CHAPTER 1

Introduction to GNU Octave

We hope you have gone through the table of contents and the preface. If not, we highly recommend you do so. This is the very first chapter of this book and we welcome you to the exciting journey of learning GNU Octave.

In this chapter, you will learn the details of GNU Octave such as its history, applications, limitations, and a comparison with other contemporary and similar tools. This chapter is mostly dedicated to general information about GNU Octave and its installation on various popular OS platforms such as Windows, Ubuntu, and Raspberry Pi Raspbian. You will not be writing any programs or learning about the functionality of GNU Octave here. The following is the list of topics you will learn about in this chapter:

- The GNU Octave Project
- Applications
- Limitations
- The community
- Comparison with other tools
- Installation of GNU Octave on various platforms
- Working with GNU Octave in different modes

A. Pajankar and S. Chandu, *GNU Octave by Example*,
https://doi.org/10.1007/978-1-4842-6086-9_1

The GNU Octave Project

GNU Octave is a high-level programming language. It is used for numerical and scientific computing. It is part of the GNU Project so it is free and open-source. In fact, anyone with the necessary skill set and will to so can contribute to its development. The homepage of GNU Octave is located at www.gnu.org/software/octave. It is basically a mathematics-oriented programming language with convenient and easy-to-learn visualization tools for scientific researchers.

The Octave interpreter is written in the C++ programming language. Octave is an interpreted programming language because it uses the Octave interpreter to run the Octave scripting language statements and scripts. Octave has a lot of dynamically loadable modules. Octave uses OpenGL or gnuplot for plotting. Octave has both a GUI (graphical user interface) and a CLI (command line interface). If any of you have experience with working with an OS shell interpreter or the Python programming language, consider Octave as almost the same as working with shell or Python programming.

History of the GNU Octave Project

The GNU Octave Project started in 1988 as a companion for a textbook that was under development for chemical engineering undergraduate students. This was done after the faculty members observed that chemical engineering students were spending a lot of time debugging FORTRAN issues, which was used for their programming exercises. Full-time development began in 1992. Gradually it became a part of the GNU Project. The following is a timeline that shows the major milestones in the development of GNU Octave:

- 1988: Conception of idea
- 1992: Beginning of full-time development
- 1994: Version 1.x.x

- 1996: Version 2.0.x and Windows port
- 2007: Version 4.0
- 2015: Version 4.0.0 with stable GUI
- 2019: Octave 5.1.0

Applications of Octave

Octave is used to solve different scientific and numerical computational problems. It can be used for linear programming and optimization. Octave is also deployed on many supercomputers because it supports parallel programming. You can find GNU Octave deployed at supercomputers in the Ohio Supercomputer Center (www.osc.edu/resources/available_software/software_list/octave), the Oak Ridge National Laboratory (www.olcf.ornl.gov/software_package/octave), the and University of Minnesota (www.msi.umn.edu/sw/octave). In the research community, Octave is actively used for data analytics, image processing, computer vision, economic research, data mining, statistical analysis, machine learning, signal processing, and many more scientific applications. You will learn how to demonstrate programs pertaining to a lot of the above-mentioned scientific computing areas with GNU Octave.

Limitations and Drawbacks of Octave

The Octave programming language was primarily developed to perform numerical and scientific computations. It is not supposed to be used as a general purpose programming language like C and C++. Also, it is our opinion that you should always choose a programming language suitable for your own programming or computational needs. If you are looking to do some system-level programming, then C and assembly languages are your friend. However, if you are a subject matter expert (for example, a

chemical engineer or a signal processing professional) who cannot spare enough time to learn the intricacies of a programming language like C, then you should use GNU Octave or the Python programming language because you can quickly write code snippets to prototype your ideas.

We mentioned that GNU Octave is an interpreted programming language. This means that it first converts the code or statements into a machine-readable code format before the computer executes them. The main drawback is that the program executes slowly compared to programs written in compiled languages such as C or Fortran. And it is certainly slower than assembly. The main advantage of this approach is that the statements are easy to write and change, and the programmer does not have to compile the code before executing it. It gives a very high degree of control to the programmers. This is why Octave is not the first choice when it comes to system programming or fast or parallel programs on a supercomputer. The C programming language is more suitable for such applications. However, as you will experience later in the book, Octave lets you solve very advanced and computationally demanding problems with only a few instructions or commands and with satisfactory speed.

Comparison of Octave with Alternatives

Octave is a part of GNU, thus it is a free and open-source package and programming environment for numerical and scientific computations. Many times it is promoted as a free alternative for MATLAB. MATLAB is a short form of Matrix Laboratory. It is also a programming environment and language for numerical and scientific computing. MATLAB is developed and maintained by Mathworks. It is a proprietary and commercial software. Octave tries to maintain a very high degree of syntax compatibility with MATLAB. Many of the programs we will demonstrate can be directly run as they are with MATLAB. Keep in mind that this applies to many, but not all, of the programs.

The other free alternatives of MATLAB are Scilab and FreeMat. The Scilab project does not attempt much to maintain syntax-level compatibility with MATLAB and Octave. The FreeMat project has not been updated since 2013.

The Online Octave Community

You can find all of the information and downloadable setup files for Octave at the project website at www.octave.org. Here you'll find the official manual, a Wiki page with tricks and tips (https://wiki.octave.org/GNU_Octave_Wiki), latest news, a more detailed history, and other relevant information. You can also get involved in the development; visit www.gnu.org/software/octave/get-involved.html for more information. StackOverflow is a good source of information and help. You can find questions related to Octave at https://stackoverflow.com/questions/tagged/octave.

There many additional packages that do not come preinstalled with the standard Octave distribution. Many of them can be downloaded from Octave Forge at https://octave.sourceforge.io. Octave Forge is a community project for collaborative development of GNU Octave extensions, called Octave packages. Here you can find specially designed packages for scientific and numerical applications such as image processing, signal processing, economics, information theory, analytical mathematics, and so on.

Installing GNU Octave

In this section, you will learn how to install GNU Octave on multiple platforms such as Windows, Ubuntu, and Raspberry Pi. All of the code examples and interactive sessions we demonstrate in this book have been tested on these platforms by the authors. So, let's begin.

Installing on Windows

You can install GNU Octave on Windows by downloading and executing the installable file from the Octave download page at www.gnu.org/software/octave/download.html. This page has options for 32-bit and 64-bit computers. There is an option for **linear algebra for large data** but you will not need it for this book. So, choose the .exe file for installing to 32-bit or 64-bit Windows computers. Other formats, 7z and .zip, are also available. But you should go for the .exe file. Download the file and execute it to install GNU Octave. Once the setup has completed successfully, add the directory location of the Octave executable to the Windows PATH variable. In my case, it is C:\Octave\Octave-5.2.0\mingw64\bin. It could be different for you based on the GNU Octave version and your computer architecture (32-bit or 64-bit).

