Trisha

AI/ML Enthusiast | Intern @CodroidHub | First-Year Engineering Student

Artificial Intelligence/Machine Learning UpSkills Notebook

From Basics to Real-World — Starts Your ML Journey

Agenda

Welcome! This notebook will help you learn Python and Jupyter Notebook step by step. You will learn how to set things up, understand how they work, and use them well.

What's Inside

- **Set Up Your Environment:**
  Learn how to install Python and Jupyter Notebook on Windows, Mac, or Linux.

- **What is Jupyter Notebook?**
  See what Jupyter Notebook is, how it works, and why it's good for coding.

- ⌨ **Useful Shortcuts:**
  Find simple keyboard shortcuts to work faster in Jupyter Notebook.

- **Python Versions:**
  Understand different Python versions and which one you should use.

- **How Python is Used:**
  Find out where Python is used in real life — like websites, AI, and science.

Summary

Click the topics above to go straight to that part in the notebook.

Build the Foundation — Your Python Environment Starts Here.

Jupyter Notebook Setup Guide

To use Jupyter Notebook smoothly, you'll need **Python** or **Anaconda** installed on your computer. Below are simple steps to set it up on **Windows**, **macOS**, or **Linux**.

✅ Option 1: Install Python & pip

(Recommended for developers)

Step 1: Download & Install Python

Visit: python.org/downloads

Choose your operating system (Windows, macOS, or Linux) and download the installer.

> **Important:** During installation, make sure you check the box `Add Python to PATH` before clicking Install.

⚙ Step 2: Check Python Installation

Open your **Command Prompt** (Windows) or **Terminal** (macOS/Linux) and run:

```
python --version
pip --version
```

This confirms Python and pip are installed correctly.

⚙ Step 3: Install Jupyter Notebook

In the same terminal, type:

```
pip install notebook
```

⚙ Step 4: Launch Jupyter Notebook

To open Jupyter Notebook, run:

```
jupyter notebook
```

Your default web browser will open at `http://localhost:8888`.

Use the interface to create or open `.ipynb` files.

✅ You're Ready!

That's it! You're all set to start working with Jupyter Notebook. Happy coding!

> **OR — Alternative Option Below**
> You may follow the next method as an alternate setup approach.

Option 2: Anaconda Installation Guide (Windows / macOS / Linux)

Anaconda is a popular Python distribution that comes with Jupyter Notebook, **conda**, and many useful data science libraries already installed. It's a great choice for beginners and professionals.

Step 1: Download Anaconda

- Go to the official website: anaconda.com/products/distribution
- Click **Download** under **Individual Edition**
- Pick the installer for your operating system:
  - **Windows:** `.exe` file
  - **macOS:** `.pkg` file
  - **Linux:** `.sh` file

Step 2: Install Anaconda

For Windows:

- Run the `.exe` file you downloaded
- ✅ Check the box: “Add Anaconda to my PATH environment variable” (recommended)
- Click **Next** and follow the steps (default settings are fine)

For macOS:

- Open the `.pkg` file
- Follow the instructions in the installer
- Restart your terminal when done

For Linux:

- Open your terminal and go to the folder where the file is downloaded
- Run this command (replace `<version>` with your file’s version):

```
bash Anaconda3-<version>-Linux-x86_64.sh
```

- Follow the prompts on your screen

✅ That’s it! Anaconda is now ready, and you can start using Python and Jupyter Notebook easily.

Your Interactive Coding Lab — Jupyter Notebooks.

What is Jupyter Notebook?

Jupyter Notebook is a powerful, open-source interactive development environment (IDE) primarily used for **data analysis**, **scientific computing**, **machine learning**, **exploratory programming**, and **education**.

It supports mixing **code**, **visualizations**, and **narrative text (Markdown)** in a single, shareable document, making it ideal for both prototyping and presentation.

Key Features

- **Interactive Code Execution**: Run code one cell at a time and see output immediately.
- **Language Support**: Though widely used for Python, it supports over 40 programming languages via *Jupyter kernels* (e.g., R, Julia, Bash).
- **Markdown & LaTeX Support**: Ideal for writing rich documentation and mathematical formulas.
- **Data Visualization**: Integrates seamlessly with libraries like `matplotlib`, `seaborn`, `plotly`, and `bokeh`.
- **Notebook Sharing**: Save and share `.ipynb` files or export as PDF/HTML slides.
- **Extensibility**: Supports plugins like **nbextensions**, **widgets**, **voilà**, and more.

