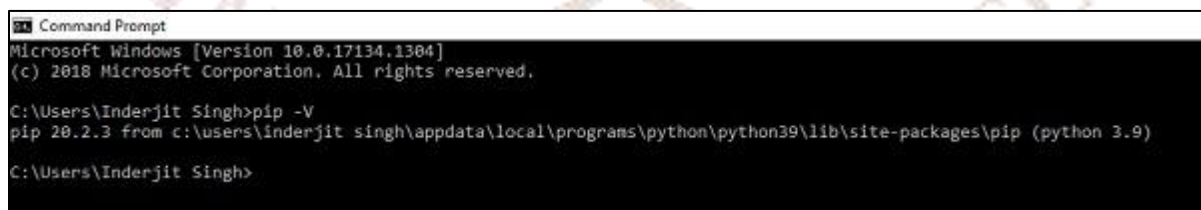
C:\Users\Inderjit Singh>python
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> _
```

#### Step 5 – Verify pip was installed

**Pip** is a powerful package management system for Python software packages. Thus, make sure that you have it installed.

To verify if the pip was installed, follow the given steps –

- Open the command prompt.
- Enter pip -V to check if pip was installed.
- The following output appears if the pip is installed successfully.



```
Command Prompt
Microsoft Windows [Version 10.0.17134.1304]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Inderjit Singh>pip -V
pip 20.2.3 from c:\users\inderjit singh\appdata\local\programs\python\python39\lib\site-packages\pip (python 3.9)

C:\Users\Inderjit Singh>
```

We have successfully installed Python and pip on our Windows system.

Steps to install Python on iOS:

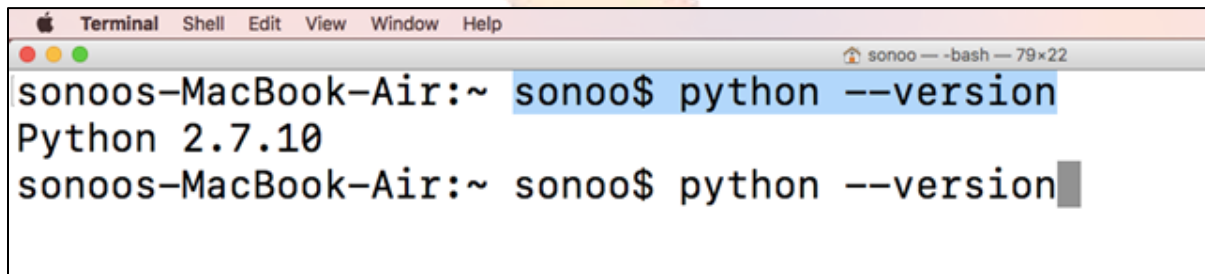
The following steps are used while installing Python3 on MacOS.

### 1) Checking Python's version on the system

We can check which version of the Python is currently installed on our system. Generally, Python 2.7 is installed by default.

Let's see how we can do it.

1. `$ python -version`

A screenshot of a macOS Terminal window. The title bar shows 'Terminal' with menu options 'Shell', 'Edit', 'View', 'Window', and 'Help'. The status bar at the top right indicates 'sonoo -- -bash -- 79x22'. The terminal text shows the prompt 'sonoos-MacBook-Air:~ sonoo\$' followed by the command 'python --version' which returns 'Python 2.7.10'. The prompt is then repeated with the same command, and the cursor is at the end of the line.

```
sonoos-MacBook-Air:~ sonoo$ python --version
Python 2.7.10
sonoos-MacBook-Air:~ sonoo$ python --version
```

It shows that **Python 2.7.10** is installed on the computer quite often.

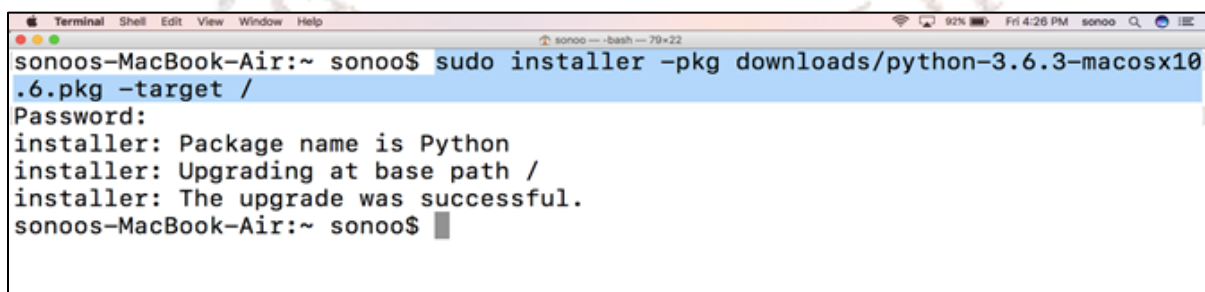
### 2) Download Python 3.6.3

To install Python 3.6.3, we must download the latest version from its official website: <https://www.python.org/downloads/>. The file is downloaded in **.pkg** format and can be installed using the **Installer** command.

### 3) Install Python 3.6.3

Since the downloaded file is already in a **.pkg** format, no mounting is required, and We can use the installer command to install Python 3.6.3.

Let's see how we can do it.

A screenshot of a macOS Terminal window. The title bar shows 'Terminal' with menu options 'Shell', 'Edit', 'View', 'Window', and 'Help'. The status bar at the top right shows system icons for Wi-Fi, battery (92%), time (Fri 4:26 PM), and user (sonoo). The terminal text shows the prompt 'sonoos-MacBook-Air:~ sonoo\$' followed by the command 'sudo installer -pkg downloads/python-3.6.3-macosx10.6.pkg -target /'. It then prompts for a password, followed by the output: 'installer: Package name is Python', 'installer: Upgrading at base path /', and 'installer: The upgrade was successful.' The prompt is then repeated.