Once you are done installing it, you need to install Python 3 because you will need the pip3 utility of Python 3 to install Jupyter Notebook and Octave Kernel for it. Also, in the end, you will learn how to connect Python 3 with GNU. You will use the Python 3 interpreter at that time. Visit the Python 3 download page located at www.python.org/downloads/ and download the setup file of Python 3 for your computer, as shown in Figure 1-1.

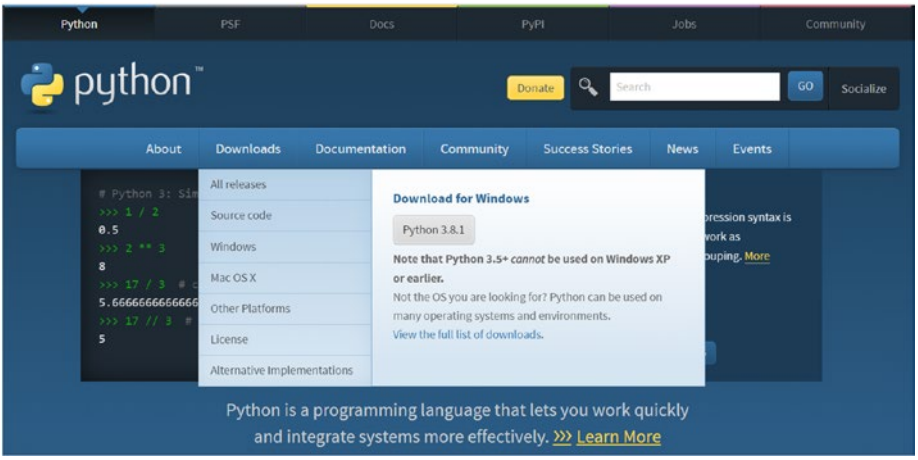

Figure 1-1. *Python project homepage*

Run the setup file to install Python 3. During installation, check the checkbox related to adding Python 3 to the PATH variable, as shown in Figure 1-2.

Figure 1-2. *Python Installation Wizard*

Also, choose the **Customize installation** option. This will show you more options, as shown in Figure 1-3.

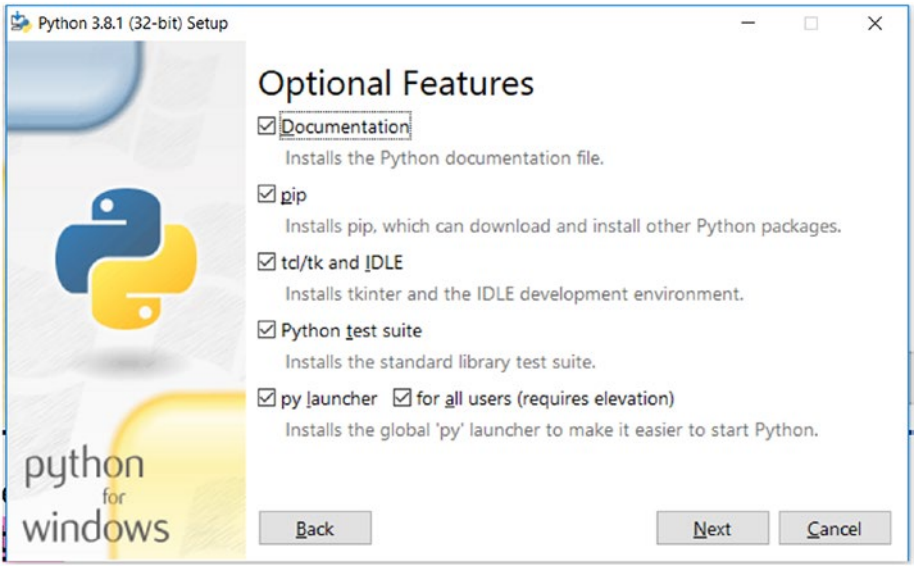

Figure 1-3. *Python installation options*

Check all the boxes and click the **Next** button to continue the setup. Complete the setup. Once done, run the following command at the Windows command prompt **cmd**:

```
python -V
```

It will return the version of Python 3 as follows:

```
Python 3.8.1
```

You can also check the version of **pip3** as follows:

```
pip3 -V
```

pip stands for **Pip installs Python** or **Pip installs Packages**. Its name is a recursive acronym. It is a package manager for the Python programming language. You can install the other needed components for our demonstrations with pip. To install **Jupyter**, run the following command at the command prompt:

```
pip3 install jupyter
```

Jupyter is an interactive environment for various programming language. You will see the details of Jupyter at the end of this chapter.

To install the Octave Kernel for Jupyter, run the following command:

```
pip3 install octave_kernel
```

The Octave Kernel for Jupyter allows us to run the Octave programs in a Jupyter notebook. As mentioned, you will see how to work with GNU Octave and Jupyter in the end of this chapter.

Installation on Ubuntu Linux

Ubuntu Linux is a distribution based on Debian Linux. Both are popular Linux distributions. Python 3 and pip3 come preinstalled in Ubuntu so you do not have to install them separately. First, update the package list for upgrades by running the following command in the **terminal** program:

```
sudo apt-get update
```

Then install GNU Octave with the following command:

```
sudo apt install octave -y
```

Then using **pip3**, install Jupyter and the Octave Kernel as follows:

```
pip3 install jupyter
pip3 install octave_kernel
```

Run the above commands and complete the setup.

Installation on Raspberry Pi with Raspbian OS

Raspberry Pi is a popular single board computer. If a desktop computer or a laptop is out of your budget, you can opt for a Raspberry Pi. The recommended operating system for Raspberry Pi is Raspbian OS, which is a Debian derivative for the ARM processor architecture that Raspberry Pi boards use. The setup of Raspberry Pi is outside of the scope of this book, but you can find detailed instructions at www.raspberrypi.org. Once you get your Raspberry Pi ready, you can run the following commands on the **lxterminal**, which is the terminal emulator for Raspbian OS, so to install Octave, Jupyter Notebook, and Jupyter Kernel, type these commands:

```
sudo apt-get update
sudo apt-get install octave -y
sudo pip3 uninstall ipykernel
sudo pip3 install ipykernel==4.8.0
sudo pip3 install jupyter
sudo pip3 install prompt-toolkit==2.0.5
sudo pip3 install octave_kernel
```

Running the above commands in sequence will install all of the required packages for this demonstration on the Raspbian OS of Raspberry Pi.

Exploring GNU Octave

Let's start exploring various aspects of GNU Octave. We will start with GUI.