Typical Use Cases

- Data cleaning and transformation
- Numerical simulation and modeling
- Visualization and storytelling
- Machine learning model training and evaluation
- Teaching and interactive assignments

How Jupyter Notebook Compares with Other Environments

Feature	Jupyter Notebook	Google Colab	VS Code (Python Extension)

Setup Required	Yes	No (cloud-based)	Yes
Runs in Browser	Yes (locally)	Yes (cloud)	No (runs locally with GUI)
Free GPU Access	No	✅ Yes (limited quota)	❌ No
Offline Usage	✅ Yes	❌ No	✅ Yes
Integrated Terminal & Git	Limited	No	✅ Full terminal, Git support
Extensibility	High (nbextensions, widgets, etc.)	Medium	High (extensions, LSPs, linters)
Markdown & LaTeX Support	✅ Yes	✅ Yes	✅ (with Markdown Preview plugin)
Deployment Ready	❌ No	❌ No	✅ Yes (production-ready development)

Summary

Jupyter Notebook excels in **interactive computing**, **data analysis**, and **rapid experimentation**. It's favored by data scientists, educators, and researchers for its ability to combine computation with documentation in one coherent file.

While tools like **VS Code** are better suited for **full-stack development and large-scale projects**, and **Google Colab** offers convenient **cloud-based execution with free GPU**, Jupyter remains a go-to tool for **local prototyping, research notebooks, and presentations**.

Work Smarter — Jupyter Notebook Shortcuts Guide

Useful Jupyter Notebook Shortcuts

Make sure you are in **Command Mode** (press `Esc`) before using these shortcuts.

Add New Cells

Action	Shortcut
Add cell below	B
Add cell above	A

Switch Cell Type

Action	Shortcut
Convert to Markdown	M
Convert to Code	Y

Run & Move

Action	Shortcut
Run selected cell	Shift + Enter
Run and insert cell below	Alt + Enter
Run and stay in same cell	Ctrl + Enter

Other Handy Shortcuts

Action	Shortcut
Delete cell	D then D
Copy cell	C
Paste cell	V
Cut cell	X
Undo deleted cell	Z
Move cell up	K then Shift+K
Move cell down	J then Shift+J

View All Shortcuts

To see a full list of shortcuts:
Press `H` in Command Mode (press `Esc` to exit editing first).

> Python Versions: Evolution of Simplicity, Power, and Performance.

Python Versions

Python Versions: Overview & Key Comparisons

Python is a simple, general-purpose programming language. It is popular because it is easy to read, flexible, and has many libraries. Over time, Python has improved a lot with new versions.

Major Python Versions

Python 2.x (Old Version, Ended)
- Released in 2000
- Syntax was not as clear as Python 3
- Support ended on January 1, 2020
- Many popular tools (like TensorFlow, Pandas) stopped supporting it

Python 3.x (Current Version)
- Released in 2008
- Has cleaner syntax and better support for Unicode (languages/text)
- Has many new libraries
- Recommended for all new projects

Key Differences: Python 2 vs Python 3

Feature	Python 2	Python 3
print Statement	print "Hello"	print("Hello")
Integer Division	3/2 = 1	3/2 = 1.5
Unicode Support	Limited	Default str is Unicode
xrange()	Exists	Replaced with range()
Community Support	Ended in 2020	Active
Package Compatibility	Getting lower	Modern and increasing

Python 3.x Minor Versions

Version	Highlights	Status
3.6	f-strings, better type hints	Ended
3.7	Data classes, built-in breakpoint()	Ended
3.8	Walrus operator (`:=`), positional-only args	Ended
3.9	Dictionary merge (`|`), better type hints	Ended
3.10	Pattern Matching (match-case)	Active
3.11	Faster, better error messages	Active
3.12	Runtime checks, faster code	Latest (2023+)

⚠ Best Practices

- Always use the latest stable version (3.12+ is best) for new work.
- For machine learning tools (like TensorFlow), use versions 3.8–3.11.
- Check your Python version by running:

```
python --version
```

> Behind Every Smart System, There's Python — The Engine of Modern Technology.

Where Python is used?

Python is one of the most popular and flexible programming languages in the world. Many small and big companies use it because it is simple to write, easy to read, and has many useful libraries.

