CHAPTER 6

Introduction to Jupyter Notebook

In the previous chapter, we learned about Python. We also had a glance at how we can use Python in its REPL shell to write our code. This Python shell, however, is not the most recommended tool to use when it comes to massive machine learning programming. This is why we have developed applications like Jupyter Notebook, which aid in such programming requirements.

Jupyter Notebook is the brainchild of Project Jupyter, which is a non-profit organization founded by Fernando Pérez. It was created with the objective of developing open source software and providing services that allow multiple languages to interact with one another for effective computing.

Jupyter Notebook is an open source web-based application that allows users to create, edit, run, and share their code with ease. This application gets its name from the main languages that it supports: Julia, Python, and R.

To fully appreciate Jupyter Notebook, let us first take a look at what a "notebook" is with regard to programming.

N. Silaparasetty, *Machine Learning Concepts with Python and the Jupyter Notebook Environment*, https://doi.org/10.1007/978-1-4842-5967-2_6

Understanding the Notebook Interface

A computational notebook or a notebook interface, or quite simply a notebook, is used for literate programming, where we add a comprehensive explanation along with our program. It is a virtual notebook; i.e., it has a notebook-style GUI that provides a word processing software's functionality, along with a kernel and a shell.

A Brief History of the Notebook

The notebook interface was first introduced around 1988, when Wolfram Mathematica 1.0 was released on the Macintosh. This system allowed users to create and edit notebook interfaces through its front-end GUI.

Then came Maple, released for Macintosh with version 4.3. It provided a GUI in the style of a notebook, which became a highly acclaimed interface for programming.

As the notebook began to grow in demand, people soon began to adapt notebook-styled kernels and backends for other programming languages, such as Python, MATLAB, SQL, and so on. Thus, the computational notebook became quite popular among coders.

Features of a Notebook

The generic features of a notebook are as follows:

1. It allows us to add cells of code, which make debugging and programming easier.
2. It can be used to display visual representations of data.
3. It allows us to add text in between each cell, which makes it easier for the coder to explain the function of each line of code.

4. Items within a notebook can easily be rearranged for narrative purposes and better readability.
5. It can be used as a tool for live presentations.
6. It can be used to create interactive reports on collected data and analytical results.

Commonly Used Notebooks

Some commonly used open-source notebooks include the following:

1. Jupyter Notebook
2. IPython
3. Apache Spark Notebook
4. Apache Zeppelin
5. JupyterLab
6. R Markdown

An Overview of Jupyter Notebook

As mentioned before, Jupyter Notebook is a web-based application developed by Project Jupyter. Its aim is to enable users to, as stated on the official website, *"create and share documents that contain live code, equations, visualizations and narrative text."*

Jupyter Notebook was developed in 2014 as a spin-off of the original IPython, which is a command shell used to carry out interactive coding. With the release of Jupyter Notebook, IPython found itself competing with it, to an extent. It still remained as a kernel for Jupyter and as a shell for Python, but everything else came under Jupyter Notebook.

Fun Fact Jupyter Notebook was originally known as IPython Notebook, since it was conceived from IPython.

The official website of Project Jupyter states that Jupyter Notebook can support over forty programming languages. Each project is stored as a notebook consisting of several cells of code, graphs, texts, and equations, which can be altered easily. These notebooks can also be conveniently distributed to others.

Features of Jupyter Notebook

Apart from the generic characteristics of a computational notebook, Jupyter Notebook has the following key features:

1. Each Jupyter Notebook is a JSON document. JSON is a language-independent data format that is derived from JavaScript. It uses human-readable text to transmit data containing arrays or attribute–value pairs.
2. Each Jupyter Notebook is usually saved with a `.ipynb` extension.
3. Jupyter Notebook is similar in style to other interfaces that originated years before it, including Maple and Mathematica (from the 1980s) and SageMath (from the 2000s).
4. Jupyter Notebook was released under the modified BSD license, which provides users with minimum limitations in the usage and distribution of the software.
5. Jupyter Notebooks can easily be shared with others through email, Dropbox, GitHub, and the Jupyter Notebook Viewer.

6. Jupyter Notebook is, at present, completely free to use, and it is intended to remain free for anyone to use at any time.

Advantages of Jupyter Notebook

Jupyter Notebook has, since its release, proved to be a powerful tool for programming, especially for high-level programmers. It has a smooth and easy-to-use interface, which is great for those who are new to programming. It also allows users to create new files and folders directly on their system for easy storage of their code.