Octave GUI

When you install Octave on Windows, you also get a shortcut to the Octave GUI on your desktop. There is another way to launch it. You can search for it in the search box of Windows by typing **Octave**. Two options will appear:

Octave GUI and **Octave CLI**. Choose the GUI option. On Ubuntu, you can launch it by searching for it in the search box and clicking the Octave icon displayed in the search output, as shown in Figure 1-4.

Figure 1-4. *GNU Octave on Ubuntu*

In the Raspberry Pi Raspbian OS menu (the raspberry fruit icon located at the top left corner on the Raspbian OS desktop), you can find it under Education, as shown in Figure 1-5.

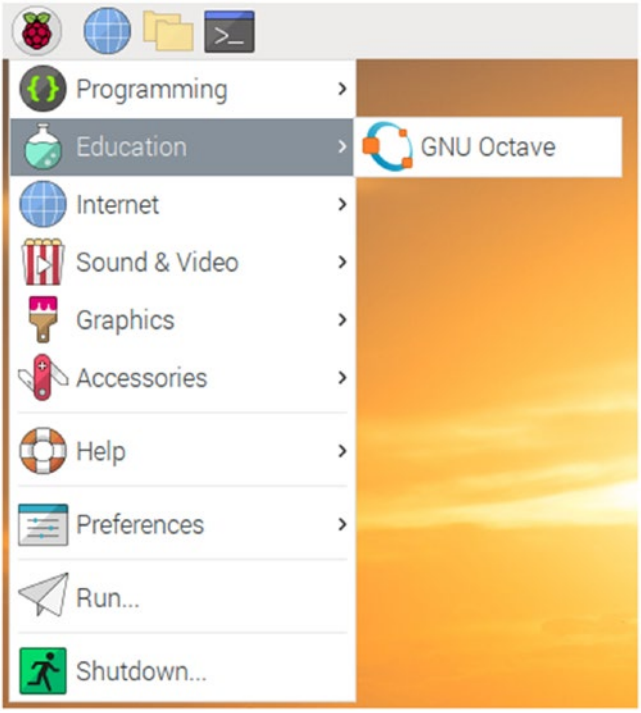

Figure 1-5. *GNU Octave on Raspbian*

When you launch GNU Octave the very first time on the Raspberry Pi with Raspbian OS, it shows the welcome message window, as shown in Figure 1-6.

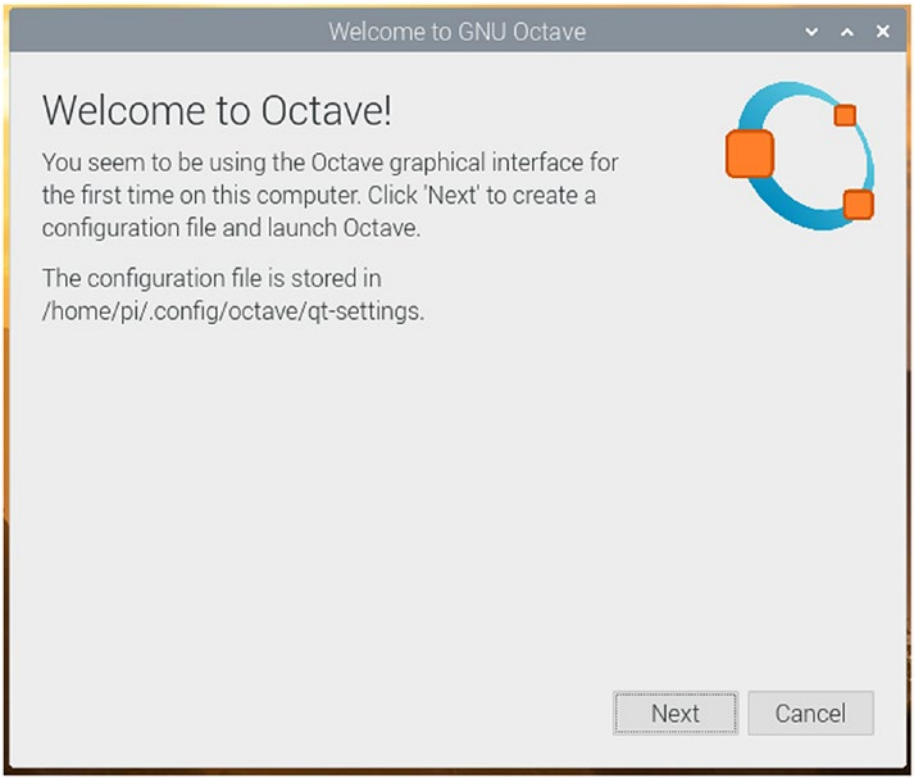

Figure 1-6. *GNU Octave welcome screen*

Click the **Next** button and you will see the window shown in Figure 1-7.

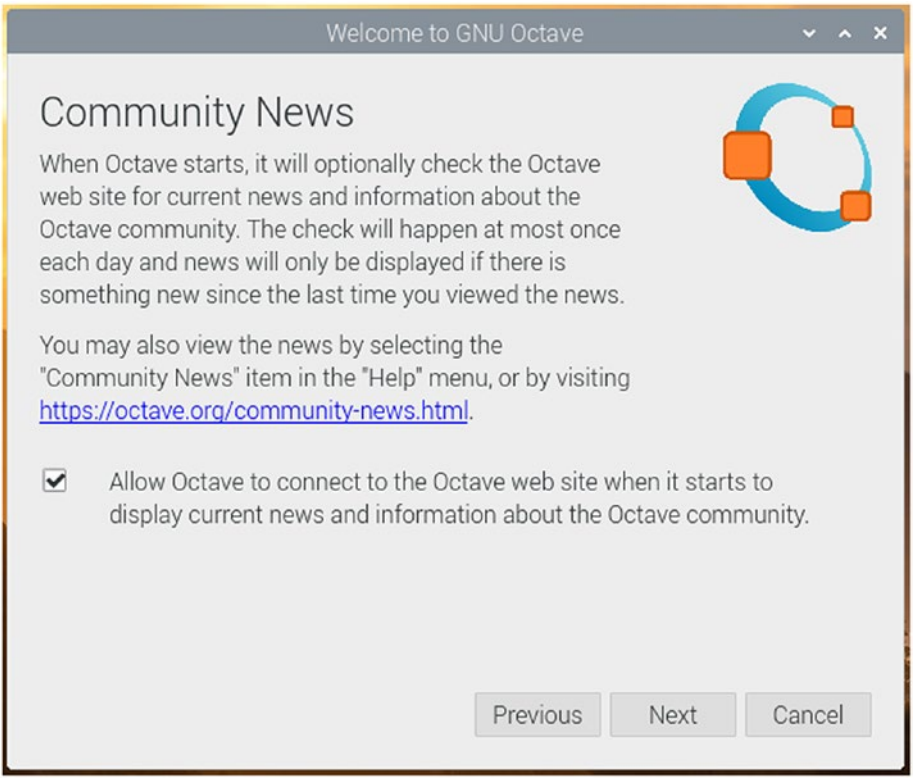

Figure 1-7. *Community news*

It is recommended to check the checkbox (to receive latest news and information about the Octave community). Click the **Next** button and you'll see the window shown in Figure 1-8.