✅ Common Use Cases of Python

Domain	Applications	Popular Libraries
Data Science	Data analysis, charts, forecasting	pandas, matplotlib, seaborn, numpy
Machine Learning / AI	Model training, language processing, deep learning	scikit-learn, TensorFlow, PyTorch, transformers
Web Development	Server logic, REST APIs, dashboards	Flask, Django, FastAPI
Automation / Scripting	Task automation, system work, data scraping	os, shutil, selenium, beautifulsoup4
DevOps & SRE	Deploy scripts, monitoring, CI/CD	fabric, Ansible, invoke
Cybersecurity	Pen testing, log checking, threat detection	scapy, shodan, requests, socket
Game Development	Prototype games, backend game logic	pygame, panda3d
Finance & FinTech	Trading, risk checks, fraud detection	pandas, quantlib, statsmodels
IoT & Robotics	Sensor data, control systems	RPi.GPIO, OpenCV, pySerial

Companies Using Python & Their Use Cases

Company	Use Case	Python Role
Google	Backend services, AI research, tools	TensorFlow (made by Google)
Netflix	Recommendations, data analytics, automation	Data pipelines, ML models
Spotify	Music suggestions, backend APIs	Data science, web backend
Facebook (Meta)	Infrastructure, AI research	PyTorch (made by Facebook)
Instagram	Web backend (originally Django)	API server, content flow
Dropbox	Desktop app, cloud sync	Cross-platform apps
NASA	Calculations, data analysis	Mission data and analysis
Reddit	Community website backend	Web backend (Python + Flask)
Quora	Content ranking, web backend	Backend logic and speed
Uber	Demand forecasts, pricing	ML models, internal tools
Industrial Light & Magic	Visual effects	Image processing and automation

Why Companies Choose Python

- Fast to write and test new ideas
- Huge library support for all fields
- Open-source and community help
- Easy to read and maintain code
- Works well with C/C++, Java, cloud, and databases

⚙ Whether you want to build AI models, automate tasks, or make backend services — Python can help you do it easily.

> Python Unlocks Potential — No Matter Your Field, There's a Script for It.

Job Scope of Python and Cross Domain Opportunities

Python is more than just a programming language — it's a gateway to thriving tech and non-tech careers across industries. Its simplicity and power make it a perfect tool for professionals from **any background**, including **biotechnology**, **mechanical engineering**, **finance**, and more.

Why Python Has Massive Career Scope

- Used across industries: tech, healthcare, energy, education, finance, and more
- Demand for Python developers continues to rise due to AI, ML, and automation
- □ Easy to learn, powerful to build — ideal for both beginners and professionals
- Open-source and community-supported with vast library ecosystem

---

Popular Career Roles with Python

Role	Where Python Fits
Data Scientist / Analyst	Data cleaning, analysis, machine learning
AI/ML Engineer	Model development and deployment
Web Developer	Backend logic and REST APIs
DevOps Engineer	Automation scripts, CI/CD pipelines
Cybersecurity Analyst	Log analysis, threat detection, pen testing
Bioinformatician	DNA sequence analysis, data visualization
Mechanical Simulation Engineer	Automation, simulation control, visualization
Financial Analyst	Risk modeling, algorithmic trading
Educator / Researcher	Teaching, simulations, data processing
IoT Developer	Device programming and data transmission

Python for Other Domain Experts

Biotechnology & Bioinformatics

- Analyzing gene expression data using `Biopython`, `pandas`, `scikit-learn`
- Simulating protein interactions and biological models
- Drug discovery pipelines and visualizations

□ Mechanical & Civil Engineering

- Automating CAD model tasks using Python APIs
- Simulation & control using Python with tools like `Matplotlib`, `SimPy`, `PyBullet`
- Data logging from sensors and real-time visualization

Finance & Economics

- Building financial dashboards and trade bots
- Risk assessment and predictive modeling
- Using libraries like `QuantLib`, `pandas-datareader`, `yfinance`

Education & Research

- Creating interactive simulations
- Statistical computing and automation of grading tools
- Building educational apps with `Tkinter`, `Streamlit`

⚙ Manufacturing & Robotics

- Automation scripts for industrial processes
- Robotics using `ROS`, `OpenCV`, and Python-based microcontrollers
- Integrating sensors with Python (e.g., Raspberry Pi + Python)

Final Thoughts

Python acts as a **bridge** between domain expertise and modern computing:

> “Domain knowledge + Python = Career multiplier”

You don’t need to become a full-stack developer to use Python effectively. You just need to apply Python to **your own field** to solve real problems faster and smarter.

Summary

Connect @
Mail (Trisha Dhiman): dhimantrisha1812@gmail.com
Contact: +91-9729832340

Trisha Dhiman
AI/ML Enthusiast | Intern @CodroidHub | First-Year Engineering Student

Trisha Dhiman