Collaborating on Research Code – Session 1

Hands-on with Git, GitHub, and VS Code

This document guides you step by step through the first hands-on session. You can follow all steps using either:

- the **terminal** (Git Bash on Windows, Terminal on macOS/Linux), or
- **VS Code**'s integrated Git tools (Source Control panel).

Both paths use the same Git concepts; only the user interface differs.

1 Pre-flight checklist

Before starting the exercises, make sure the following items are ready.

1.1 Tools installed

1. **Python 3.10+**
   - Check in a terminal:

     ```
     python --version
     ```

     or, on some systems:

     ```
     python3 --version
     ```

2. **Git**
   - Check:

     ```
     git --version
     ```

3. **Visual Studio Code (VS Code)** (optional, but recommended)
   - Open it once to ensure it runs properly.

If something is missing, install it now or pair with someone who has a complete setup.

1.2 Configure your Git identity

In a terminal (Git Bash on Windows, Terminal on macOS/Linux), set your name and email:

```
git config --global user.name "Your Name"
git config --global user.email "your.email@university.edu"
```

You only need to do this once per machine.

2 Choose your starting point

You can choose to start on one of the following **paths**:

Path A – New repository from scratch You create a very small Python project and initialize a new Git repository yourself.

Path B – Clone an example repository You clone a small example repository provided by the instructor and then modify it.

You can also work in pairs if that helps.

3 Path A – create a repository from scratch

3.1 Create the project folder

1. Choose a location (e.g. Desktop or your home directory).
2. Create a folder, e.g. `git-workshop-01`.
3. Open **VS Code** and select `File` → `Open Folder...`, then choose `git-workshop-01`.
4. Inside VS Code, create a new file `hello_world.py` with the following content:

```
print("Hello, world!")
```

3.2 Initialize Git

You may use either the **terminal** or the **VS Code GUI**.

Option 1: Terminal

1. In VS Code, open a terminal: `Terminal` → `New Terminal`. It should start in the `git-workshop-01` folder.
2. Run:

```
git init
git status
```

You should see `hello_world.py` listed as an *untracked* file.

Option 2: VS Code Source Control Panel

1. Click the **Source Control** icon on the left (the branch-like icon).
2. Click **"Initialize Repository"**.
3. You should now see `hello_world.py` under *Changes*.

3.3 First commit

Terminal

1. Stage and commit:

```
git add hello_world.py
git commit -m "feat: add first hello world script"
git log --oneline --decorate --graph
```

VS Code

1. In the **Source Control** panel:
   - Hover over `hello_world.py` under *Changes* and click the + to stage it.
   - In the message box at the top, type:
     `feat: add first hello world script`
   - Click the **Commit** icon (tick mark) or press `Ctrl+Enter`.
2. Optionally, check the status in a terminal:

   ```
   git status
   ```

   You should see: `nothing to commit, working tree clean`.

3.4 Modify the script and commit again

1. Modify `hello_world.py` to:

   ```
   print("Hello, world!")
   print("The world ignores you, but the code runs.")
   ```

2. Commit the change:

   Terminal:
   ```
   git status
   git add hello_world.py
   git commit -m "feat: add second line to hello world"
   git log --oneline --decorate --graph
   ```
   VS Code:
   - In Source Control, stage the modified file.
   - Commit with a message such as
     `feat: add second line to hello world`.

Checkpoint A: You should now have **two commits** in your local history.

4 Path B – Clone an example repository

The code can be found in `https://github.com/mandli/RESCUER_workshop.git`

4.1 Clone the repository

Option 1: terminal

1. In a terminal:

   ```
   git clone https://github.com/mandli/RESCUER_workshop.git
   cd RESCUER_workshop
   git status
   ```

Option 2: VS Code

1. Open VS Code.
2. Click **Source Control** → **"Clone Repository"**.
3. Paste the URL.
4. Choose a folder (e.g. Desktop).
5. When VS Code asks, open the cloned repository.

4.2 First local change and commit

Go to `Collaborative_Software_Development/hands-on/session1/code/` and pick a very simple change, for example:

- add a comment to `hello_world.py`, or
- create a new file `notes.txt` explaining what the script does.

Then commit your change:

Terminal:
```
git status
git add .
git commit -m "docs: add notes about example script"
git log --oneline --decorate --graph
```

VS Code:
- Stage the changed / new file(s) in the Source Control panel.
- Commit with a message such as `docs: add notes about example script`.

Checkpoint B: You should have at least **one new commit** on top of the cloned history.

5 Common step – create a feature branch

This part is the same whether you used Path A or Path B (or both). Make sure you are inside your repository folder.

5.1 Create a new branch

Choose a short feature name, e.g. `feature/print-name` with your name.

Terminal

```
git checkout -b feature/print-name
git branch
```

or

```
git branch feature/print-name
git checkout feature/print-name
git branch
```

You should see `feature/print-name` with a star (⋆) next to it.

VS Code

1. Click the branch name in the bottom-left status bar (it probably shows `main` or `master`).
2. Select **"Create new branch. . ."**.
3. Enter `feature/print-name` as the name.
4. Confirm that the status bar now shows `feature/print-name`.

5.2 Implement a small feature

Examples:

- Extend `hello_world.py`:

  ```
  name = input("What is your name? ")
  print(f"Hello, {name}!")
  ```

- Or create a second script `run_experiment.py` that prints:

  ```
  print("Experiment started.")
  ```

After editing, commit your work:

Terminal:
```
git status
git add .
git commit -m "feat: ask for user name in greeting"
```

VS Code:
- Stage the modified / new files.
- Commit with a descriptive message, e.g. `feat: ask for user name in greeting`.

You can create **one or two small commits**, each doing one logical change.

6 Optional – connect to GitHub and push

This part is optional and depends on whether you have a GitHub account.

6.1 Create a GitHub repository

On GitHub:

1. Click **"New"** to create a repository.
2. Use the same name as your local folder, e.g. `git-workshop-01`.
3. **Do not** initialize the repository with a README or license (your local repo already has commits).
4. Click **"Create repository"**.

GitHub will show instructions for pushing an existing repository.

6.2 Add the remote and push your branches

In your local repository folder:

```
git remote add origin https://github.com/<your-username>/git-workshop-01.git
git branch -M main   # optional, if your main branch is still called 'master'
git push -u origin main
git push -u origin feature/print-name
```

Follow any authentication prompts. Git may open a browser window for you to log in.

VS Code Alternative

After adding the remote once, you can also use *"Publish Branch"* or *"Sync Changes"* buttons in the Source Control panel to push.

6.3 Optional – open a pull request

On GitHub:

1. Go to your repository page.
2. You should see a suggestion to **"Pull requests"** from `feature/print-name`.
3. Click on **"New pull request"**, then:
   - Compare the different branches you want to merge
   - Title: `feat: ask for user name in greeting`
   - Short description: explain what changed and why.
4. Click **"Create pull request"**.

7 Wrap-up

By the end of this session you should have:

- A local Git repository with **multiple commits**.
- At least one **feature branch** (`feature/...`) with one or more commits.
- Optionally, a **GitHub repository** with:
  - `main` pushed, and
  - your feature branch pushed,
  - possibly a pull request opened.

In the next session we will use a more realistic 2D diffusion / transport code base, work in pairs, and practice branching, pull requests, code review, conflict resolution, and basic testing/CI in a research context.