Practice Project

Scenario

You are creating a small Python project with these files:

- `main.py`
- `utils/math_utils.py`
- `README.md`

Tasks

1. **Setup**

   a. Initialize a new Git repo.

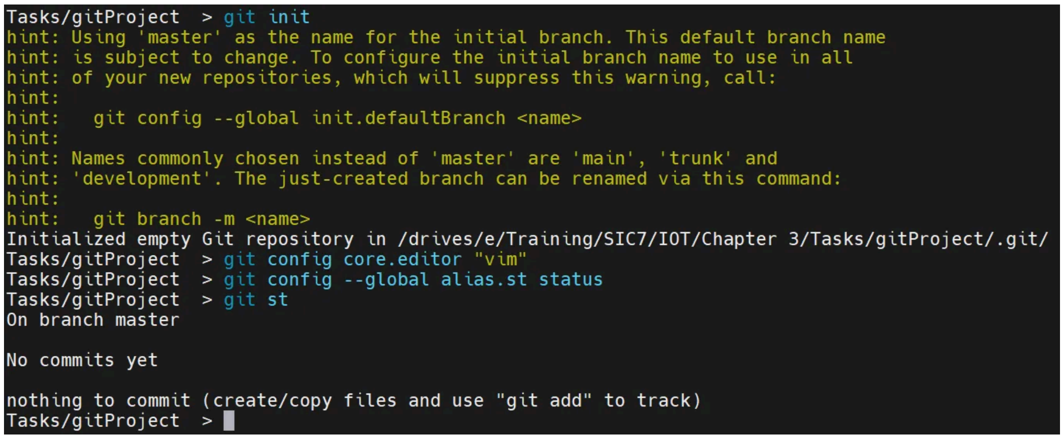
   ```
   git init
   ```

   b. Configure your default editor (pick nano, vim, or code --wait).

   ```
   git config core.editor "vim"
   ```

   c. Add an alias so you can type `st` instead of the full status command.

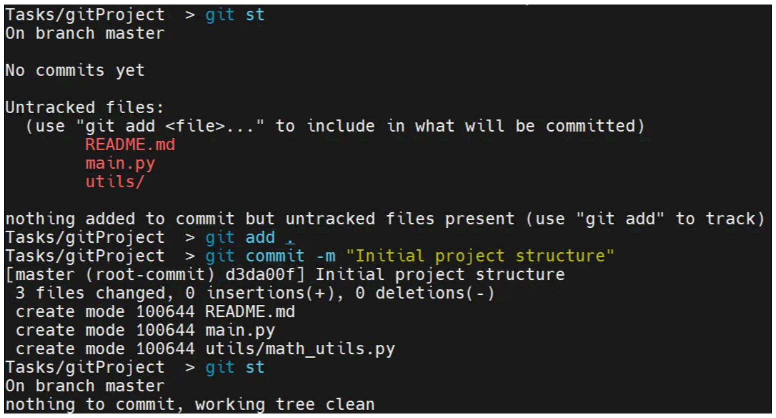
   ```
   git config --global alias.st status
   ```



```
Tasks/gitProject  > git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /drives/e/Training/SIC7/IOT/Chapter 3/Tasks/gitProject/.git/
Tasks/gitProject  > git config core.editor "vim"
Tasks/gitProject  > git config --global alias.st status
Tasks/gitProject  > git st
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
Tasks/gitProject  >
```

2. **First Commit**
   a. Add all project files and make your first commit: “Initial project structure”.

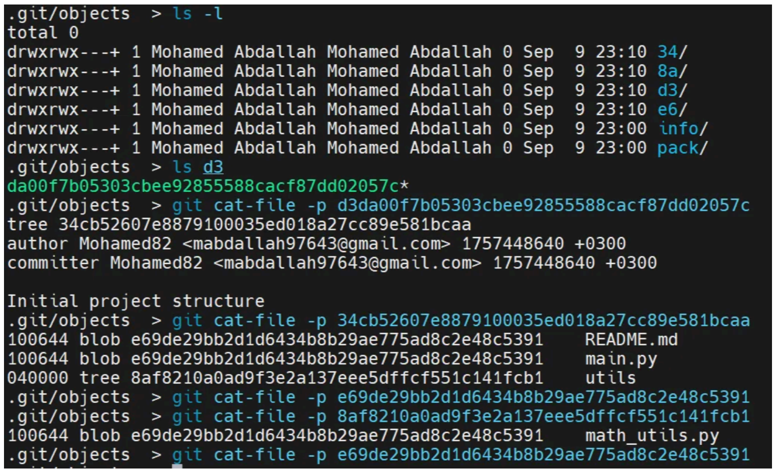
```
git add .
git commit -m "Initial project structure"
```



```
Tasks/gitProject  > git st
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md
        main.py
        utils/

nothing added to commit but untracked files present (use "git add" to track)
Tasks/gitProject  > git add .
Tasks/gitProject  > git commit -m "Initial project structure"
[master (root-commit) d3da00f] Initial project structure
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
 create mode 100644 main.py
 create mode 100644 utils/math_utils.py
Tasks/gitProject  > git st
On branch master
nothing to commit, working tree clean
```