```
sonoos-MacBook-Air:~ sonoo$ sudo installer -pkg downloads/python-3.6.3-macosx10
.6.pkg -target /
Password:
installer: Package name is Python
installer: Upgrading at base path /
installer: The upgrade was successful.
sonoos-MacBook-Air:~ sonoo$
```

Since the installer is used with superuser permissions, sudo forces the terminal to prompt the user to fill in the admin password, and the process installs Python 3.6.3 to the root directory, which is mentioned with the target option.

#### 4) Verify Python3

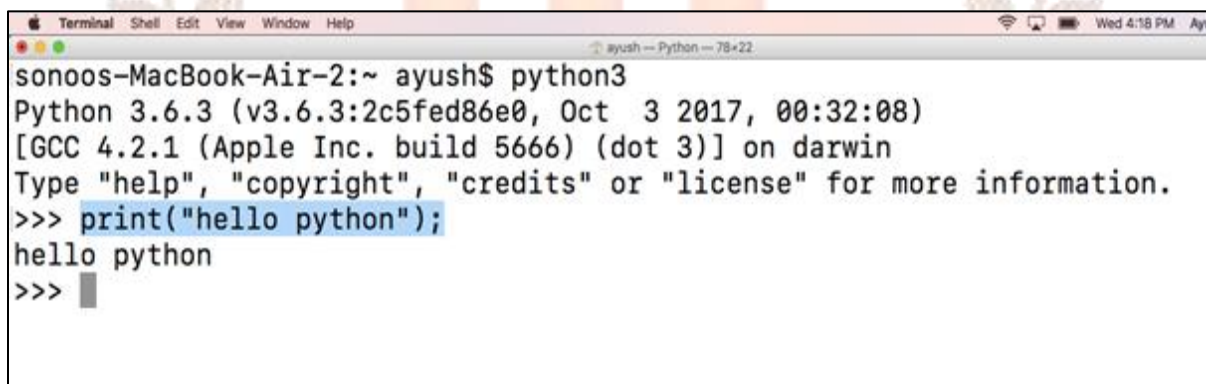
We can use the Python- version command to check which Python version is installed on the machine. Since the installed version is Python 2.7.10 by default, it shows Python 2.7.10. but it gives us the flexibility to check the version of Python 3 on our computer.

Let's see how we can use Python 3 to check which version of Python 3 is running.

1. \$ python -version

#### 5) Working on Python's script mode

We simply type python3 on the terminal to work on the Python command line. **Python shell opens where we can run Python statements such as print statements as we did here.**



```
Terminal Shell Edit View Window Help
ayush -- Python -- 78x22
sonoos-MacBook-Air-2:~ ayush$ python3
Python 3.6.3 (v3.6.3:2c5fed86e0, Oct 3 2017, 00:32:08)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello python");
hello python
>>> █
```

To run a Python file (.py) on the terminal, we simply type the file name, and the file will be interpreted.

We have installed Python3 on our MacOS.

Installing Python on a Linux system

Most of the Linux OS has Python pre-installed. To check if your device is pre-installed with Python or not, just go to the terminal using **Ctrl+Alt+T**

Now run the following command:

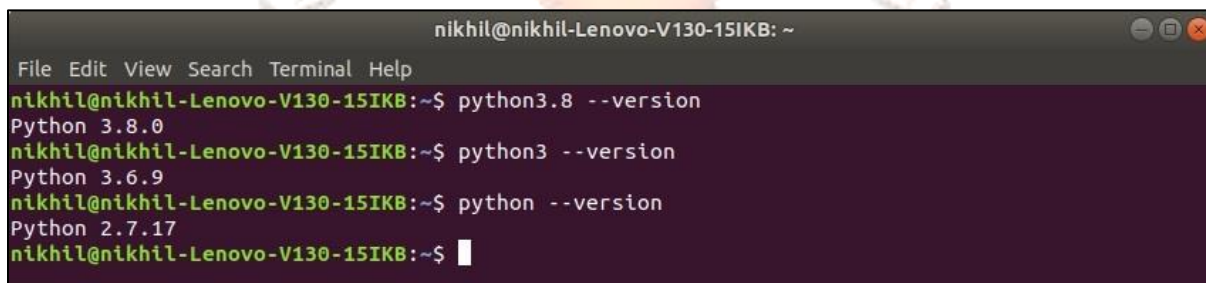
**For Python2**

```
python --version
```

**For Python3.x**

```
python3.x --version
```

If Python is already installed, it will generate a message with the Python version available.