Let's take a better look at what makes Jupyter Notebook stand out as a programming application. It has the following features:

- It makes the overall programming experience better.
- It is an interactive application.
- It is open source; i.e., it is free to download, install, and use.
- It allows users to add notes, comments, and headings in between lines of code in a notebook in the markdown format, which is especially useful when sharing code with others.
- It is convenient to edit code as each line of code can be added to a separate cell, and this cell can be deleted, modified, replaced, and so on.
- It is very easy to share and distribute code with others.
- Each notebook can be converted into different file formats, like HTML, markdown, PDF, and so on.

Jupyter Notebook is in great demand now, but it did arrive pretty late into the programming world. Before its conception, there were other applications such as text editors and IDEs that coders used, and that are still in use even today.

Text Editors and IDEs

Earlier, programmers would type all of their code into a text editor like Windows Notepad. These text editors allowed them to type in their code and then install extra plugins that added bonus features. After that, they had to transfer all the code to the command prompt to run it.

Later, IDEs were created to give programmers an environment that provided them with all the features they would need to develop their code. They would not need to write and run their code in separate applications, or install new plugins each time. They could easily create, edit, debug, and run their code in a single workspace.

Let us first take a look at the classic text editors to see how they were used to program.

Getting Acquainted with Text Editors

Over the years, programmers have used all kinds of tools and environments for their code, including the very basic text editor.

The text editor is a computer program that is used, as its name suggests, to edit plain text.

They are usually provided by default with operating systems. They allow users to edit files like documentations and source code. Some examples of text editors are the TextEdit application on Mac OS, Vim on Linux, and the widely known Notepad on Windows.

Text editors are great for developers who are new to the field and who are still familiarizing themselves with coding. They are also readily available on the system. This is why most people prefer to start out with text editors.

However, with the increasing complexity of advanced programs, and especially with the introduction of artificial intelligence and machine learning, programmers felt the need to create workspaces that would make the process much easier. Hence, they came up with something called an IDE.

Getting Acquainted with the IDE

An IDE, or integrated development environment, allows us to write, edit, test, and debug our code by providing us with the necessary tools and services.

For example, with the help of an IDE, we can manage resources, debug errors, and complete our code very easily. Most IDEs are limited to a single programming language, but some allow users to work with multiple languages.

Features of an IDE

Most IDEs come with the following features:

1. **Text editor:** It allows users to write and edit code, and also provides syntax highlighting according to the language being used.
2. **Auto-completion of code:** It identifies the next possible input provided by the coder, and inserts that component accordingly. This reduces the chance of errors, and also significantly decreases the amount of time spent programming.
3. **Debugging tools:** They seek out any errors in the code and proceed to rectify them, thus saving time and making the programmer's work easier.
4. **Compilers:** They are used to translate the code into a format that the machine can understand and process.

Benefits of an IDE

Programming with an IDE is considered advantageous for the following reasons:

1. It is a single environment in which the programmer can access all the required tools and utilities.
2. It can auto-complete code and debug errors on its own, reducing the effort and time spent by the programmer.
3. It manages the syntax on its own as well, which is especially useful when it comes to indentations.
4. The code can be reverted, if needed, without any major inconvenience.
5. Project collaboration becomes easier.

Some Popular IDEs

Three of the most commonly used IDEs are the following:

- **IDLE:** IDLE, or the Integrated Development and Learning Environment, is automatically installed along with Python. It is lightweight and simple, making it easy to learn. It provides tools that are similar to those in text editors. It allows cross-platform usage and multi-window text editing. It is a good start for those who are new to IDEs.
- **Spyder:** Spyder, or the Scientific Python Development Environment, is an open source IDE. It is great for anyone who is a beginner to IDEs. It has the features of a text editor, but with a GUI, making it easy for

people to transition from the simple programming application to this more advanced one. It even allows the installation of extra plugins for added benefit. It is also visually similar to RStudio, allowing people to switch easily from R to Python.

- **Pycharm:** Pycharm is a professional Python IDE. It was made by JetBrain. It provides code editors, error highlighting, and a debugger, all with a GUI. It can also be personalized by allowing the user to change its color, theme, and so on. It integrates Numpy and Matplotlib, making it easy to work with graphs and array viewers.

Note Although IDEs have always been used to describe a working environment that allows a programmer to write and edit code, debug errors, and so on, the main definition of an IDE is slowly being altered as a result of the introduction of other tools such as Jupyter Notebook that also allow users to easily develop code.