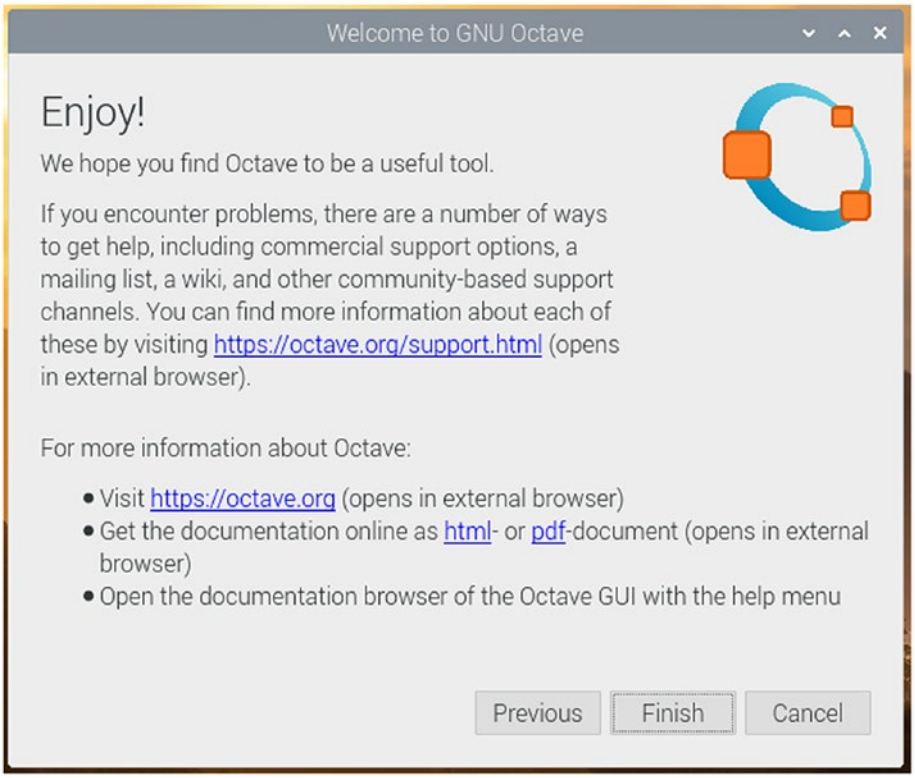

Figure 1-8. *Help information*

Click the **Finish** button and the Octave GUI will be launched.

The GUI Window looks the same on all platforms. Figure 1-9 shows the Octave GUI window running on a Windows computer.

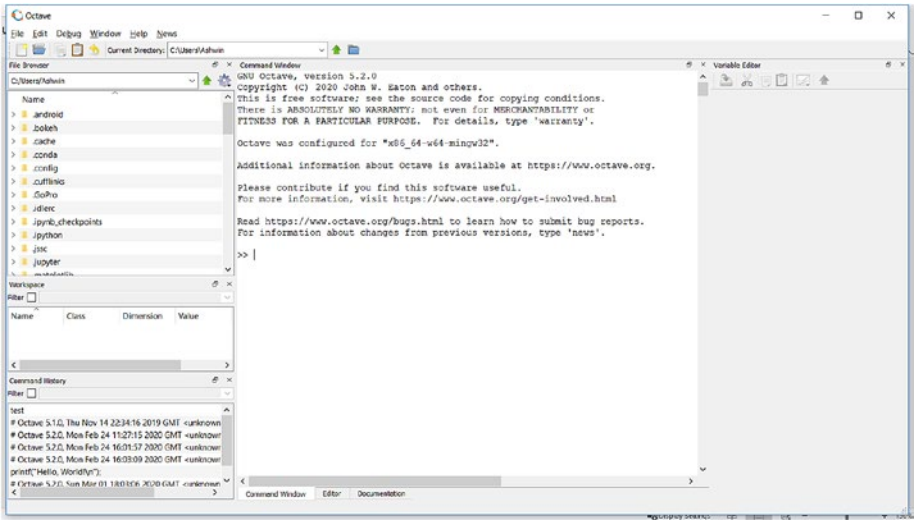

Figure 1-9. *GNU Octave GUI on Windows*

Let's look at the details of the components in this window one by one. In Figure 1-9, you can clearly see that the GUI is divided into three vertical sections. The middle section is the Octave interpreter prompt. You can interact with it like the command prompt of an OS. It runs the Octave statements, which you will see soon. The vertical section on the right is the **variable editor**. The vertical section on the left is divided into three sub-sections: a **file browser**, a **workspace**, and a **command history** window. You can rearrange these spaces anytime you want by dragging and dropping them within the GUI window.

The top offers a menubar with the usual file operations and their shortcuts. And if you pay close attention, in the bottom of the window, you'll see three tabs that read **Command Window**, **Editor**, and **Documentation**. The command window is the interactive mode command prompt that you can see in the screenshot. The Editor tab opens a code editor window, and Documentation will bring up an index of the browsable documentation. You will explore all of these things one by one.

But first, let's get started with the customary **Hello World!** program. Go to the interactive window and type in `printf("Hello, World!\n");` and then press Enter. It prints the string enclosed in the double quotes in the interactive window, as shown in Figure 1-10.

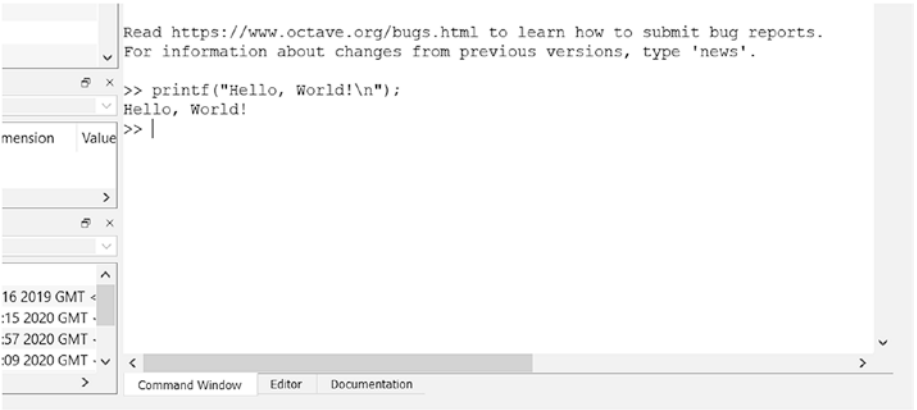

Figure 1-10. *The command window of the GNU Octave GUI*

You can even create a single-line program of this code and save it. Go to the editor by choosing the Editor tab at the bottom. Type the same line as above in the editor and save it. Octave automatically assigns the `.m` extension to the file. MATLAB uses the same extension. The simple program is shown in Figure 1-11.