```
nikhil@nikhil-Lenovo-V130-15IKB: ~  
File Edit View Search Terminal Help  
nikhil@nikhil-Lenovo-V130-15IKB:~$ python3.8 --version  
Python 3.8.0  
nikhil@nikhil-Lenovo-V130-15IKB:~$ python3 --version  
Python 3.6.9  
nikhil@nikhil-Lenovo-V130-15IKB:~$ python --version  
Python 2.7.17  
nikhil@nikhil-Lenovo-V130-15IKB:~$
```

**Download and Install Python:**

Before starting with the installation process, you need to download it. For that, all versions of Python for Linux are available on [python.org](https://python.org).



Python 3.8.1	Dec. 18, 2019	<a href="#">Download</a>	<a href="#">Release Notes</a>
Python 3.7.6	Dec. 18, 2019	<a href="#">Download</a>	<a href="#">Release Notes</a>
Python 3.6.10	Dec. 18, 2019	<a href="#">Download</a>	<a href="#">Release Notes</a>
Python 3.5.9	Nov. 2, 2019	<a href="#">Download</a>	<a href="#">Release Notes</a>
Python 3.5.8	Oct. 29, 2019	<a href="#">Download</a>	<a href="#">Release Notes</a>
Python 2.7.17	Oct. 19, 2019	<a href="#">Download</a>	<a href="#">Release Notes</a>
Python 3.7.5	Oct. 15, 2019	<a href="#">Download</a>	<a href="#">Release Notes</a>
Python 3.8.0	Oct. 14, 2019	<a href="#">Download</a>	<a href="#">Release Notes</a>

[View older releases](#)

Download the required version and follow the further instructions for the installation process.

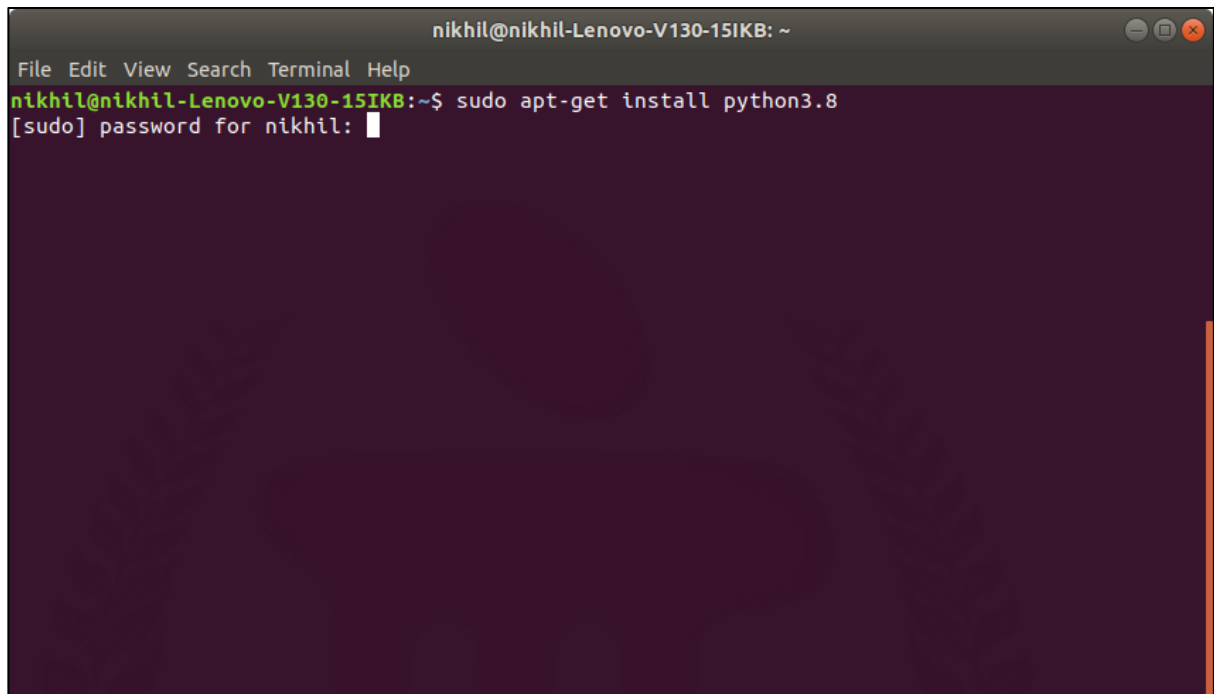
**Beginning the installation.**

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

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

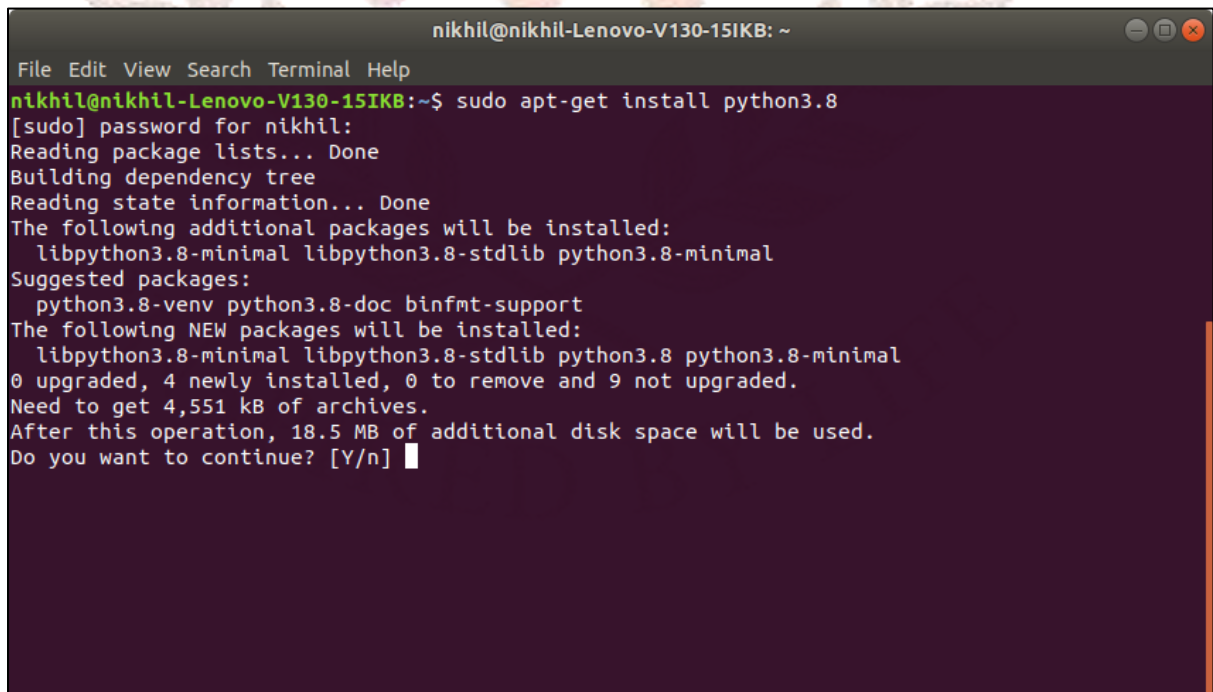


- **Getting Started:**