IDE vs. Text Editor

Text editors have always been very simple to use. Even beginners to the programming world could easily use them to code, without having to worry about learning to use a new application. They required less effort in terms of understanding the programming interface.

IDEs, on the other hand, require a little bit of familiarization before a programmer can feel comfortable enough to make full use of its features. However, they have extra capabilities and tools that simplify the programming experience.

The conclusion: It all depends on our need and preference. If we don't want to spend time learning how to use an application, and would rather make use of a simple interface for our code, we can use a text editor. And, if we want to invest a little time in learning how to use an application, which will then help us later with the rest of our programming requirements, we can use an IDE.

Now that we know what text editors and IDEs are, we can see how the notebook interface, and specifically Jupyter Notebook, is more beneficial to programmers compared to similar applications.

Jupyter Notebook vs. Other Programming Applications

Why would we want to choose Jupyter Notebook over other programming applications? Well, let's have a look at the following differences between Jupyter Notebook and other such applications:

- **Tools:** Jupyter Notebook provides users with tools and utilities that make the programming experience much faster and easier. Compared to other IDEs, Jupyter Notebook has more services available.
- **Graphical User Interface:** The GUI of Jupyter Notebook varies because it is meant to look like a notebook and not like a general IDE. This makes it easier on the eye and quite simple to understand.
- **Usability:** It is easier to use Jupyter Notebook compared to other IDEs because of its easily accessible features.

- **Learning:** Compared to other IDEs, Jupyter Notebook may take a little time to grasp, just because of how different it is from what we are used to. However, once we do learn it, it becomes extremely convenient to use.
- **Web-based:** Jupyter Notebook runs on the browser, unlike other IDEs, which work on the local system.
- **Visualization:** Although some IDEs provide users with a great platform for visualization, other IDEs don't. Jupyter Notebook does, though, thus making it easier for a programmer to use plots and other such visualization techniques.

In this way, Jupyter Notebook outdoes its competitors in the programming world.

Jupyter Notebook sounds like a blast, doesn't it? Well, it is! Once we get the hang of it, we can thoroughly enjoy programming with it. Let's now learn how to set up our Jupyter Notebook environment on our machine.

Installing Jupyter Notebook

As mentioned in the previous chapter, one advantage of using Anaconda is that the installation of Jupyter Notebook becomes quite an easy task to achieve. There is no hassle of navigating through various applications just to download it. All we need to do is the following:

1. Open the Anaconda Navigator.
2. Select the working environment, as shown in Figure 6-1.

3. Click on the option to install Jupyter Notebook.

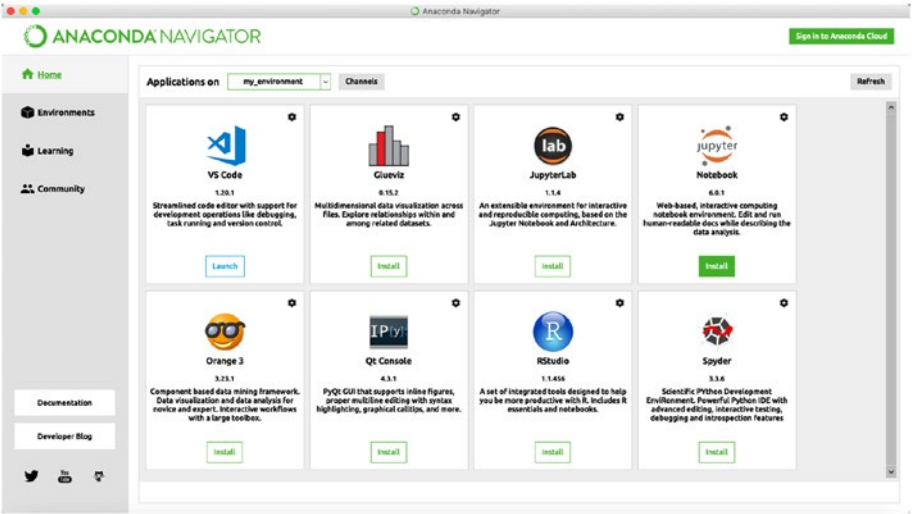

Figure 6-1. *Installing Jupyter Notebook*

Et voila! Jupyter Notebook is now ready for use. In the next few sections, we will explore its interface so as to get ourselves comfortable with the layout and working of the application.

Note When installing, we must make sure that we install Jupyter Notebook and not JupyterLab. There's a difference!

Launching Jupyter Notebook

The first thing we will need to do is select our working environment. Here, I have chosen `myenv`.

