Introduction to Jupyter Notebooks

What is a Jupyter Notebook?

- Jupyter Notebooks started as iPython, a tool mainly for Python programming.
- It got its name "Jupyter" when it began supporting more languages like **Julia**, **Python**, and **R**.
- Now, it works with many programming languages, not just Python.

What makes Jupyter Notebooks useful?

- It's a **browser-based tool**, meaning you don't need to install heavy software—just use it through your browser.
- A Jupyter Notebook is like a **digital scientist's lab notebook**, where you write and store code, notes, data visualizations, and results all in one place.
- You can write **narrative text** to explain your work, include **code blocks** to test things, and see the **output** (like graphs and tables) right there in the notebook.
- After you're done, you can export your work as a **PDF** or **HTML** to share it easily.

JupyterLab vs. Jupyter Notebooks

- **JupyterLab** is an upgraded version of Jupyter Notebooks.
- It lets you work on multiple notebooks and files at once, with features like text editors, code terminals, and much more.
- It's open-source and supports many file formats (like CSV, JSON, PDF).

How do you use Jupyter Notebooks?

- You can run Jupyter Notebooks without installing anything on your computer. Services like **IBM's Skills Network Labs** and **Google Colab** allow you to access Jupyter notebooks on the cloud.
- If you want to install it locally, you can use **pip** (a Python package installer) or download **Anaconda**, which comes with Jupyter pre-installed.

In Summary:

- **Jupyter Notebooks** are key for **data science**, helping you document and share your work.

- **JupyterLab** extends Jupyter Notebooks, allowing you to work with more files and code in one flexible space.
- You can access Jupyter through the cloud or install it locally using pip or Anaconda.

Getting Started with Jupyter

Basic Operations in Jupyter Notebooks:

- **Running a cell**: To run code in a Jupyter notebook, select the cell you want to run and click the **Run** button at the top. You can also use the shortcut **Shift + Enter** to run a selected cell.
- **Running multiple cells**: If you have more than one cell, you can click **Run All Cells** to execute all the code in the notebook at once.
- **Inserting a new cell**: To add a new cell, click the **plus (+) button** in the toolbar. This is useful when you want to add more code or notes.
- **Deleting a cell**: If you want to delete a cell, highlight it, then click **Edit** on the top menu, and select **Delete Cells**. Alternatively, press the **D** key twice on your keyboard to quickly delete it.
- **Moving cells**: You can also move cells up or down by highlighting the cell and dragging it to the desired position.

Working with Multiple Notebooks:

- Jupyter lets you work with multiple notebooks at once.
- To open a new notebook, click the **plus button** in the toolbar and select the file you want to open. You can open several notebooks side by side and even compare or run code between them.

Presenting Your Work in Jupyter:

- Jupyter Notebooks aren't just for writing and running code; you can also **present your results** directly within the notebook.

- Use **Markdown** to add headings, titles, and text descriptions to make your work look professional. To switch to Markdown mode, click **Code** in the toolbar and choose **Markdown**.
- You can also turn your notebook into a **presentation** by converting cells and outputs into slides using Jupyter's built-in functionality. This is useful for sharing your work with others, combining code, visualizations, and explanations.

Shutting Down Your Notebook:

- Once you've finished your work, you can **shut down your notebook** to free up memory.
- Click the **stop icon** (the second icon from the top on the sidebar). You can stop individual notebooks or all of them at once.
- After shutting down, you'll see "**no kernel**" at the top right, confirming that the notebook is no longer active.

In Summary:

- You now know how to **run, delete, and insert cells**, work with multiple notebooks, **present your work** using Markdown and slides, and **properly shut down** your notebook sessions when you're done.

Understanding Jupyter Kernels

What is a Jupyter Kernel?

- A **kernel** is essentially the **computational engine** behind your Jupyter Notebook—it runs the code you write and gives you the results.
- When you open a notebook, a related kernel automatically starts in the background to process the code.

Different Kernels for Different Languages:

- **Jupyter** supports many programming languages, not just Python. Some popular kernels relevant to **Data Science** include:
    - **Python**: The most common language for data science.

    - **Julia**, **R**, **Swift**, and **Apache** are also available in the **Skills Network** virtual environment.

How to Use Kernels:

- You can select the kernel you want to use when starting a notebook, depending on the language you're working with.
- To see which kernel is running, check the **top-right corner** of your notebook. It will show the name of the kernel currently in use.
- If you want to change the kernel, click on the top-right kernel name and choose another one from the **drop-down menu**.

Installing Kernels Locally:

- If you're running Jupyter on your own machine (instead of in a virtual environment like **Skills Network Labs**), you may need to manually install other kernels for different languages using the **command line**.

In Summary:

- A **Jupyter kernel** is what runs your code and provides the output.
- You can work with **multiple programming languages** by switching between kernels in Jupyter.
- In a virtual environment, several kernels are pre-installed, but if you're using Jupyter locally, you might need to install additional ones.

Understanding Jupyter Architecture

Jupyter's Two-Process Model:

- Jupyter operates on a **two-process model**: the **kernel** and the **client**.
    - The **client** is the interface that you interact with, which in Jupyter's case, is your **browser**.

    - The **kernel** is the engine that runs the code you write, processes it, and sends the results back to the client (your browser) for display.