```
nikhil@nikhil-Lenovo-V130-15IKB: ~  
File Edit View Search Terminal Help  
nikhil@nikhil-Lenovo-V130-15IKB:~$ sudo apt-get install python3.8  
[sudo] password for nikhil: 
```

- **Assigning DiskSpace:**



```
nikhil@nikhil-Lenovo-V130-15IKB: ~  
File Edit View Search Terminal Help  
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] 
```

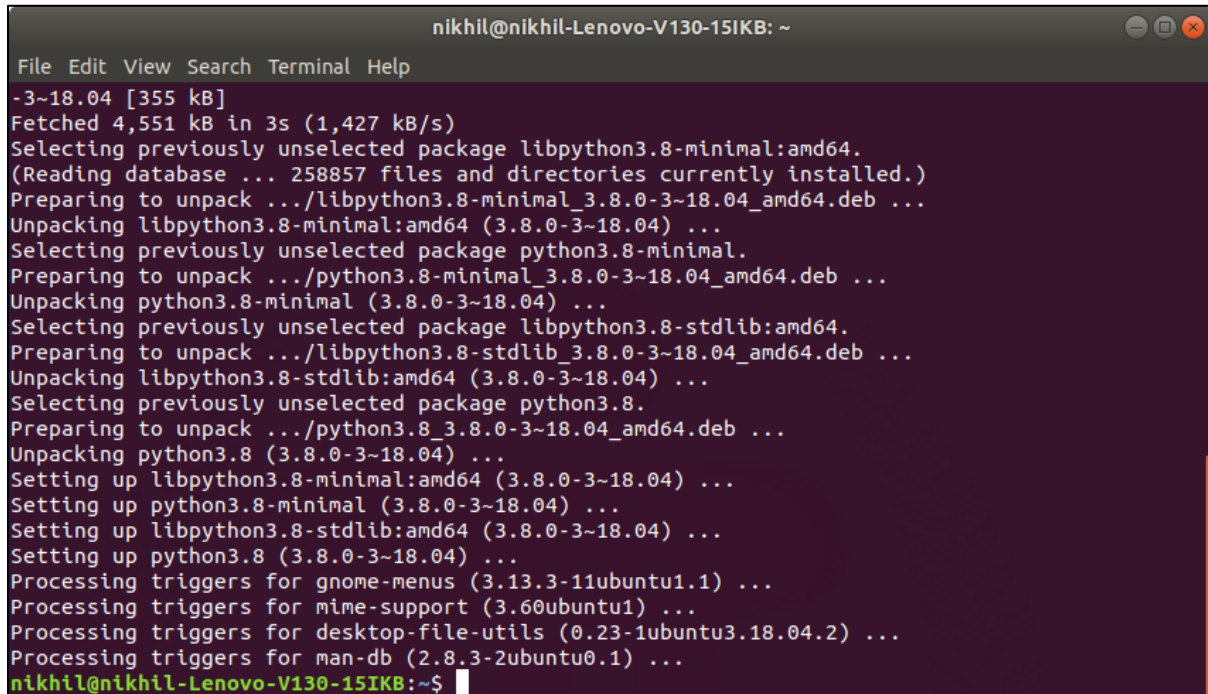
- **Fetching and Installing Packages:**

```
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://in.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 libpython3.8-minimal
amd64 3.8.0-3~18.04 [704 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu bionic-updates/universe amd64 python3.8-minimal amd
64 3.8.0-3~18.04 [1,816 kB]
Get:3 http://in.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://in.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 (3.13.3-11ubuntu1.1) ...
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) ...
```

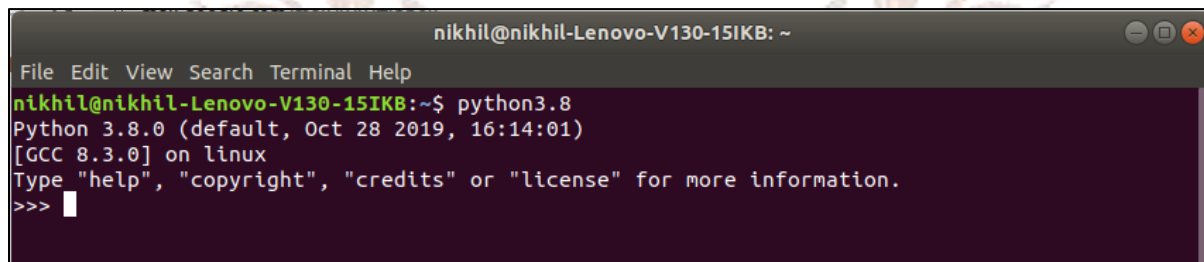
- **Finished Installation:**



```
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 (3.13.3-11ubuntu1.1) ...  
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 into your terminal.