Next, we need to open up the Jupyter Notebook window. We can do this by opening the Anaconda application and then clicking on "Launch" under the Jupyter Notebook icon.

Since Jupyter Notebook is a web-based application, it opens in our browser. The first window that opens is a dashboard, which gives us a glimpse of our work so far, including files, folders, and notebooks. It will look like Figure 6-2.

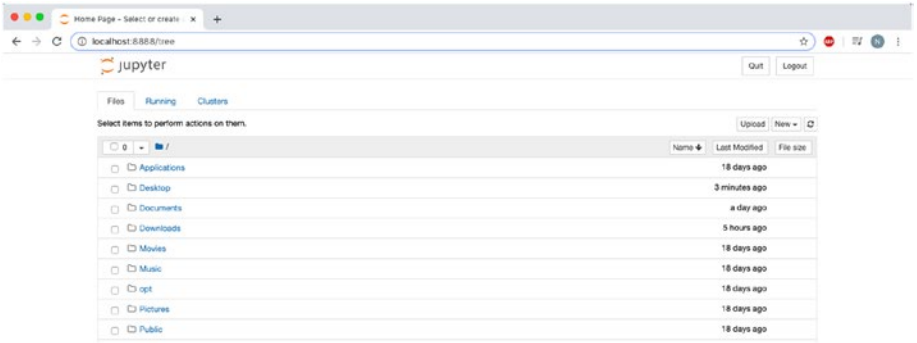

Figure 6-2. *The Jupyter Notebook Dashboard*

The URL bar contains a link that represents the notebook server, and indicates that it is running from our local machine. The link will appear like this this - `http://localhost:8888/tree`.

The rest of the dashboard is quite self-explanatory, but we will run through it anyway. Here's a breakdown of some of the basic but most important features of the Jupyter Notebook interface, as shown in Figure 6-3:

Figure 6-3. *Some important features of the Jupyter Notebook dashboard*

1. The Logout button allows us to log out of our Jupyter Notebook session.
2. The Upload button allows us to upload a readily available Jupyter Notebook that we can use.
3. The New button allows us to create a new Python notebook, file, folder, or terminal.
4. The File tab shows us an ordered list of all our files and folders.
5. The Running tab shows us any terminals or notebooks that are open and running.
6. The Name button allows us to toggle the way our list of files and folders is displayed; i.e., in ascending or descending alphabetical order.
7. We can even select the "Last Modified" option to display our items based on the last time that they were modified.
8. The little check-box option with a "0" beside it allows us to select all folders, notebooks, files, and items that are open and running. We can even select all of the items at once.
9. In our list of items, the ones with a folder icon next to them represent the folders that we have on our computer, as shown in Figure 6-4.

Figure 6-4. *List of folders*

10. Once we create Jupyter notebooks and text files, they will begin to appear on the dashboard. The items with a page icon next to them represent the documents that have a `.txt` extension, and the ones with a notebook icon next to them represent the Jupyter notebooks, which have a `.ipynb` extension, as shown in Figure 6-5.

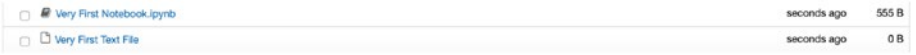

Figure 6-5. *A notebook and a file*

Now that we are aware of the general features of the Jupyter Notebook interface, let's see what happens when we select an item from our list by clicking on the check box next to it. When we select an item, we will have a number of available options, as shown in Figure 6-6:

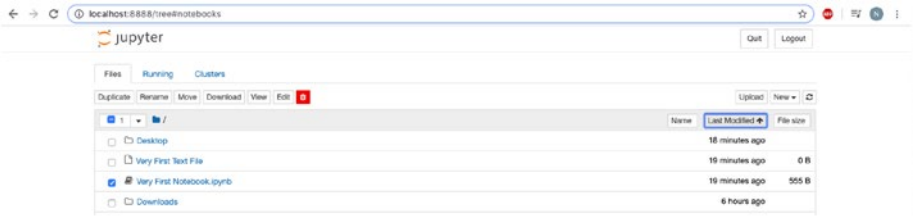

Figure 6-6. *Controls available for each item*

1. We can Rename the item.
2. We can Duplicate the item to make another copy of it.
3. We can Move the item to another location.
4. We can Download the item.
5. We can View the item, which will open in a new tab in our browser window.
6. We can Edit the item.
7. We can Delete the item by clicking on the red trash can symbol.
8. We can Shutdown a notebook that is open and running, as shown in Figure 6-7.