   b. Explore the `.git/objects/` directory. (Hint: use a Git plumbing command to read the content of a blob or tree (cat-file)).

```
cd .git/objects
git cat-file <hashed-number>
```



```
.git/objects  > ls -l
total 0
drwxrwx---+ 1 Mohamed Abdallah Mohamed Abdallah 0 Sep  9 23:10 34/
drwxrwx---+ 1 Mohamed Abdallah Mohamed Abdallah 0 Sep  9 23:10 8a/
drwxrwx---+ 1 Mohamed Abdallah Mohamed Abdallah 0 Sep  9 23:10 d3/
drwxrwx---+ 1 Mohamed Abdallah Mohamed Abdallah 0 Sep  9 23:10 e6/
drwxrwx---+ 1 Mohamed Abdallah Mohamed Abdallah 0 Sep  9 23:00 info/
drwxrwx---+ 1 Mohamed Abdallah Mohamed Abdallah 0 Sep  9 23:00 pack/
.git/objects  > ls d3
da00f7b05303cbee92855588cacf87dd02057c*
.git/objects  > git cat-file -p d3da00f7b05303cbee92855588cacf87dd02057c
tree 34cb52607e8879100035ed018a27cc89e581bcaa
author Mohamed82 <mabdallah97643@gmail.com> 1757448640 +0300
committer Mohamed82 <mabdallah97643@gmail.com> 1757448640 +0300

Initial project structure
.git/objects  > git cat-file -p 34cb52607e8879100035ed018a27cc89e581bcaa
100644 blob e69de29bb2d1d6434b8b29ae775ad8c2e48c5391    README.md
100644 blob e69de29bb2d1d6434b8b29ae775ad8c2e48c5391    main.py
040000 tree 8af8210a0ad9f3e2a137eee5dffcf551c141fcb1    utils
.git/objects  > git cat-file -p e69de29bb2d1d6434b8b29ae775ad8c2e48c5391
.git/objects  > git cat-file -p 8af8210a0ad9f3e2a137eee5dffcf551c141fcb1
100644 blob e69de29bb2d1d6434b8b29ae775ad8c2e48c5391    math_utils.py
.git/objects  > git cat-file -p e69de29bb2d1d6434b8b29ae775ad8c2e48c5391
```

No output from **blob**, because files are empty

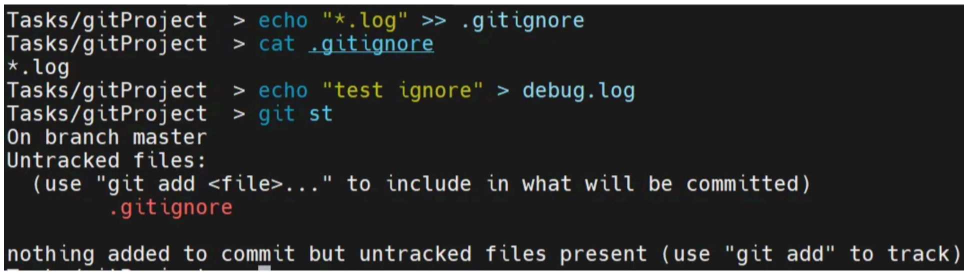
3. **Ignore Files**

   a. Create a rule to ignore all log files.

   ```
   echo "*.log" >> .gitignore
   ```

   b. Test by creating a debug.log file and check that Git ignores it.