How Jupyter Notebooks Work:

- **Notebooks** hold your code, metadata, outputs, and other content.
- When you **save** a notebook, it is stored as a **JSON file** with a `.ipynb` extension (pronounced **dot I PYNB**).
- The **Notebook server** is responsible for saving and loading these notebook files on your disk.
- The **kernel** processes and runs the code cells when you execute them.

Converting Files in Jupyter:

- Jupyter provides a tool called **NB Convert** that allows you to convert notebook files into other formats, such as **HTML**.
    - First, a **preprocessor** modifies the notebook content.
    - Then, an **exporter** converts the notebook to the desired file format.
    - Finally, a **postprocessor** completes the final touches, such as generating links or other finishing steps, to provide the finished output.

In Summary:

- Jupyter has a **two-process architecture** with the **client (your browser)** and the **kernel** (which runs code).
- The **Notebook server** is in charge of saving and loading notebooks, which are stored in a `.ipynb` format.
- **NB Convert** allows you to transform your notebooks into other formats like HTML using preprocessing, exporting, and postprocessing steps.

Overview of Anaconda Jupyter Environments

What is Anaconda?

- **Anaconda** is a free, open-source platform that supports Python and R—two of the most popular languages in data science and machine learning.
- It comes with over **1,500 libraries** (like **NumPy, Pandas, Matplotlib**) that are useful for data science.
- Anaconda also provides **community support** for Python users.

Anaconda Navigator:

- **Anaconda Navigator** is a graphical interface (GUI) that lets you easily install new packages without using command line commands.
- Through this interface, you can launch various applications like **JupyterLab** or **VS Code**.

JupyterLab vs Jupyter Notebook:

- **JupyterLab** is an advanced, web-based application built on top of Jupyter Notebook. It allows you to write code, make interactive visualizations, and include equations and rich text.
- Anaconda's version of JupyterLab comes preloaded with key Python libraries, making it easier for data science projects.

How to Use JupyterLab in Anaconda:

1. To launch JupyterLab in Anaconda Navigator, click the **Launch** button.
2. If the **Launch** button isn't available, click **Install** first, and then Launch.
3. To start a notebook, search for **Jupyter Notebook(anaconda3)** in your search bar and press Enter.
4. Once opened, you can create a new notebook by selecting **New > Python 3**.

Working with Notebooks:

- **Code Cells:** You can write and run Python code in these cells.
- **Markdown Cells:** These are for writing formatted text, like headings or explanations.

VS Code and Jupyter:

- **VS Code** is a free, open-source editor often used for debugging and other coding tasks.

- It supports many features, like **syntax highlighting**, **auto-indentation**, and works across platforms like **Linux, Windows, and macOS**.
- To run Jupyter Notebooks in VS Code, you need to install Python-related extensions.

Installing VS Code:

- You can install VS Code separately from Anaconda by visiting **code.visualstudio.com**, or you can launch it directly from the Anaconda Navigator.

Summary:

- Jupyter is popular for combining code, visualizations, and text in one document, making it great for data science projects.
- **Anaconda Navigator** simplifies working with Jupyter environments, offering both **JupyterLab** and **VS Code** for creating and running notebooks.
- While you can install Jupyter and VS Code outside of Anaconda, doing so within Navigator ensures all components are pre-configured and ready to go.

Cloud-Based Jupyter Environments Overview

What is a Computational Notebook?

- Computational notebooks combine **code**, **outputs**, **explanatory text**, and **multimedia** into one document.
- **Jupyter Notebook** is one of the most popular types of computational notebooks, supporting many programming languages.

Popular Cloud-Based Jupyter Environments:

1. **JupyterLite:**
    a. **JupyterLite** is a lightweight version of JupyterLab that runs **entirely in the browser**.
    b. It doesn't need a dedicated server—just a **web server**—making it ideal for static websites.

c. **JupyterLite** supports various **visualization libraries** like Altair, Plotly, and ipywidgets, making it great for interactive visualizations.
d. It comes with the latest JupyterLab features and improvements.
e. To use JupyterLite, go to **jupyter.org/try-jupyter/lab** and select **Python (Pyodide)**.
    i. **Python Pyodide** and **Python Pyolite** are the common kernels used in JupyterLite.
    ii. **Pyolite** is the default kernel, designed to handle intensive computations efficiently within the browser.

2. **Google Colaboratory (Google Colab):**
    a. **Google Colab** is a free cloud-based Jupyter environment that integrates with **Google Drive** and **GitHub**.
    b. You can upload, share, and execute notebooks without any setup or installation.
    c. Google Colab is ideal for quick, "**on the fly**" development, especially for **machine learning** and **data science** projects since it comes pre-installed with popular libraries like **scikit-learn** and **matplotlib**.
    d. To open a Colab notebook, go to **Google Drive**, click **New**, select **More**, and then click **Google Colaboratory**.
    e. Colab supports **code cells** and **Markdown cells**, allowing you to mix text and code easily.

Key Takeaways:

- **Jupyter** is a widely-used computational notebook platform due to its support for multiple programming languages.
- Popular cloud-based environments include **JupyterLite** and **Google Colab**.
- JupyterLite runs entirely in the browser, making it lightweight, while Google Colab is a powerful, pre-configured tool for machine learning and data science in the cloud.