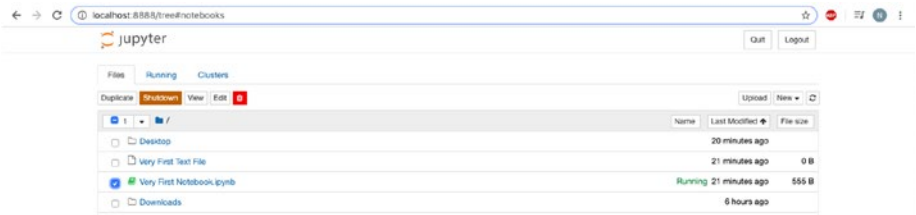

Figure 6-7. *Option to shut a notebook down*

9. We can even select several items at the same time and perform any available action on them.

Let us now create a brand new Jupyter notebook and explore all the features within it.

Inside a Jupyter Notebook

To create a new Jupyter Notebook, all we have to do is click on 'New' on the dashboard, and then select the kernel of our choice. Here, we select the 'Python 3' kernel, as shown in Figure 6-8.

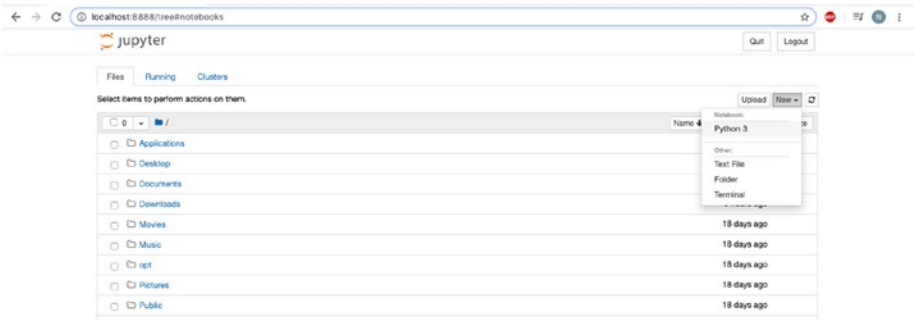

Figure 6-8. *Opening a new Jupyter Notebook with a Python 3 Kernel*

We will get a new tab with a notebook user interface (UI) that looks like Figure 6-9.

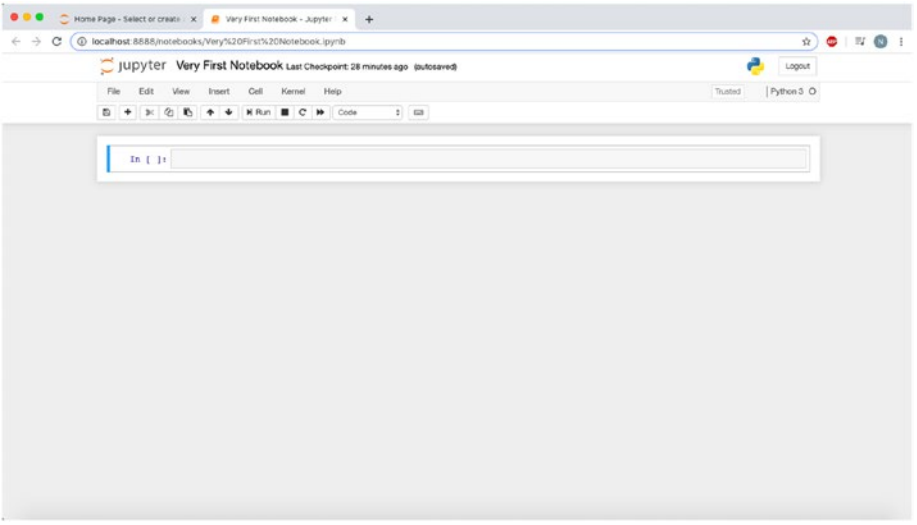

Figure 6-9. *A new notebook*

The notebook UI is quite self-explanatory as well. However, just like before, we will have a quick run-through of all its main features.

1. At the top, the title of the notebook is displayed. It starts out as "Untitled," and when we click on it, we can change the name based on our preference, as shown in Figure 6-10.

