GETTING STARTED WITH JUPYTER NOTEBOOKS

A Quick-Start Guide for Your First Code Execution

An **Interactive Python Notebook (`.ipynb`)** is a powerful environment that combines live executable code, computational math equations, rich explanatory visualizations, and formatted text blocks into a single interactive document.

1. Environment Setup

Before executing code, ensure your notebook environment is ready:

1. Open your preferred data science platform (*JupyterLab, VS Code, Google Colab, or Anaconda*).
2. Create a brand new notebook document and name it `my_first_notebook.ipynb`.

2. Anatomy of a Notebook: The Component Cells

Jupyter Notebooks are constructed out of discrete modular blocks called **Cells**. Cells primarily function in one of two states: **Markdown Cells** (for formatted text documentation) and **Code Cells** (for live Python execution).

1 Cell 1: Documenting with Markdown

Instructions: Click inside your first cell, change the drop-down selector in your top toolbar from **Code** to **Markdown**, paste the content below, then press `Shift` + `Enter` to build it.

```
# My First Jupyter Notebook

Welcome to your first execution! This is a **Markdown cell**, which means it's used
for writing text, documentation, and notes rather than running code.

### What we will do next:
1. Define some variables.
2. Run a simple math calculation.
3. Print a success message.
```

Rendered Output Preview:

> ## My First Jupyter Notebook
>
> Welcome to your first execution! This is a **Markdown cell**, which means it's used for writing text, documentation, and notes rather than running code.
>
> ### What we will do next:
>
> 1. Define some variables.
> 2. Run a simple math calculation.
> 3. Print a success message.

2 Cell 2: Variable Definition & Assignment (Code)

Instructions: Create a new cell (defaults to Code), paste the following Python script, and press `Shift` + `Enter` to save variables to active memory.

```
# This is a Code cell. Lines starting with '#' are comments.

# 1. Let's define some variables
user_name = "Explorer"
lucky_number = 7

# 2. Let's do a quick calculation
result = lucky_number * 10

print("Variables successfully saved to memory!")
```

3 Cell 3: Persistent Memory & Evaluation (Code)

Instructions: Create another Code cell. This demonstrates Jupyter's persistent state: it remembers active variables across isolated code snippets. Paste and run using `Shift` + `Enter`.

```
# Because you ran the cell above, Jupyter remembers 'user_name' and 'result'
print(f"Hello, {user_name}!")
print(f"Your lucky number multiplied by 10 is: {result}")
print("🚀 You have successfully executed your first notebook!")
```

3. Essential Keyboard Shortcuts

Mastering basic shortcuts significantly enhances coding and iteration speed.

`Shift + Enter`	Runs the currently selected cell and advances focus to the next cell.
`Ctrl + Enter`	Runs the currently selected cell and stays in place.
`Esc → A`	Enters Command Mode and inserts a empty new cell **Above**.
`Esc → B`	Enters Command Mode and inserts a empty new cell **Below**.
`Esc → M`	Converts the currently active cell type to a **Markdown** text cell.
`Esc → Y`	Converts the currently active cell type back to a **Code** cell.

> 💡 **Important Execution Tip: Understanding Execution Indicators**
>
> Pay attention to the label box to the left of your code cells (e.g., `In [ ]`):
>
> - `In [*]` : The asterisk indicates that your Python cell is **actively processing** the execution.
> - `In [1]` : A clear integer represents a finished run, showing the order queue of execution history.