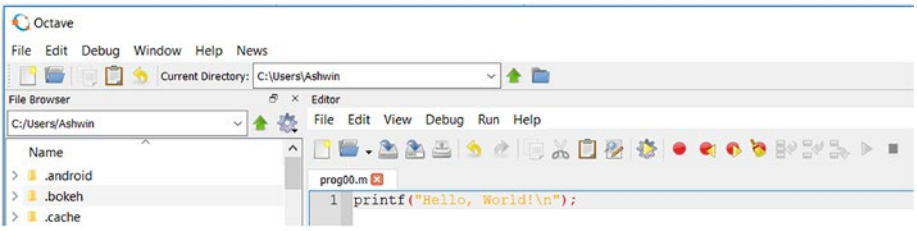

Figure 1-11. *A simple program*

Here, we saved the program on the disk in the computer. Under the Editor section, you can see all the options any IDE (integrated development environment) has. You can change the font in the editor by pressing the ***Ctrl*** key on the keyboard and moving the scroll wheel of the mouse at the same time. After saving, click the Run symbol (the gear and yellow triangle; in Linux, it is a paper plane). After that, it shows the dialog box, as shown in Figure 1-12.

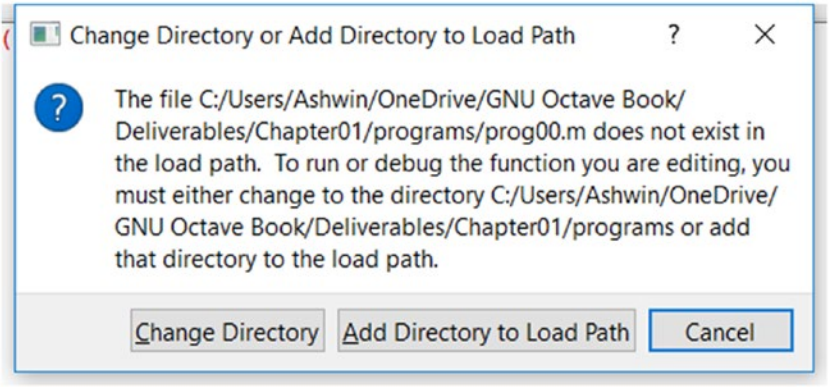

Figure 1-12. *Dialog box to load path*

Click the **Add Directory to the Load Path** button. The program is executed by the Octave interpreter and you can see the output in the interactive tab, as shown in Figure 1-13.

```
>> printf("Hello, World!\n");
Hello, World!
>>

>> prog00

Hello, World!
```

Figure 1-13. *Output of the simple program*

As you can see, it prints the program name without the extension and then shows the output.

Congratulations! You have just run your first GNU Octave program.

Octave CLI

You can also launch the command line independently. In Windows, you can either search for Octave and choose **Octave CLI** from the results or you can run the command `octave` in the command prompt to launch the CLI. In the Linux flavors like Ubuntu and Raspbian OS, you can run the same command, `octave`, in the command prompt to launch the Octave CLI. In order to exit the CLI, you must run the `exit` command. You can also run the `.m` octave files from the command prompt using the Octave interpreter. Suppose, on Windows, that the absolute path of your Octave program file is `C:\Book\Chapter01\programs\prog00.m`. You can execute the program using the Octave interpreter by running the following command at the command prompt:

```
octave "C:\Book\Chapter01\programs\prog00.m"
```

Similarly, on Raspberry Pi, suppose the absolute path of the Octave program file is `/home/pi/prog00.m`. You can run it from the command prompt with the following command:

```
octave "/home/pi/prog00.m"
```

Octave Programming with Jupyter Notebook

Jupyter Notebook is web-based notebook that is used for interactive programming of various programming languages like Python, Octave, Julia, and R. It is very popular with people who work in research domains. A Jupyter notebook can have code, visualizations, output, and rich text.

The advantage of a Jupyter notebook over Octave's own interactive prompt is that you can edit the code and see the new output instantly, which is not possible in the Octave command prompt. Another advantage is that you have the code and output in the same document. You can even share it on the cloud. There are many services online that help you store and execute your Jupyter notebook scripts on cloud servers.

Let's see how to use Jupyter Notebook for writing and executing Octave code. Open the command prompt of your OS (cmd in Windows, terminal in Ubuntu, and lxterminal in Raspbian OS). Run the following command there:

```
jupyter notebook
```

The Jupyter Notebook server process will be launched and the command prompt window will show a server log, as shown in Figure 1-14.

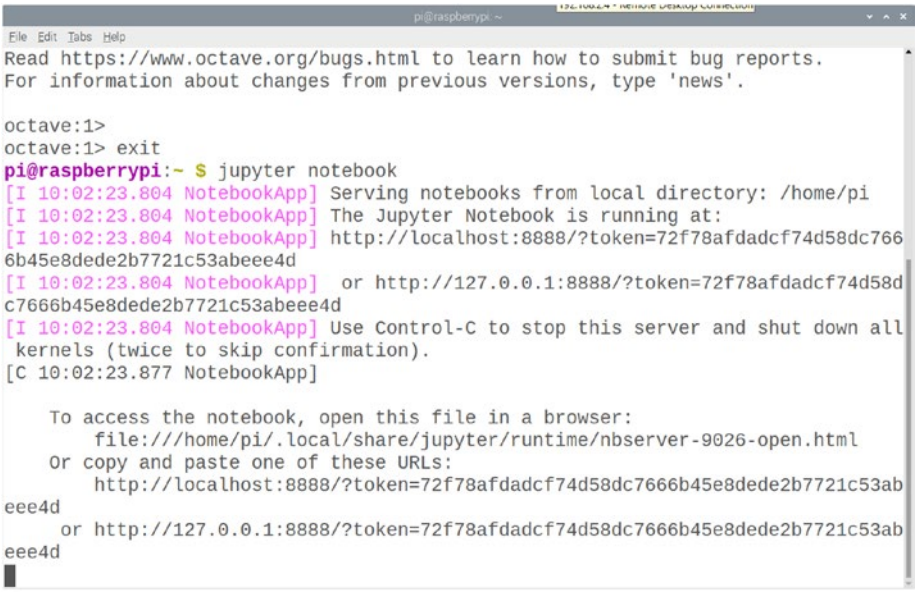

Figure 1-14. *Launching a new Jupyter Notebook process*

Also, it launches a webpage in the default browser in the OS. If the browser window is already open, it launches the page in a new tab of the same browser window. Another way to open the page (in case you accidentally close this browser window) is to visit `http://localhost:8888/` in your browser. Figure 1-15 shows the page you'll see.