   ```
   echo "test ignore" > debug.log
   git st
   ```

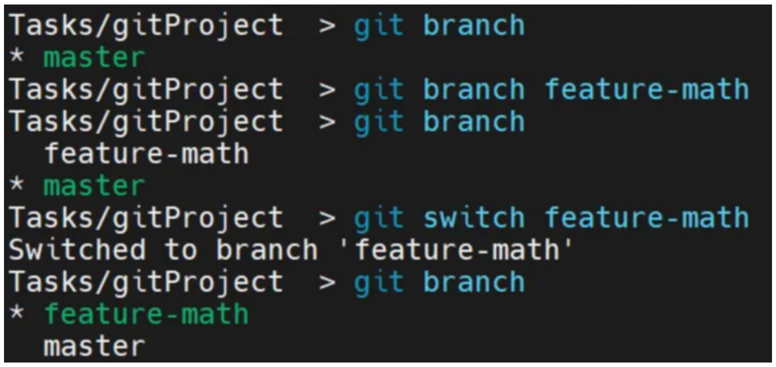


```
Tasks/gitProject  > echo "*.log" >> .gitignore
Tasks/gitProject  > cat .gitignore
*.log
Tasks/gitProject  > echo "test ignore" > debug.log
Tasks/gitProject  > git st
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
```

4. **New Feature (Branching)**

   a. Create a new branch called `feature-math`.

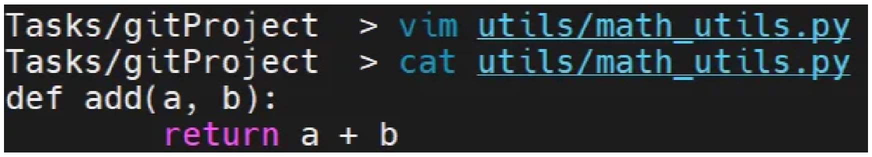
   ```
   git branch feature-math
   ```



```
Tasks/gitProject  > git branch
* master
Tasks/gitProject  > git branch feature-math
Tasks/gitProject  > git branch
  feature-math
* master
Tasks/gitProject  > git switch feature-math
Switched to branch 'feature-math'
Tasks/gitProject  > git branch
* feature-math
  master
```

b. Inside `utils/math_utils.py`, add a function:

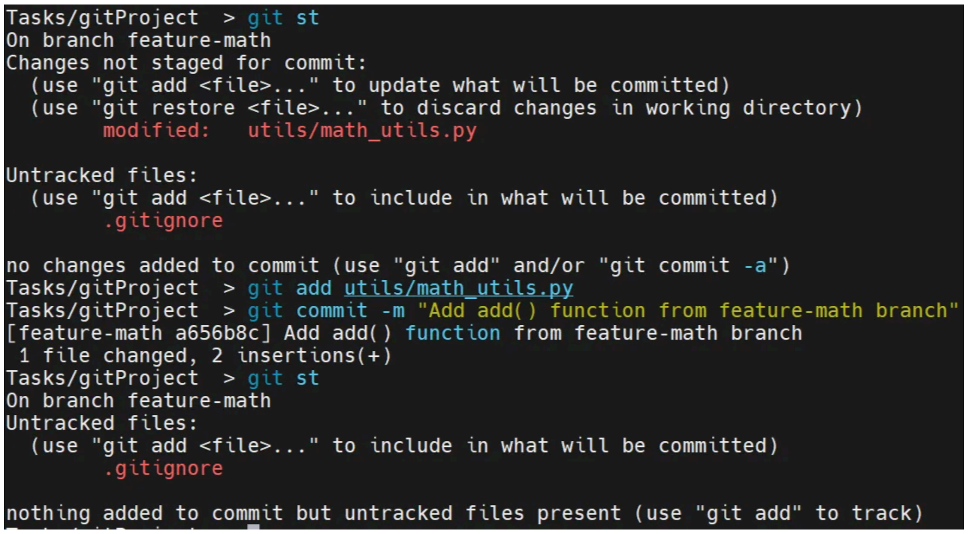
```
def add(a, b):
    return a + b
```



```
Tasks/gitProject  > vim utils/math_utils.py
Tasks/gitProject  > cat utils/math_utils.py
def add(a, b):
        return a + b
```

c. Commit this change to the branch.

```
git add utils/math_utils.py
git commit -m "Add add() function from feature-math branch"
```



```
Tasks/gitProject  > git st
On branch feature-math
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   utils/math_utils.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

no changes added to commit (use "git add" and/or "git commit -a")
Tasks/gitProject  > git add utils/math_utils.py
Tasks/gitProject  > git commit -m "Add add() function from feature-math branch"
[feature-math a656b8c] Add add() function from feature-math branch
 1 file changed, 2 insertions(+)
Tasks/gitProject  > git st
On branch feature-math
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
```

---

5. **Merging**

a. Switch back to main.

```
git switch master
```

b. Merge the branch into main.