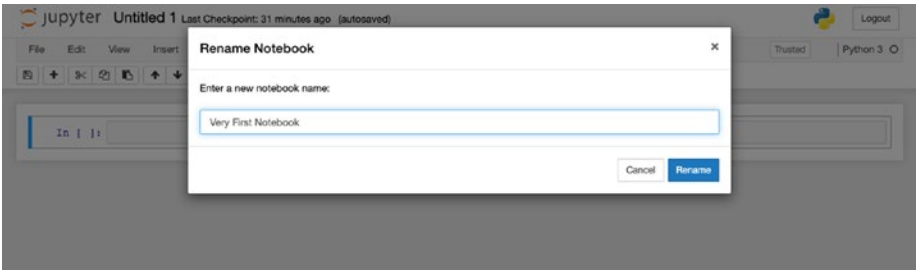

Figure 6-10. *Renaming a notebook*

2. Next to the title of our notebook, we will see "Last Checkpoint," with a timing. That indicates the last time the notebook was auto-saved.
3. Below this is the menu bar, containing a series of drop-down menus, as shown in Figure 6-11.

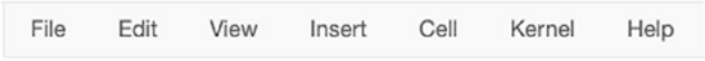

Figure 6-11. *The Menu Bar*

4. After this comes the tool bar, containing tools that we will need as we use Jupyter Notebook, as shown in Figure 6-12. We can hover over each tool icon to know what it does.

Figure 6-12. *The Tool Bar*

5. Finally, we have the area where we type in all of our input and view our output, as shown in Figure 6-13.

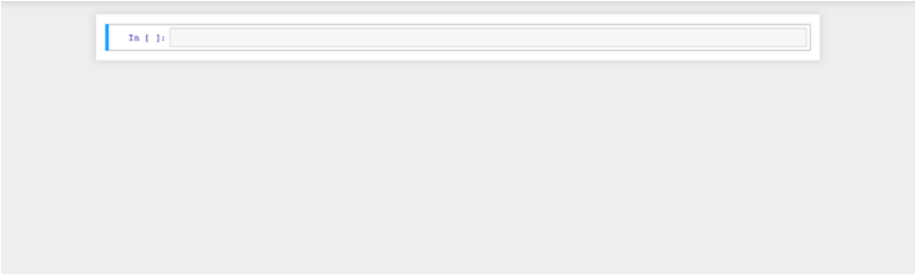

Figure 6-13. *This is where the different kinds of cells appear, allowing us to enter our input*

You might have noticed that the menu bar contains the Cell menu and the Kernel menu. These are two terms that are very important in the Jupyter Notebook environment.

Cell

A cell is nothing but the box in which we type all our input, which can either be code, regular text, or headings.

When we first open our Jupyter notebook, we will see that the first cell is a "Code" cell. This cell allows us to enter the commands, functions, variables, constants, and all other inputs that are a part of our program. When we execute this cell, the output, if any, is displayed beneath it.

Let's try typing the following in the "Code" cell:

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

Now, we can execute the cell by clicking on the Run button from the tool bar. We can also just use the keyboard shortcut, which is `Shift+Return`. We will find that the code line is executed and the output is printed out right below the cell, as shown in Figure 6-14.

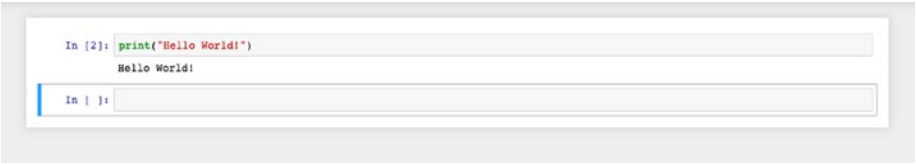

Figure 6-14. *Executing a code cell*

The second type of cell is a "Markdown" cell. Markdown is a formatting syntax that is used to style plain text. Thus, this cell is used to enter any text that is not a part of the code. This could be explanations or notes that are needed in between the code, either to make it easier for us as we program, or to make it more comprehensive for someone else who is going through it. Once we type in all the necessary text and execute the cell, it becomes a regular text box that is visible in our program.

Let's try this text in the "Markdown" cell:

```
Hello World!
```

Our Markdown cell will display an output as shown in Figure 6-15.

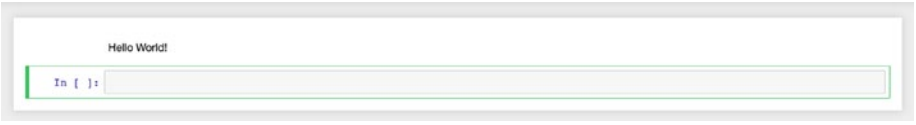

Figure 6-15. *Entering regular text*

The third type of cell is the "Heading" cell. This cell is used to add headings throughout our program. This allows us to make our entire program look much more organized, especially if we have more than one program running within the same notebook.