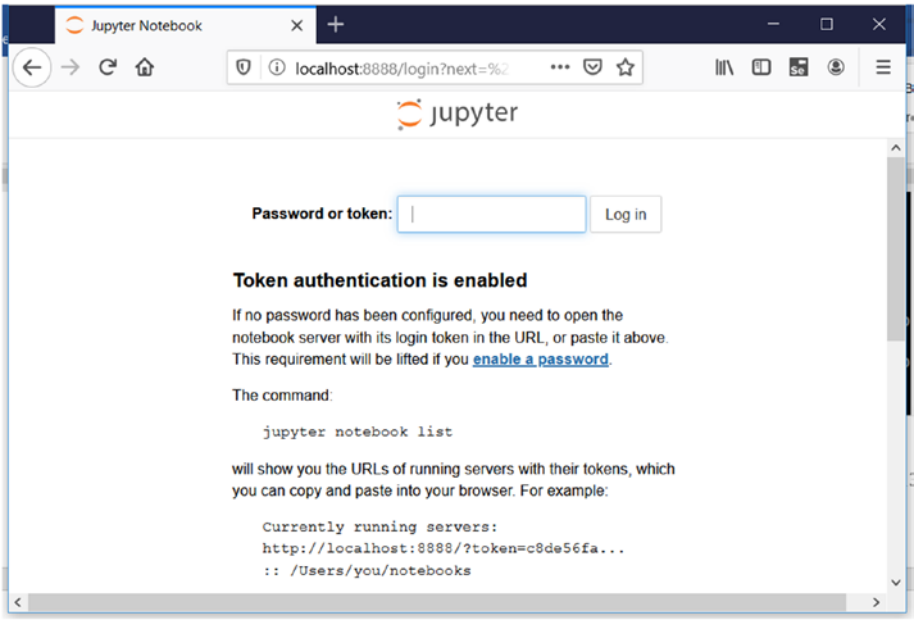

Figure 1-15. *Logging in with a token*

The token can be found in the server logs. The following is a sample server log with tokens. To access the notebook, open this file in a browser:

```
        file:///C:/Users/Ashwin/AppData/Roaming/jupyter/
        runtime/nbserver-8420-open.html
    Alternatively, you can copy and paste one of these URLs:
        http://localhost:8888/?token=e4a4fab0d8c22cd01b6530d5da
        ced19d32d7e0c3a56f925c
     or http://127.0.0.1:8888/?token=e4a4fab0d8c22cd01b6530d5da
     ced19d32d7e0c3a56f925c
```

In the log above, you can see a couple of URLs. They refer to the same page (localhost and 127.0.0.1 are the same hosts). You can either directly copy and paste any of these URLs into the address bar of the browser tab and open the Jupyter Notebook homepage or you can visit `http://localhost:8888/` as discussed and then paste the token in the server log (in this case, it is `e4a4fab0d8c22cd01b6530d5daced19d32d7e0c3a56f925c`) and log in. It will take you to the same homepage.

Note that every instance of the Jupyter Notebook server will have its own token, and the token here will not work with your Jupyter Notebook. The token is only valid for that server process.

So, if you follow any one of the routes explained above, you will see a homepage tab in the browser window, as shown in Figure 1-16.

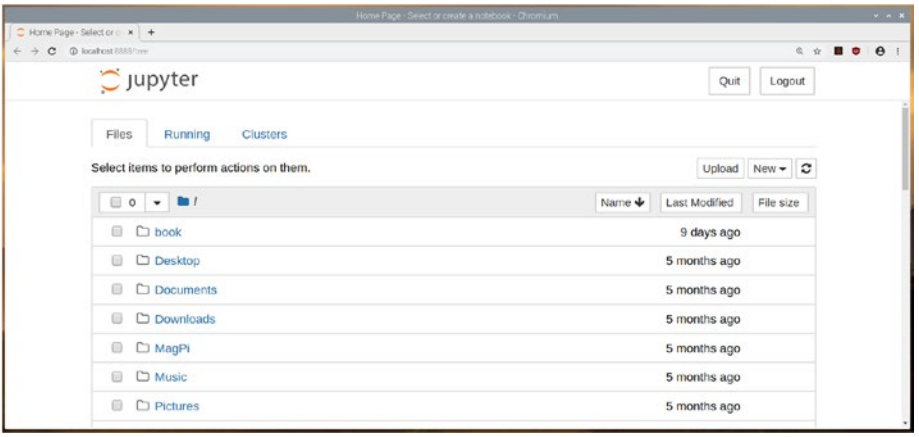

Figure 1-16. *A new homepage tab of Jupyter Notebook*

As you can see, there are three tabs in the webpage itself: **Files**, **Running**, and **Clusters**. The Files tab shows the directories and files in the directory from where you launched the notebook server from the command prompt. In the above example, we executed the command `jupyter notebook` from **lxterminal** of our Raspberry Pi. And the present

working directory is the ***home*** directory of the ***pi*** user `/home/pi`. This is why we can see all the files and directories in the home directory of our RPi computer in Figure 1-16.

In the top right corner are the **Quit** and **Logout** buttons. If you click the Logout button, it logs out from the current session; in order to log in, you need the token or URL with the embedded token from the notebook server log, as discussed. If you click the Quit button, it stops the notebook server process running in the command prompt and shows the modal message box, as shown in Figure 1-17.

Figure 1-17. *The message shown after clicking the Quit button*

In order to work with it again, you need to execute the command `jupyter notebook` again in the command prompt.

On the top right side, just below the Quit and Logout buttons is a small button with the refresh symbol. It is the refresh button. It refreshes the homepage. You can also see the **New** button. Once clicked, it shows a dropdown menu, as shown in Figure 1-18.