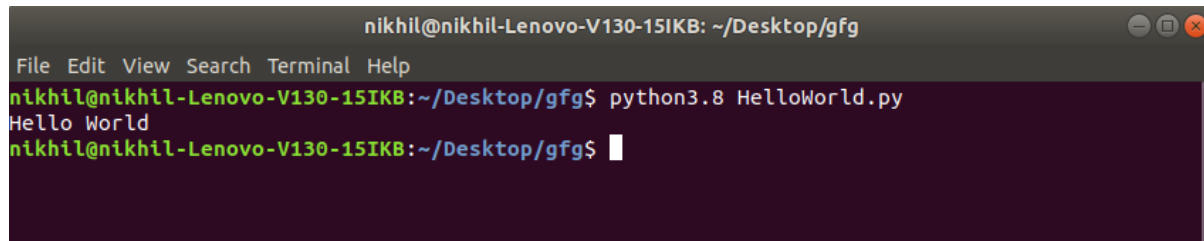
python3.8



```
nikhil@nikhil-Lenovo-V130-15IKB: ~  
File Edit View Search Terminal Help  
nikhil@nikhil-Lenovo-V130-15IKB:~$ python3.8  
Python 3.8.0 (default, Oct 28 2019, 16:14:01)  
[GCC 8.3.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>>
```

Let's consider a simple Hello World Program.

```
# Python program to print  
  
# Hello World  
  
print("Hello World")
```

**Output:**A screenshot of a terminal window with a dark background. The title bar at the top reads 'nikhil@nikhil-Lenovo-V130-15IKB: ~/Desktop/gfg'. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the command 'python3.8 HelloWorld.py' being executed, followed by the output 'Hello World' on the next line. The prompt 'nikhil@nikhil-Lenovo-V130-15IKB:~/Desktop/gfg\$' is visible at the bottom.

## 4. BASICS OF PYTHON

**Python Syntax:**

Python uses a clean and easy-to-read syntax, making it an excellent choice for beginners and experienced programmers. Some critical syntax rules include:

- **Indentation:** Python uses indentation (whitespace) to define code blocks, such as loops and functions, instead of braces or other delimiters. Indentation is crucial for the correct interpretation of code.
- **Variables:** Variables are used to store data values. In Python, you can declare a variable by assigning a value. Variable names are case-sensitive and can contain letters, numbers, and underscores but should start with a letter or an underscore.
- **Comments:** Comments are used to explain code and are preceded by the `#` symbol. The Python interpreter ignores comments.
- **Print Statement:** The `print()` function is used to display output. You can print text and variable values using `print()`.
- **Data Types:** Python supports various data types, including integers, floats, strings, lists, tuples, dictionaries, and more.

**Python Data Types:**

1. **Integers (int):** Whole numbers, e.g., 1, -5, 100.
2. **Floats (float):** Numbers with decimal points, e.g., 3.14, -0.5, 2.0.
3. **Strings (str):** Ordered sequences of characters enclosed in single or double quotes, e.g., "Hello, World!".

### Example Program:

Here's a simple Python program that calculates the area of a rectangle using user-provided length and width:

```
# Input length and width from the user

length = float(input("Enter the length of the rectangle: "))

width = float(input("Enter the width of the rectangle: "))

# Calculate the area

area = length * width

# Display the result

print("The area of the rectangle is:", area)
```

In this program:

- We use the `input()` function to take user input for the length and width of the rectangle.
- The `float()` function converts the input values to floating-point numbers.
- We calculate the area by multiplying the length and width.
- Finally, we use the `print()` function to display the result.

You can run this program in a Python environment (such as IDLE or Jupyter Notebook) to interact with it and calculate the area of a rectangle based on your input values.

This example showcases some of the fundamental aspects of Python, including input/output, variable assignment, and basic arithmetic operations.

### Python Control Structures:

Control structures guide the flow of execution in a program. Python uses indentation to define blocks of code.

If-Else Statements: Used for conditional execution.



if x > 10:

```
    print("x is greater than 10")
```

elif x == 10:

```
    print("x is exactly 10")
```

else:

```
    print("x is less than 10")
```

For Loops: Ideal for iterating over sequences like lists or strings.

for i in range(5): # ranges from 0 to 4

```
    print(i)
```

While Loops: Executes as long as a condition is true.

i = 0

while i < 5:

```
    print(i)
```

```
    i += 1
```

Python Data Structures:

Python's built-in data structures are versatile, allowing for efficient data manipulation and storage, crucial for visualisation.

Lists

Lists are ordered collections that can hold various data types and are mutable.

```
numbers = [1, 2, 3, 4, 5]
```

Dictionaries

Dictionaries store key-value pairs, providing a way to associate information with keys.

```
person = {"name": "Alice", "age": 30}
```

## Tuples

Tuples are ordered collections like lists, but they are immutable.

```
coordinates = (10.0, 20.0)
```

## Sets

Sets are unordered collections of unique elements.

```
unique_numbers = {1, 2, 3, 4, 5}
```

## Python Operators:

### Arithmetic Operators:

Arithmetic operators are used with numeric values to perform everyday mathematical operations:

Addition (+): Adds two operands. E.g., 5 + 3 equals 8.

Subtraction (-): Subtracts the right operand from the left operand. E.g., 5 - 3 equals 2.

Multiplication (\*): Multiplies two operands. E.g., 5 \* 3 equals 15.

Division (/): Divides the left operand by the right operand. E.g., 5 / 2 equals 2.5.

Modulus (%): Returns the remainder of the division. E.g., 5 % 2 equals 1.

