Here's a comparison table highlighting the **advantages** and **disadvantages** of Python Notebook (`.ipynb`) and regular Python scripts (`.py`):

Feature	Python Notebook (`.ipynb`)	Python Script (`.py`)
Format	JSON-based interactive documents.	Plain text files containing Python code.
Execution	Code is executed in cells; allows step-by-step execution.	Runs as a whole script from start to finish.
Ease of Debugging	Easy to debug small sections of code by running cells individually.	Must rerun the entire script unless breakpoint debugging is used.
Visualization	Integrates directly with rich output (e.g., Matplotlib, Seaborn, Pandas plots) displayed inline.	Visualizations open in a separate window; less seamless.
Interactivity	Excellent for interactive coding and exploration, often used in data analysis and machine learning.	Limited interactivity; best for standalone applications or reusable scripts.
Documentation	Supports Markdown and inline text, making it great for tutorials, reports, and documentation.	Comments can be added, but no Markdown or visual formatting is supported.
Dependencies	Requires a Jupyter environment (e.g., JupyterLab or Jupyter Notebook).	Runs with a standard Python interpreter; no extra setup needed.
Collaboration	Collaboration is harder due to `.ipynb` file's JSON format; tools like JupyterHub/Git may help.	Easier collaboration through version control tools like Git (text diffs).
Execution Context	Keeps variables and states in memory between cells, which can lead to unintended dependencies between cells.	Fresh execution context each time the script is run, minimizing unintended variable reuse.
Performance	Slower startup due to notebook server initialization; not ideal for production.	Faster execution and better for production.

File Size	Can become bulky due to JSON structure and embedded outputs (e.g., images).	Lightweight text files.
Best Use Cases	Exploratory data analysis, teaching, sharing visual results.	Writing reusable modules, deploying applications, and running production workflows.
Code Reusability	Harder to reuse; typically designed for one-off tasks or iterative development.	Designed for modularity; easy to import and reuse functions or classes in other scripts.

When to Use Each

- **Use `.ipynb`**:
    - When working on **data analysis**, **machine learning**, or **teaching programming**.
    - If you need **inline visualization**, **documentation**, and interactivity.
- **Use `.py`**:
    - For creating **production-ready scripts** or **modular libraries**.
    - When working in environments with version control, continuous integration, or automation.