Conda & Jupyter Notebook – Proper Structured Notes

These notes are a **clean, exam-ready + practical reference** for managing **Conda environments and Jupyter kernels** correctly. They are based on real debugging experience and best practices.

---

1. Basic Diagnostic Commands (First Check)

Run these in **Command Prompt / Anaconda Prompt**:

```
python --version
jupyter --version
jupyter kernelspec list
```

Purpose: - Verify Python is installed - Verify Jupyter exists - Verify available kernels

---

2. Updating Conda Safely (Base Environment Only)

```
conda update conda -y
conda update anaconda -y
```

✔️Always run these in `base` ❌Never update conda inside project environments

---

3. What Is a Conda Environment?

A **conda environment** is an isolated Python workspace.

Each environment has: - Its own Python version - Its own libraries - Its own Jupyter kernel (if registered)

🔝Goal: **Prevent dependency conflicts and kernel crashes**

---

4. Why the Kernel Was Getting Jammed

Root Causes:

- Mixing `pip install` with `conda install` in `base`
- Installing unstable libraries (e.g. `pandas-visual-analysis`, `anywidget`)
- Broken kernel path
- Corrupted base environment

Symptoms:

- Jupyter opens
- Kernel never executes
- No visible error (silent failure)

---

5. The Correct Fix Strategy

Instead of repairing a corrupted base environment:

1. Create a **new clean environment**
2. Install only required libraries
3. Register a fresh Jupyter kernel

This **bypasses corruption completely**.

---

6. Creating a New Environment

```
conda create -n eda python=3.10 -y
```

Activate it:

```
conda activate eda
```

---

7. Installing Packages (Correct Method)

Always activate the environment first:

```
conda activate eda
```

Then install required packages:

```
conda install jupyter ipykernel numpy pandas matplotlib seaborn scikit-learn
-y
```

⚠️Never install packages without activating the environment.

---

8. Jupyter Is Environment-Specific

Each environment is **independent**.

So: - Jupyter in `base` ≠ Jupyter in `eda`

Error:

```
'jupyter' is not recognized
```

Meaning: ➡️Jupyter is not installed in that environment.

---

9. Registering Environment as a Jupyter Kernel

This step connects **Jupyter ↔ Conda environment**.

```
python -m ipykernel install --user --name eda --display-name "Python (eda)"
```

Now `eda` appears as a selectable kernel inside Jupyter.

---

10. Starting Jupyter (Best Practice)

```
conda activate eda
jupyter notebook
```

Inside notebook:

```
Kernel → Change Kernel → Python (eda)
```

The notebook remembers this kernel.

---

11. Verifying the Active Environment (Inside Notebook)

```
import sys
print(sys.executable)
```

Correct output:

```
.../envs/eda/python.exe
```

---

12. Stopping Jupyter Notebook (Correct Way)

From the same terminal:

```
CTRL + C
```

Then:

```
y
```

This is a **clean and safe shutdown**.

---

13. Understanding Jupyter Logs

Example:

```
Kernel started
Interrupted...
Parent appears to have exited
```

Meaning: - Kernel started correctly - User pressed Ctrl+C - Kernel shut down normally

⚠️ `WARNING` does NOT mean failure

---

14. Removing (Unregistering) a Jupyter Kernel

Sometimes an environment is deleted, but its **kernel still appears in Jupyter**. This kernel must be removed manually.

⚠️Always deactivate the environment first.

Step 1: Deactivate environment

```
conda deactivate
```

Step 2: List all registered kernels

```
jupyter kernelspec list
```

You will see output like:

```
python3
eda
clean_ds
```

Step 3: Remove the kernel

```
jupyter kernelspec remove clean_ds
```

Type `y` and press Enter to confirm.

✅Kernel is now completely removed from Jupyter.

---

15. Renaming a Conda Environment

Conda **cannot rename environments** directly.

Correct method: 1. Create new environment with desired name 2. Install packages 3. Register kernel 4. Remove old environment

Example (remove old env):

```
conda remove -n clean_ds --all -y
```

---

16. Listing Conda Environments

```
conda env list
```

---

16. Daily Recommended Workflow

1. Open Anaconda Prompt
2. Activate environment

   ```
   conda activate eda
   ```

3. Start Jupyter

   ```
   jupyter notebook
   ```

4. Work normally

5. Stop Jupyter (Ctrl + C → y)
6. Deactivate environment

```
conda deactivate
```

---

17. Golden Rules (Must Remember)

✔️One project = one environment
✔️Prefer `conda install` over `pip`
✔️Register kernel once per environment
✔️Backup notebooks regularly

❌Do not experiment in `base`
❌Do not install random visualization libraries
❌Do not mix environments

---

18. Interview-Ready Answer

Q: How do you manage Python environments?

A:

> I use conda environments to isolate dependencies. Each project has its own environment and Jupyter kernel to avoid version conflicts and ensure reproducibility.

---

19. Final Mindset

Breaking environments means **you are learning deeply**.

Environments can always be rebuilt. Your **knowledge, logic, and code are permanent**.

✅You now understand Conda + Jupyter at a professional level.