Exponentiation (\*\*): Raises the left operand to the power of the right. E.g., 5 \*\* 3 equals 125.

Floor Division (//): Divides and returns the integer part of the quotient. E.g., 5 // 2 equals 2.

### Assignment Operators:

Assignment operators are used to assign values to variables:

Assign (=): Assigns a value to a variable. E.g., x = 5.

Add and Assign (+=): Adds the right operand to the left operand and assigns the result. E.g., x += 3 is equivalent to x = x + 3.

And similar operations for -, \*, /, %, //, \*\*.

### Comparison Operators:

Comparison operators compare two values and return a Boolean value (True or False):

Equal (==): Checks if the value of two operands is equal. For example,  $5 == 3$  is False.

Not Equal (!=): Checks if the value of two operands is not equal. E.g.,  $5 != 3$  is True.

Greater Than (>): Checks if the left operand is greater than the right operand. E.g.,  $5 > 3$  is True.

Less Than (<): Checks if the left operand is less than the right operand. E.g.,  $5 < 3$  is False.

Greater Than or Equal to (>=): Checks if the left operand is greater than or equal to the right operand.

Less Than or Equal to (<=): Checks if the left operand is less than or equal to the right operand.

Logical Operators:

Logical operators are used to combine conditional statements:

And (and): Returns True if both statements are true.

Or (or): Returns True if one of the statements is true.

Not (not): Reverse the result returns False if the result is true.

Bitwise Operators:

Bitwise operators act on operands as if they were strings of binary digits. They operate bit by bit:

AND (&), OR (|), XOR (^), NOT (~), Left Shift (<<), Right Shift (>>).

Identity Operators:

Identity operators compare the memory locations of two objects:

Is (is): Returns True if both variables are the same object.

Is not (is not): Returns True if both variables are not the same object.

Membership Operators:

Membership operators test if a sequence is presented in an object:

In (`in`): Returns True if a sequence with the specified value is present in the object.

Not in (`not in`): Returns True if a sequence with the specified value is not present in the object.

## 5. DATA VISUALISATION LIBRARIES

### 1. *Matplotlib*:

Overview: Matplotlib is one of Python's most widely used data visualisation libraries. It provides a flexible and extensive toolkit for creating static, animated, and interactive plots and charts.

Key Features:

- Line plots, scatter plots, bar plots, histograms, and more.
- Customisation options for colours, labels, titles, and legends.
- Support for LaTeX for mathematical notations.
- Integration with NumPy for seamless data manipulation.

Advantages: Matplotlib is highly customisable, making it suitable for creating publication-quality plots. It is also the foundation for many other Python visualisation libraries.

Disadvantages: Creating complex and interactive visualisations can be verbose and require extensive code.

### 2. *Seaborn*:

Overview: Seaborn is built on top of Matplotlib and focuses on enhancing the aesthetics of plots and simplifying the creation of complex statistical visualisations.

Key Features:

- Stylish and informative statistical visualisations, such as violin plots, pair plots, and heatmaps.
- Built-in themes and colour palettes to improve plot aesthetics.
- Integration with Pandas DataFrames for easy data handling.

**Advantages:** Seaborn simplifies the creation of aesthetically pleasing statistical visualisations with concise code. It is beneficial for exploring relationships in datasets.

**Disadvantages:** It may have limitations when creating highly customised or non-standard plots.

### 3. *Plotly:*

- **Overview:** Plotly is a versatile library for creating interactive and web-based visualisations. It supports various chart types and is suitable for creating dashboards and web applications.
- **Key Features:**
  - Interactive plots with zoom, pan, hover, and clickable elements.
  - Support for 3D visualisations, geographic maps, and real-time updates.
  - Integration with Dash for building interactive web applications.
- **Advantages:** Plotly excels at creating engaging and dynamic visualisations that can be embedded in web applications. It provides APIs for multiple programming languages, not just Python.
- **Disadvantages:** Some advanced features may require a deeper understanding of Plotly's API and JavaScript.

### 4. *Bokeh:*

**Overview:** Bokeh is another library that creates interactive and web-based visualisations. It emphasises interactivity and is designed for building data-driven web applications.

#### **Key Features:**

- High-performance interactive plots, including linked and brushed selections.
- Integration with widgets for creating interactive dashboards.
- Support for streaming data and real-time updates.

**Advantages:** Bokeh is ideal for creating interactive dashboards and web applications with complex visualisations. It has a powerful and flexible API.

**Disadvantages:** The learning curve can be steep, especially for complex applications.



### 5. *Altair*:

Overview: Altair is a declarative statistical visualisation library that generates Vega-Lite JSON specifications. It simplifies creating interactive visualisations using a concise and intuitive syntax.

#### Key Features:

- Declarative grammar for specifying visualisations.
- Automatic generation of interactive plots from Pandas DataFrames.
- Easy customisation of plots using method chaining.

Advantages: Altair's simplicity and conciseness make it an excellent choice for quickly generating interactive visualisations with minimal code.

Disadvantages: It may not offer the same level of flexibility as lower-level libraries like Matplotlib for highly customised plots.

These are just a few of Python's data visualisation libraries. The library choice depends on your specific project requirements, familiarity with the library's API, and the type of visualisations you need to create. Combining libraries, such as using Seaborn for initial exploration and Matplotlib for fine-tuning, is also standard in data visualisation projects.