Let's try typing this in the "Heading" cell:

```
My Program
```

The heading will appear as shown in Figure 6-16.

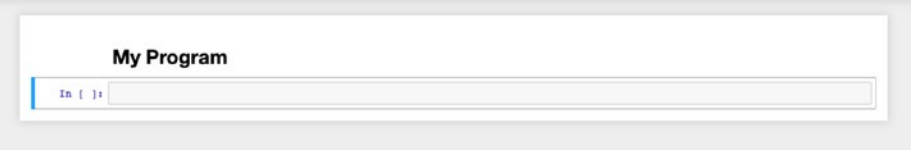

Figure 6-16. *Entering a heading*

We can also just open a regular Markdown cell and type the following in -

```
# My Program
```

The '#' symbol is used to convert the sentence into a heading. The number of times we use the symbol indicates the level of the heading. For example, a single hash is used to obtain a level one heading.

We can change the type of cell that we want to use by selecting it from the list of options in the Tool Bar.

Kernel

A kernel runs the code that is contained within the Jupyter notebook.

A kernel is not limited to a single cell, but rather to the entire notebook. When we execute code in a selected cell, the code runs within the kernel and sends any output back to the cell to be displayed.

There are kernels for more than a hundred languages, including Python, C, R, and Java. When we create a new notebook from the Jupyter Notebook dashboard, we are basically selecting our kernel by choosing the Python version that we desire to use. In this case, when we select "Python 3," we are telling our system to open a Python 3 kernel.

Now that we have some idea of what a cell and a kernel are, let's come back to the menu bar and explore what the Cell and Kernel drop-down menus allow us to do.

The Cell Drop-Down Menu

Figure 6-17 shows the different options available within the Cell drop-down menu.

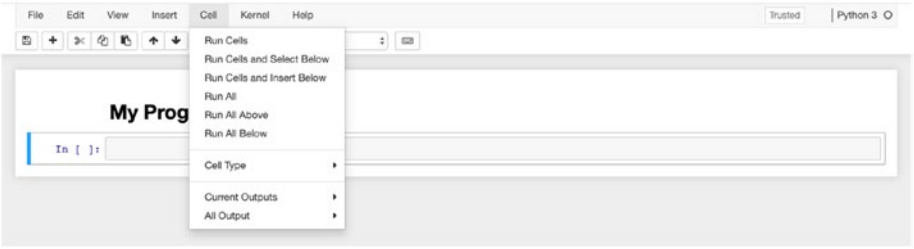

Figure 6-17. *Cell drop-down menu*

1. **Run Cells:** This executes the code that is in the selected cell or cells, and gives an output, if any.
2. **Run Cells and Select Below:** This executes the selected cells and then selects the cell below them.
3. **Run Cells and Insert Below:** This executes the selected cells and then inserts an extra cell just below them.
4. **Run All:** This executes all the cells in the notebook.
5. **Run All Above:** This runs all the cells that are above the selected cell.

6. **Run All Below:** This runs all the cells that are below the selected cell.
7. **Cell Type:** This allows us to select the type of cell you require.
8. **Current Outputs:** This gives us the option to either Toggle, Toggle Scrolling, or Clear the selected output.
9. **All Output:** This gives us the option to either Toggle, Toggle Scrolling, or Clear all the output in the notebook.

The Kernel Drop-Down Menu

Figure 6-18 shows the different options available within the Cell drop-down menu.

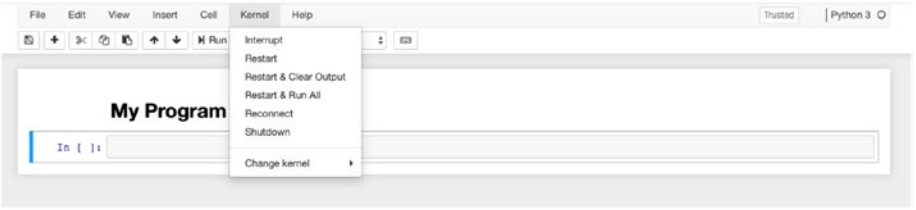

Figure 6-18. *Kernel drop-down menu*

1. **Interrupt:** This interrupts the running process as the code is being executed.
2. **Restart:** This restarts the entire kernel, retaining the previously obtained outputs.
3. **Restart and Clear Output:** This restarts the entire kernel, clearing the previously obtained outputs.