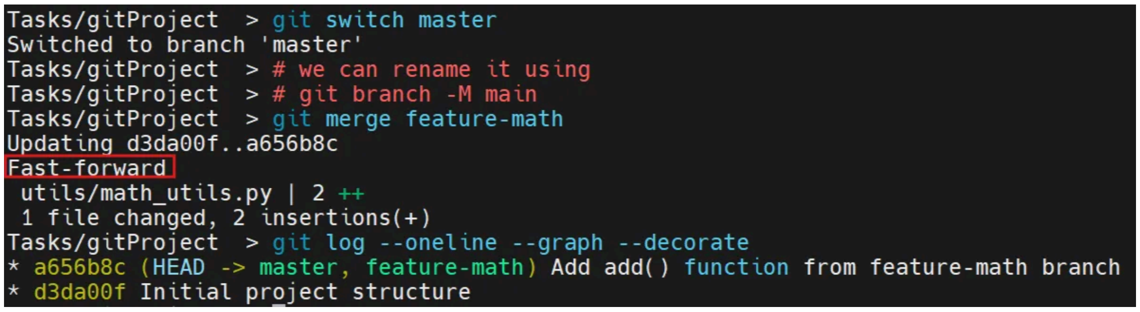
```
git merge feature-math
```

c. Check if the merge was fast-forward or a 3-way merge and what is the difference between the two ways and show your answer using a diagram or using (log command ← bonus) ?

   i. It is fast-forward.

      - **Fast-forward merge**: straight line history.
      - **3-way merge**: branch divergence and merge commit.

   ii. `git log --oneline --graph --decorate`



```
Tasks/gitProject  > git switch master
Switched to branch 'master'
Tasks/gitProject  > # we can rename it using
Tasks/gitProject  > # git branch -M main
Tasks/gitProject  > git merge feature-math
Updating d3da00f..a656b8c
Fast-forward
 utils/math_utils.py | 2 ++
 1 file changed, 2 insertions(+)
Tasks/gitProject  > git log --oneline --graph --decorate
* a656b8c (HEAD -> master, feature-math) Add add() function from feature-math branch
* d3da00f Initial project structure
```

---

6. **Undo / Unstage**

   a. Edit README.md (e.g., add "This is a math project") and stage it.

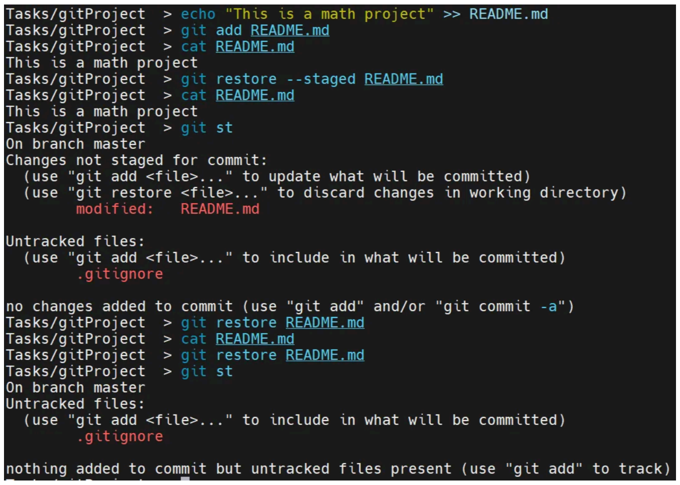
```
echo "This is a math project" >> README.md
git add README.md
```

   b. Oops! Unstage it without deleting your changes.

```
git restore --staged README.md
```

c. Then discard your changes completely.

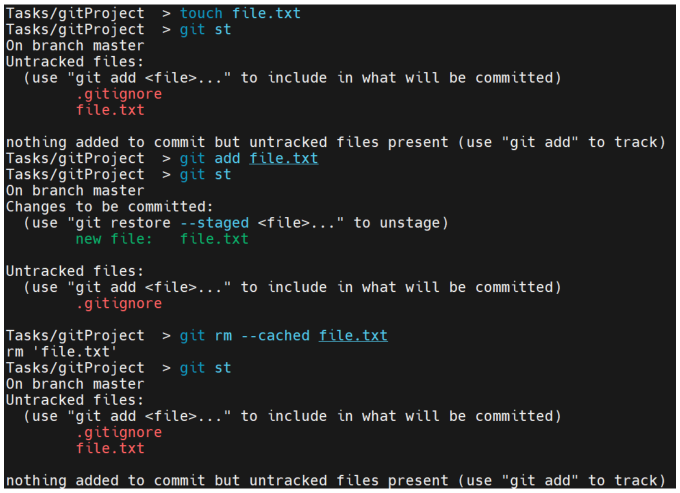
```
git restore --worktree README.md
```



```
Tasks/gitProject  > echo "This is a math project" >> README.md
Tasks/gitProject  > git add README.md
Tasks/gitProject  > cat README.md
This is a math project
Tasks/gitProject  > git restore --staged README.md
Tasks/gitProject  > cat README.md
This is a math project
Tasks/gitProject  > git st
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

no changes added to commit (use "git add" and/or "git commit -a")
Tasks/gitProject  > git restore README.md
Tasks/gitProject  > cat README.md
Tasks/gitProject  > git restore README.md
Tasks/gitProject  > git st
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
```