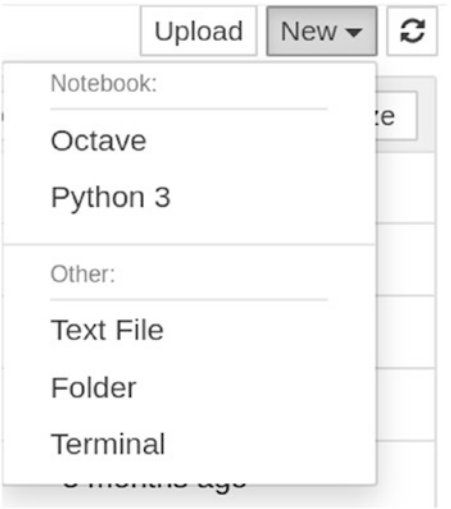

Figure 1-18. *Options for a new notebook*

As you can see, the dropdown is divided into two sections: **Notebook** and **Other**. You can create Octave and Python 3 notebooks. If your computer has other languages installed that are supported by Jupyter notebook, those languages will show up here. You can also create text files and folders. You can open the command prompt in the web browser by clicking **Terminal**. The output of **lxterminal** running in a separate web browser tab is shown in Figure 1-19.

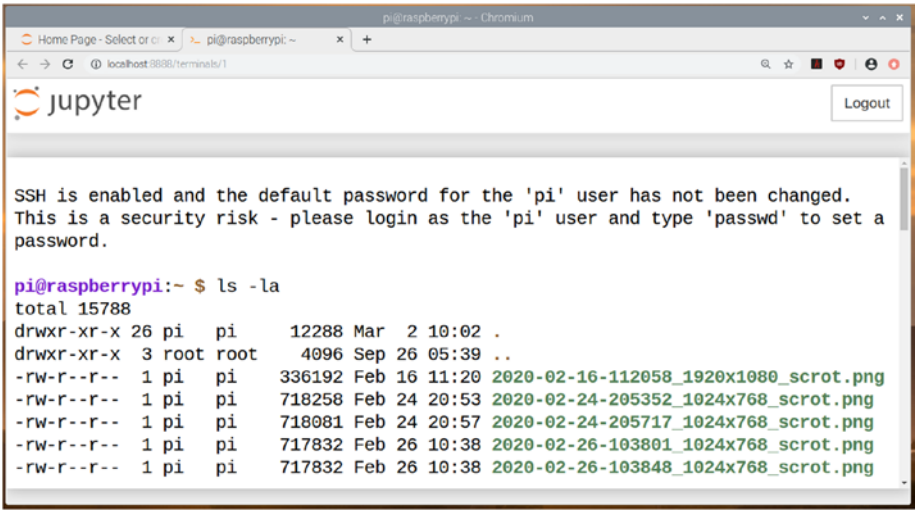

Figure 1-19. *A new lxterminal window within the browser*

Clicking Octave in the dropdown creates a new Octave notebook, as shown in Figure 1-20.

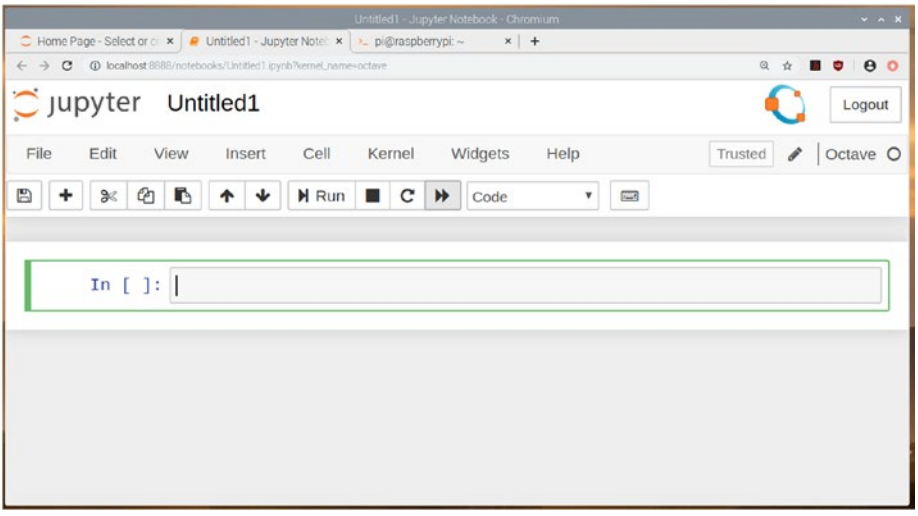

Figure 1-20. *A new GNU Octave notebook*

If you go to the homepage again by clicking the homepage tab in the browser and then opening the **Running** tab in the homepage, you can see the entries corresponding to the terminal and the Octave notebook, as shown in Figure 1-21.

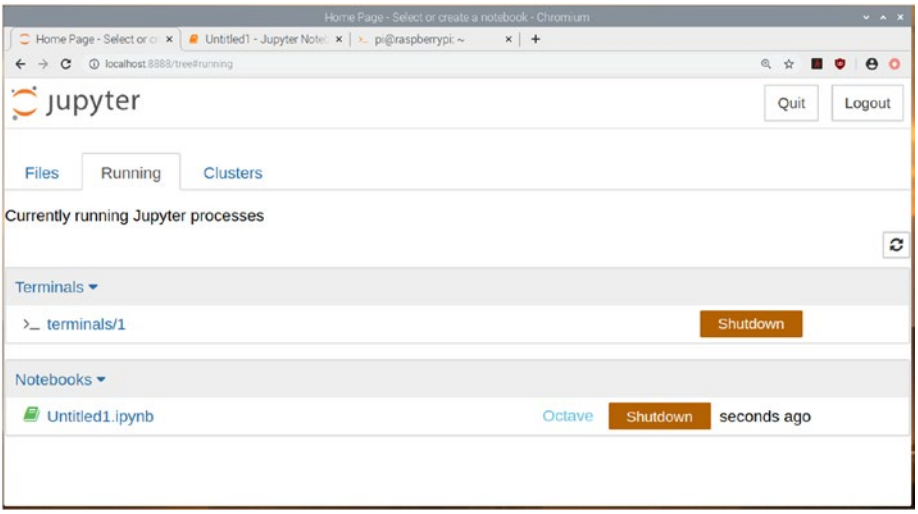

Figure 1-21. *Summary of current Jupyter Notebook subprocesses*

Octave Code and Richtext in Notebook

Go to the Octave **Untitled1** tab again and type in the following statement in the text area (also known as a cell):

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

Click the Execute button. Jupyter will execute the code as an Octave statement and show the result immediately below the cell, as shown in Figure 1-22.