4. **Restart and Run All:** This restarts the entire kernel and once again proceeds to execute all the cells.
5. **Reconnect:** This allows the kernel to reconnect.
6. **Shutdown:** This shuts the active kernel down.
7. **Change Kernel:** This allows us to change our kernel to any version or language that we want.

There you have it! This was an overview of some of the most basic but important features of Jupyter Notebook.

Now that you are familiar with the working environment of Jupyter Notebook, let's go ahead and practice some Python programming with the help of Jupyter Notebook.

Additional Information

The Jupyter Project is a very interesting initiative, especially for data scientists and machine learning enthusiasts who need a reliable and convenient space to work on their projects. Let's have a look at two more very useful features that come under Project Jupyter, and that can be useful to some of us in our machine learning journey.

JupyterHub

JupyterHub allows multiple users to share resources in order to program. Each user has their own workspace where they can code without worrying about installations and maintenance.

It can run either on a user's system or on the cloud. It is customizable, flexible, portable, and scalable, making it a great interface for programmers. It also has its own community for users to discuss and contribute.

Jupyter nbviewer

Jupyter nbviewer is a free and publicly available instance of nbviewer, which is a web-based application that allows us to view a notebook as a static HTML web page. It also provides us with a link that we can use to share the notebook with others.

Apart from viewing a single notebook, we can also view notebook collections. These notebooks can even be converted into other formats.

Voila

Voila is used to convert a Jupyter notebook into a stand-alone web application that can be shared with others. It consists of an interactive dashboard that is customizable and allows users to view the notebook in a secure environment.

It can work in any Jupyter kernel, independent of the type of programming language used. It is a great choice for non-technical users who desire to view the results of the notebook without having to see the code cells or execute the code.

Google Colaboratory

Google's Colaboratory or Colab is a free online Jupyter environment. It runs in the cloud and stores its notebooks to the user's Google Drive.

As of October 2019, Colab mainly supports Python 2 and Python 3 kernels.

However, it is also possible for Colab to support R, Swift, and Julia.

Keyboard Shortcuts

First of all, you need to know that there are two modes of working with Jupyter Notebook, as follows:

- Command Mode, which allows us to navigate around the notebook with our arrow keys.
- Edit Mode, which allows us to edit the selected cell.

Table 6-1 lists some of the most useful keyboard shortcuts that we can use while working with Jupyter Notebook.

Table 6-1. *Keyboard Shortcuts for Jupyter Notebook*

Mac	Windows and Linux	Action
Cmd + Shift + P	Ctrl + Shift + P	Access keyboard shortcuts
Shift + Enter	Shift + Enter	Executes the code
Esc	Esc	Enters Command Mode when in Edit Mode
Enter	Enter	Enters Edit Mode when in Command Mode
A	A	Inserts a new cell above the selected cell while in Command Mode
B	B	Inserts a new cell below the selected cell while in Command Mode
D + D (Press D twice)	D + D (Press D twice)	Deletes the selected cell while in Command Mode
Shift + Tab	Shift + Tab	Displays the available documentation for the item entered into the cell
Ctrl + Shift + -	Ctrl + Shift + -	Splits the selected cell into two at the point where the cursor rests while in Edit Mode
F	F	Finds and replaces code while in Command Mode
Shift + J / Shift + Down	Shift + J / Shift + Down	Selects the chosen cell as well the cell below it
Shift + K / Shift + Up	Shift + K / Shift + Up	Selects the chosen cell as well as the one above it
Shift + M	Shift + M	Merges multiple cells

Summary

In this chapter, we have gained an understanding of the importance of the notebook interface, when compared to IDEs and text editors. We then explored the Jupyter Notebook application, its features, and its user interface.

The great thing about Jupyter Notebook is that it looks quite complex and technical, but in reality it is not too difficult to use, once you get the hang of it. Overall, it is a great tool to use for all your programming purposes. Not just that, it can also be used to display your results and present your output in a manner that is not too hard on the eyes.

In the next few chapters, we will begin some actual programming with the help of Jupyter Notebook. We will get a feel of how we can use the notebook interface effectively to enter, run, and debug our code. And, finally, once we have gained some familiarity with Jupyter Notebook, we will proceed with using the interface to develop our machine learning models.

Quick Links

Learn more about Project Jupyter: https://jupyter.org/about

Jupyter Documentation: https://jupyter.org/documentation

Try Jupyter: https://jupyter.org/try

JupyterHub: https://jupyter.org/hub

Jupyter Notebook Viewer: https://nbviewer.jupyter.org/

Google Colab: https://colab.research.google.com/notebooks/intro.ipynb