GeoBig data— Four‑Step Hands‑On Lab

*Goal: A minimal, reproducible workflow that sets up a Python environment, runs a Jupyter Notebook (in VS Code and in the browser), and practices essential Git commands.*

## Prerequisites

* Operating system with admin rights (Windows/macOS/Linux)
* Internet access to download installers
* Basic terminal access (Anaconda Prompt or Terminal)

## Step 1 — Install Anaconda/Miniconda and create the course environment

*Outcome: a clean environment and a registered Jupyter kernel.*

# Create & activate environment

conda create -n geopy -c conda-forge python=3.10 geopandas shapely pyproj matplotlib jupyterlab jupyterlab-git notebook ipykernel

**Quick checks:**

python -V

jupyter --version

## Step 2 — Install & configure VS Code; create/test a Jupyter Notebook (VS Code + Browser)

*Outcome: Notebook runs in VS Code and in JupyterLab (browser).*

* Install VS Code and extensions: Python and Jupyter (Microsoft).
* Select interpreter: Command Palette → Python: Select Interpreter → Python (geopy).
* Create a working folder and a test notebook:

**In the notebook, run:**

import sys, geopandas as gpd, matplotlib

print(sys.executable)

print(gpd.\_\_version\_\_, matplotlib.\_\_version\_\_)

**Browser mode (JupyterLab):**

# With the 'geopy' environment active

jupyter lab

# Create 01\_test.ipynb notebook and run the same test cell

## Step 3 — Local Git practice: add / commit / branch / checkout / restore / reset

*Outcome: Understand the basic lifecycle and how to undo changes safely.*

**A) Initialize and first commit**

cd ~/geopy-lab (or git clone git@github.com:geodynamic-lab/geoBigData2025Q3.git)

git init

echo "# StructGeo Lab" > README.md

git status

git add README.md

git commit -m "chore: initial commit"

git branch -M main

**B) Feature branch: edit and commit**

git checkout -b feat/notebook-demo

# Edit 01\_test.ipynb (add a cell), save:

git add 01\_test.ipynb

git commit -m "feat: add demo cell in notebook"

**C) Merge back to main**

git checkout main

git merge feat/notebook-demo

git log --oneline --graph

**D) git restore vs git reset (scoped experiment)**

1) Unstage a change (keep your file edits):

echo "One more line" >> README.md

git add README.md

git restore --staged README.md

2) Discard working tree edits (careful):

git restore README.md

3) Move history pointer and optionally reset index/worktree:

git reset --soft HEAD~1

git reset --mixed HEAD~1

git reset --hard HEAD~1

**Rules of thumb**

* `restore`: file-focused; manipulates working tree / index; does not rewrite history.
* `reset`: history-focused; moves HEAD and resets index (and possibly worktree).
* `checkout/branch`: switch/create branches; `checkout` can also restore a file to a commit.
* `commit`: snapshot the index into history.

## Step 4 — Archive results for reproducibility

*Outcome: Others can recreate your environment and re-run your steps.*

# Export concise environment spec (recommended)

conda env export --name structgeo --from-history > environment.yml

# Or fully pinned (explicit) spec

conda list --explicit > spec.txt

git add environment.yml spec.txt

git commit -m "chore: add reproducible env files"

git tag -a v0.1 -m "structgeo lab v0.1"

**(Optional) push to a private GitHub repository:**

git remote add origin https://github.com/<you>/geoBigData2025Q3.git

git push -u origin main --tags

### Submission checklist

* `01\_test.ipynb` saved with outputs (screenshot acceptable).
* `git log --oneline` screenshot with ≥ 2 meaningful commits.
* `environment.yml` (or `spec.txt`) included.