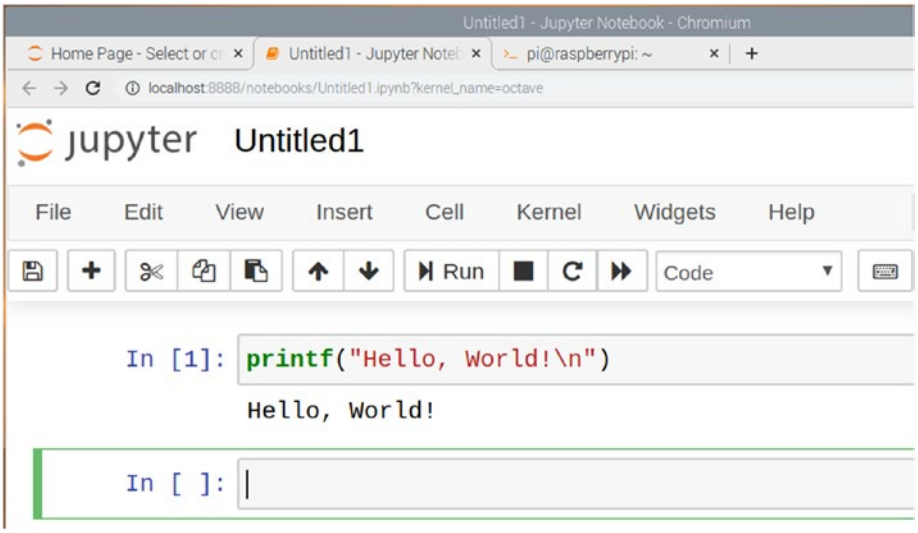

Figure 1-22. *Code output in Jupyter Notebook*

As you can see, after execution, it automatically creates a new cell below the result and sets the cursor there. Let's discuss the menu bar and the icons above the programming cells. You can save the file by clicking the floppy disc icon. You can add a new empty cell after the current cell by clicking the + icon. The next three icons are for cutting, copying, and pasting. The up and down arrows can shift the position of the current cell up and down, respectively. The next option is to run the cell, which you already saw in action. The next three icons are to interrupt the kernel, restart the kernel, and restart the kernel and rerun all the cells in the notebook. Next is a dropdown menu that tells you what type of cell it should be. Figure 1-23 shows the options when the dropdown menu is clicked.

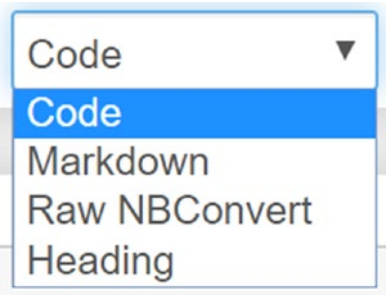

Figure 1-23. *Types of cells in a Jupyter notebook*

The cell is treated as an Octave code cell when you choose the **Code** option. It is treated as a Markdown cell when you choose the **Markdown** option. Markdown is a markup language that can create rich text output. For example, anything followed by # creates a heading, anything followed by ## creates a sub-heading, and so on. Just type the following lines in a markdown cell and execute them:

```
# Heading 1
## Heading 2
```

During our Octave demonstrations, we will mostly use markdown for headings. However, you can further explore markdown on your own. You can find more information about it at https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Working%20With%20Markdown%20Cells.html. The output of the demonstration above is shown in Figure 1-24.

Figure 1-24. *Headings in Markdown mode*

You can even change the name of the notebook file by clicking its name in the top part of the notebook. You'll see a modal box for renaming it, as shown in Figure 1-25.

Figure 1-25. *Renaming a notebook in Jupyter*

Rename it if you wish to do so. If you browse the location on disc from where you launched the Jupyter notebook from the command prompt, you will find the file with an `.ipynb` extension. It stands for **IPython Notebook**.

In the same way, you can use the Jupyter notebook for doing interactive programming with the other programming languages that support Jupyter. You will mostly use this notebook format to store your code snippets for interactive sessions. This is because everything is saved in a single file, which can be shared easily, as discussed. You will also see how to add code to `.m` files and execute it to see the visual output as you proceed further in this book.

You can clear the output of a cell or the entire notebook. In the menu bar, click the **Cell** menu. In the dropdown, **Current Outputs** and **All Output** have a **Clear** option, which clears the output of the cells. The options are shown in Figure 1-26.

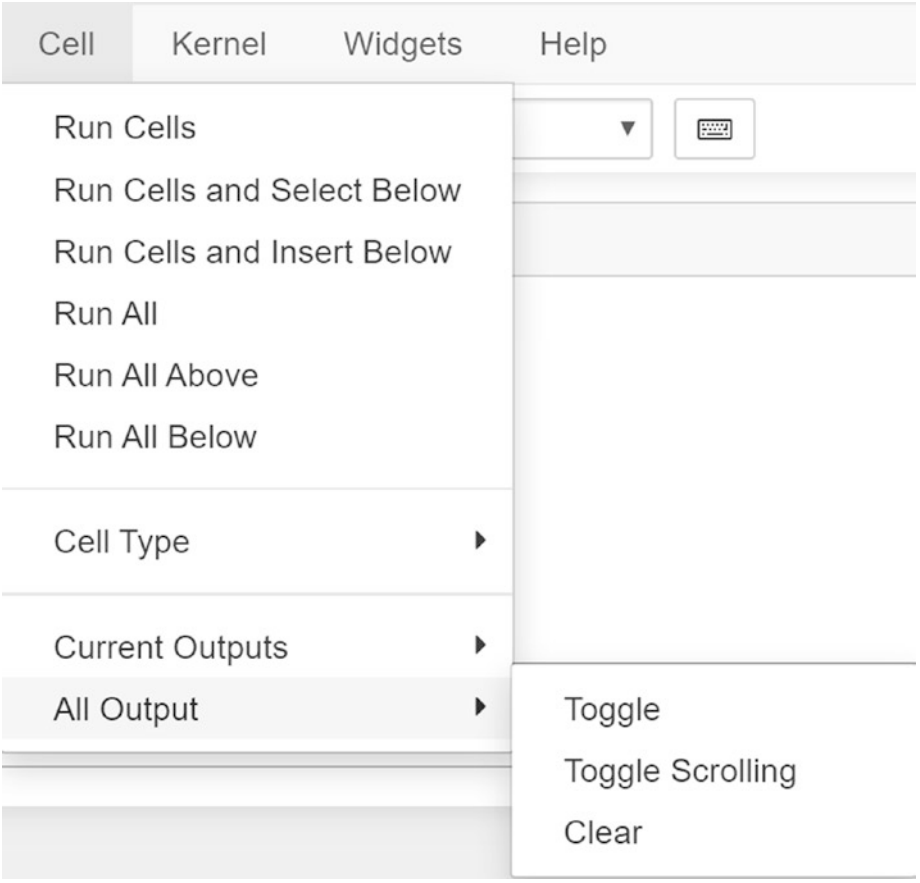

Figure 1-26. *Clearing output in Jupyter*

Summary

In this chapter, you got started with Octave installation on various platforms. You then explored how to run a simple statement in various ways. You also studied Jupyter Notebook and its use in scientific and numerical programming with Octave. This chapter was a bit light on the programming part. However, from the next chapter onwards, you will dive deeper into programming with GNU Octave.