Lab 11

Exercise 1 — Creating and Tracking a Project

Create a new project and track it with Git.

```
mkdir math-project
cd math-project
git init
```

Create a file `norm.py`:

```
def compute_norm(v):
    return sum(x*x for x in v)**0.5
```

Then run:

```
git status
git add norm.py
git commit -m "Add norm function"
```

Tasks

1. What does `git status` show before and after `git add`?
2. Why is committing better than manually copying files such as `final_v2.py`?

Exercise 2 — Branching and Merging

Create a new branch:

```
git switch -c experiment
```

Modify `norm.py`:

```
def compute_norm(v):
    return round(sum(x*x for x in v)**0.5, 2)
```

Commit the change.
Switch back to `main` and modify the same function differently:

```
def compute_norm(v):
    return sum(x*x for x in v)**0.5
```

Commit again, then merge:

```
git merge experiment
```

Tasks

1. Did a merge conflict occur?
2. Explain the meaning of:
   - `<<<<<<< HEAD`
   - `=======`
   - `>>>>>>> experiment`
3. Resolve the conflict and complete the merge.

Exercise 3 — Testing and Refactoring

Create a file `test_norm.py`:

```
from norm import compute_norm

def test_compute_norm():
    assert compute_norm([3,4]) == 5
```

Now extend `norm.py`:

```
def normalize(v):
    n = compute_norm(v)
    return [x/n for x in v]
```

Tasks

1. Why is it better to reuse `compute_norm` instead of rewriting the formula?
2. Intentionally break `compute_norm`. Does the test detect the error?
3. Why are tests important in scientific computing?

Exercise 4 — Reproducible Environments

Create a virtual environment:

```
python -m venv env
source env/bin/activate
pip install numpy
```

Tasks

1. What problem do virtual environments solve?
2. What could happen if two collaborators use different library versions?
3. How does this relate to reproducibility in research?