## 6. PYTHON IN THE DATA SCIENCE ECOSYSTEM

### Python's Comparative Advantages

Python has emerged as a leading programming language in the data science community due to its simplicity, flexibility, and a robust ecosystem of libraries and frameworks. Several factors contribute to Python's prominence in data science:

- **Comprehensive Libraries**  
Python boasts an extensive collection of libraries for data analysis (Pandas), machine learning (scikit-learn, TensorFlow, PyTorch), and data visualisation (Matplotlib, Seaborn, Plotly). This rich ecosystem enables data scientists to perform various tasks, from preprocessing data to deploying predictive models, all within the same programming environment.
- **Ease of Learning and Use**  
Python's syntax is designed to be intuitive and readable, making it accessible to beginners and convenient for experienced programmers. This ease of use accelerates the development process and facilitates the quick prototyping of data science projects.
- **Community Support**  
A vibrant and growing community surrounds Python, providing resources, tutorials, and forums for learning and troubleshooting. This extensive support network is invaluable for both novice and experienced data scientists.
- **Versatility and Integration**  
Python's flexibility allows it to be used in various stages of the data science workflow, including data gathering, analysis, modelling, and visualisation. Additionally, Python integrates well with other technologies and platforms, enabling seamless workflows and interoperability with databases, web applications, and cloud services.

### Comparing Python with Other Tools

While Python is preferred for many data science tasks, comparing its features and capabilities with other tools and languages helps understand its relative strengths and weaknesses.

- R vs. Python

R is another popular language for statistical analysis and visualisation. While R has a steep learning curve and is primarily used for statistical analysis, Python is more versatile, making it suitable for a broader range of data science applications beyond statistics.

Python's extensive library for machine learning and deep learning is more developed than R's, making Python the go-to choice for these fields.

- MATLAB vs. Python

MATLAB is a high-performance language for technical computing but comes with licensing costs. Python, being open-source, offers a free and flexible alternative with similar capabilities, especially with libraries like NumPy and SciPy for numerical computing.

- Excel vs. Python

Excel is widely used in business settings for data analysis and visualisation but is limited in handling large datasets and lacks advanced analytics capabilities. With its powerful data manipulation libraries (e.g., Pandas) and visualisation tools, Python can handle larger volumes of data and perform more complex analyses.

### Python's Role in the Data Science Ecosystem

Python is a unifying force in the data science ecosystem, offering tools and libraries spanning the entire data analysis lifecycle. Its ability to integrate with other technologies (e.g., SQL databases, big data platforms like Apache Spark, and web frameworks like Django) further enhances its utility and applicability in diverse data science projects.

## 7. INTERACTIVE PYTHON NOTEBOOKS

Interactive Python notebooks, particularly Jupyter Notebooks, have revolutionised how data scientists and researchers conduct exploratory data analysis, data visualisation, machine learning, and much more. These notebooks combine executable code, rich text, equations, and visualisations into a single document, facilitating a dynamic and interactive computational environment.

Jupyter Notebooks is an open-source web application that allows you to create and share documents that contain live code, equations, visualisations, and narrative text. The name "Jupyter" refers to the core supported programming languages Julia, Python, and R, with Python being the most popularly used in these notebooks.

### Key Features

- **Interactive Coding:** Code cells allow interactive editing and execution of Python code.
- **Rich Text:** Support for Markdown and HTML for narrative text, including explanations and comments.
- **Equations:** Integration with LaTeX for displaying mathematical equations.
- **Visualisation:** Inline support for displaying plots and graphics created with Python visualisation libraries like Matplotlib, Seaborn, and others.
- **Export Options:** Notebooks can be exported to various formats, including HTML, PDF, and slideshows, facilitating easy sharing and presentation.

### Setting Up Jupyter Notebooks

Jupyter Notebooks can be installed and run in various environments, from local machines to remote servers and cloud platforms.

### Installation

The easiest way to install Jupyter Notebooks is by installing Anaconda, a distribution of Python and R for scientific computing and data science. Anaconda includes the Jupyter Notebook as part of its default package.

Alternatively, Jupyter can be installed using pip, Python's package manager:

```
pip install notebook
```

## Running Jupyter Notebooks

To start a Jupyter Notebook server, open a terminal or command prompt, navigate to your project directory or a directory where you wish to save your notebooks and run:

```
jupyter notebook
```

## Using Jupyter Notebooks for Data Visualisation

Interactive Python notebooks are particularly well-suited for data visualisation due to their interactive nature and the ability to mix code with narrative and images.

## Creating Visualisations

- Import data visualisation libraries (e.g., Matplotlib, Seaborn) at the beginning of your notebook.
- Use code cells to write Python code for data manipulation and visualisation.
- Visualisations appear inline, directly below the code cells that produce them.

## Sharing Insights

- Use Markdown cells to add explanations, hypotheses, conclusions, or any narrative alongside your code and visualisations.
- Share your notebooks with peers or publicly to provide a comprehensive view of your analysis and findings.

## Best Practices

- **Code Organisation:** Organise your code into logical sections, each dedicated to a specific part of your analysis or visualisation.
- **Comments and Documentation:** Use comments within code cells and Markdown for detailed explanations to ensure your notebooks are understandable to others.
- **Interactive Widgets:** Use Jupyter's support for interactive widgets to create dynamic, user-interactive visualisations.



## 8. BEST PRACTICES IN CODE VISUALISATION

Effective data visualisation is not just about creating visually appealing charts and graphs but also about writing clean, readable, and maintainable code.

### Writing Clean and Understandable Code:

The clarity of your code directly affects its maintainability and the ease with which you and others can modify or reuse it. Here are some guidelines to ensure your code is clean and understandable:

#### Use Descriptive Variable Names:

Good Practice: Use meaningful variable names that reflect the data or the purpose of the variable. For example, `daily_sales_sum` is more informative than `dss` or `temp`.