d. Explan for me what is the difference between restore `--staged` , `--worktree` and `rm --cached` ? And show your explanation in your terminal <3.

- `-staged` : unstage.
- `-worktree` : restore file contents to last commit.
- `rm --cached` : stop tracking file but keep it in your folder.



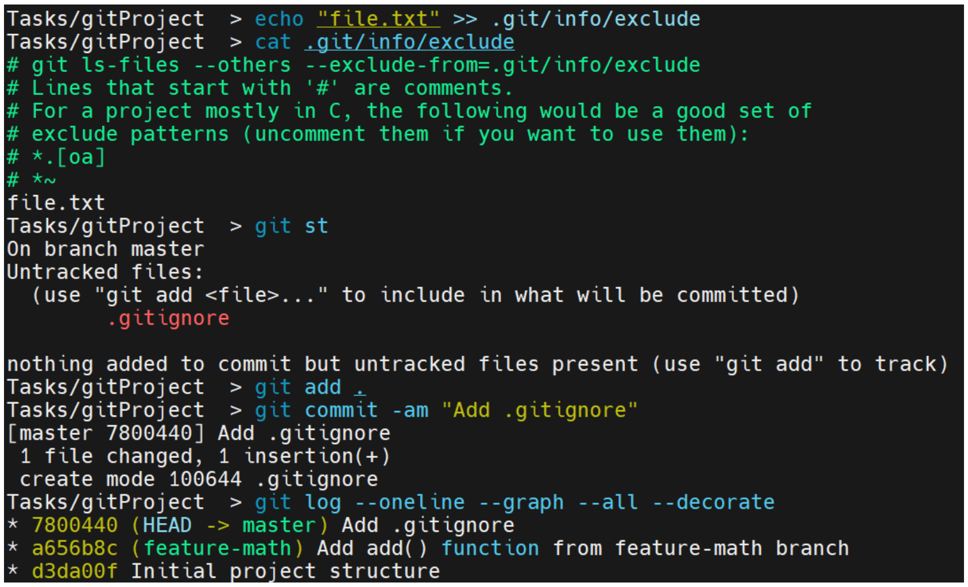
```
Tasks/gitProject  > touch file.txt
Tasks/gitProject  > git st
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        file.txt

nothing added to commit but untracked files present (use "git add" to track)
Tasks/gitProject  > git add file.txt
Tasks/gitProject  > git st
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

Tasks/gitProject  > git rm --cached file.txt
rm 'file.txt'
Tasks/gitProject  > git st
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        file.txt

nothing added to commit but untracked files present (use "git add" to track)
```

`—worktree` & `—staged` are tested before

7. **Bonus Challenge**

   a. Ignore a file using `.git/info/exclude` instead of `.gitignore.`

   b. Visualize the commit history as a graph (Hint: compact one-line graph view)



```
Tasks/gitProject  > echo "file.txt" >> .git/info/exclude
Tasks/gitProject  > cat .git/info/exclude
# git ls-files --others --exclude-from=.git/info/exclude
# Lines that start with '#' are comments.
# For a project mostly in C, the following would be a good set of
# exclude patterns (uncomment them if you want to use them):
# *.[oa]
# *~
file.txt
Tasks/gitProject  > git st
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore

nothing added to commit but untracked files present (use "git add" to track)
Tasks/gitProject  > git add .
Tasks/gitProject  > git commit -am "Add .gitignore"
[master 7800440] Add .gitignore
 1 file changed, 1 insertion(+)
 create mode 100644 .gitignore
Tasks/gitProject  > git log --oneline --graph --all --decorate
* 7800440 (HEAD -> master) Add .gitignore
* a656b8c (feature-math) Add add() function from feature-math branch
* d3da00f Initial project structure
```