#### Follow a Consistent Coding Style

Consistency is Key: Adhere to a consistent coding style regarding naming conventions, spacing, and indentation. The Python community recommends following the PEP 8 style guide, which provides guidelines for writing readable Python code.

#### Modularize Code

Functions and Modules: Break down complex visualisation code into smaller, reusable functions or modules. This makes your code more readable and maintainable and facilitates code reuse across different projects.

#### The Importance of Commenting and Documentation

Well-documented code is crucial for understanding the what, why, and how of your data visualisation logic. Comments and documentation significantly make your code accessible to others, including your future self.

#### Inline Comments and Docstrings

- **Inline Comments:** Use inline comments sparingly to explain non-obvious parts of the code or to provide context for complex operations.
- **Docstrings:** Use docstrings (triple-quoted strings) at the beginning of your modules, functions, classes, and methods to describe their purpose and behaviour. Docstrings can also briefly overview the arguments, return values, and potential side effects.

## External Documentation

**README Files and Wikis:** For larger projects, supplement your code with external documentation, such as README files, wikis, or dedicated documentation websites. These can provide an overview of the project, installation instructions, examples of usage, and explanations of the architecture and design decisions.

## Best Practices for Visualization-Specific Code

Regarding data visualisation, some additional best practices specifically tailor your code to enhance the clarity and effectiveness of the visualisations it produces.

### Use Visualisation Libraries Wisely

**Library Features:** Leverage the features of visualisation libraries like Matplotlib, Seaborn, or Plotly to their fullest. Utilise built-in functions and parameters to reduce the custom code needed for everyday tasks.

### Consistency in Visual Design

**Visual Style:** Maintain a consistent visual style across your visualisations. Consider creating a style guide or using predefined styles offered by your visualisation library to ensure consistency.

### Interactive and Dynamic Visualisations

**Interactivity:** For interactive visualisations, clearly document the interactive elements and their intended use. Comment on event-driven functions or callbacks to make the code easier to follow.

## 9. SUMMARY

In this chapter, we embarked on a journey into Python, a versatile and beginner-friendly programming language known for its readability and extensive library ecosystem. We began by understanding Python's clean and intuitive syntax, exploring fundamental concepts such as indentation, variable declaration, comments, and the `print()` function. We delved into the realm of data types, covering integers, floats, strings, and the power of variables in storing and manipulating data. The chapter also introduced us to essential Python libraries for data science and visualisation, including Matplotlib, Seaborn, Plotly, Bokeh, and Altair. These libraries allow us to create visualisations, from static plots to interactive web-based dashboards. Finally, we learned how to set up our Python working environment on different platforms, ensuring we can harness Python's capabilities in data analysis and visualisation tasks. Learn about the various interactive notebooks and best practices of code visualisation.

## 10. QUESTIONS

### Self-Assessment Questions

#### SELF-ASSESSMENT QUESTIONS – 1

1. What is Python known for in terms of its syntax and readability?
2. Why is proper indentation significant in Python code?
3. How do you declare a variable in Python?
4. Explain the role of comments in Python code.
5. Which function displays output in Python, and how can it be customised?
6. Name three common data types in Python.
7. Describe the purpose of Matplotlib in Python.
8. What is the critical feature of Seaborn that distinguishes it from Matplotlib?
9. How does Plotly contribute to the creation of interactive visualisations?
10. Briefly explain the focus of Bokeh as a data visualisation library.

**Terminal Questions**

1. Explain the importance of proper indentation in Python code. Provide examples illustrating how indentation defines code blocks and influences program execution.
2. Describe the critical characteristics of Python variables, including naming conventions, data types, and their role in storing data. Provide examples to illustrate variable declarations and assignments.
3. Compare and contrast Python's `print()` function and comments (`#`). Explain how each is used in Python code and provide examples.
4. Explore the data types in Python, including integers, floats, and strings. Please provide examples of each data type and explain their practical applications in programming.
5. Discuss the role and significance of Matplotlib in Python's data visualisation landscape. Provide examples of different types of plots that can be created using Matplotlib.
6. Compare and contrast Seaborn and Plotly as Python data visualisation libraries. Highlight their key features, use cases, and advantages. Provide examples to showcase the types of visualisations that can be created with each library.
7. Write a short note on Interactive Python Notebooks.
8. List the steps for efficient best practices for code visualisation.

## 11. ANSWERS

### Self-Assessment Questions

1. Python is known for its simplicity and readability in terms of syntax.
2. Proper indentation is essential in Python code because it defines code blocks, such as loops and functions, and ensures correct code structure.
3. You declare a variable in Python by assigning a value to it. For example: `my_variable = 42`
4. Comments in Python are used to explain code and are preceded by the `#` symbol. The Python interpreter ignores them.
5. The `print ()` function is used to display output in Python. It can be customised by providing different arguments and formatting options.
6. Three common data types in Python are integers (`int`), floats (`float`), and strings (`str`).
7. Matplotlib is used in Python to create a wide range of plots and charts, including static, animated, and interactive ones.
8. Seaborn is known for enhancing the aesthetics of statistical visualisations and provides built-in themes and colour palettes.
9. Plotly allows the creation of interactive and web-based visualisations with features like zoom, pan, and hover interactions.
10. Bokeh focuses on interactivity and is designed to create data-driven web applications with high-performance interactive plots.

### Terminal questions

1. Refer Section 2
2. Refer Section 3
3. Refer Section 3
4. Refer Section 4
5. Refer Section 5
6. Refer Section 5
7. Refer Section 7
8. Refer Section 8