EPFL, CIVIL-127

Programming and software development for engineers

👋 Hello

Questions, comments, feedback:

- Forum
  - You can post questions in English or French
- alok.menghrajani@epfl.ch or alexandre.alahi@epfl.ch
- Teaching Assistants
  - Valentin and Weijiang
- Student Assistants
  - Johanna and Mehdi

2D arrays: plain list

```python
WIDTH, HEIGHT = (5, 4)
grid = [0] * (WIDTH * HEIGHT)

def offset(x, y):
    if not 0 <= x < WIDTH:
        raise IndexError("invalid offset: ", x, y)
    if not 0 <= y < HEIGHT:
        raise IndexError("invalid offset: ", x, y)
    return x * HEIGHT + y

def get(x, y):
    return grid[offset(x, y)]

def set(x, y, v):
    grid[offset(x, y)] = v

def printGrid():
    for y in range(HEIGHT):
        for x in range(WIDTH):
            print(get(x, y), end=" ")
        print()

set(4, 3, "A")
set(1, 2, "B")
printGrid()
```

2D arrays: plain list

Result:

```
0 0 0 0 0
0 0 0 0 0
0 B 0 0 0
0 0 0 0 A
```

2D arrays: list of lists

```python
grid = [[0] * 4] * 5
print(grid)
```

Result:

```
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

2D arrays: list of lists

```python
WIDTH, HEIGHT = (5, 4)
grid = [[0] * WIDTH] * HEIGHT

def get(x, y):
    return grid[y][x]

def set(x, y, v):
    grid[y][x] = v

def printGrid():
    for y in range(HEIGHT):
        for x in range(WIDTH):
            print(get(x, y), end=" ")
        print()

set(4, 3, "A")
set(1, 2, "B")
printGrid()
```

Don't do this! ⚠️

2D arrays: list of lists

Result:

```
0 B 0 0 A
0 B 0 0 A
0 B 0 0 A
0 B 0 0 A
```

We have an incorrect behavior.

2D arrays: list of lists

```python
WIDTH, HEIGHT = (5, 4)
grid = [[0] * WIDTH for _ in range(HEIGHT)]

def get(x, y):
    return grid[y][x]

def set(x, y, v):
    grid[y][x] = v

def printGrid():
    for y in range(HEIGHT):
        for x in range(WIDTH):
            print(get(x, y), end=" ")
        print()

set(4, 3, "A")
set(1, 2, "B")
printGrid()
```

2D arrays: list of lists

Result:

```
0 0 0 0 0
0 0 0 0 0
0 B 0 0 0
0 0 0 0 A
```

2D arrays: dict

```python
WIDTH, HEIGHT = (5, 4)
grid = {}

def get(x, y):
    if (x, y) not in grid:
      return 0
    return grid[(x, y)]

def set(x, y, v):
    grid[(x, y)] = v

def printGrid():
    for y in range(HEIGHT):
        for x in range(WIDTH):
            print(get(x, y), end=" ")
        print()

set(4, 3, "A")
set(1, 2, "B")
printGrid()
```

2D arrays: dict

Result:

```
0 0 0 0 0
0 0 0 0 0
0 B 0 0 0
0 0 0 0 A
```

pip install <package>

- Do leverage existing libraries
- https://docs.python.org/3.12/library/index.html
  - pprint
  - itertools
  - functools
  - datetime
  - …
- colorful and colorama
- numpy
- scipy
- …

git

git ≠ GitHub

- git
  - open source
  - command line tool
  - 3rd party graphical interfaces
- GitHub
  - Owned by Microsoft
  - Builds on top of git
  - Lots of collaborative features (pull requests, issue tracking, forums, wiki, pages, CI/CD, etc.)

What is git?

- distributed version control system
- enables collaboration
- enables iterative changes and rollbacks
- resolves conflicts

Resources

- git man pages
  - man == manual. To learn about e.g. `git add`, type: `$ man git-add`
  - also available online, e.g. https://git-scm.com/docs/git-add
- Pro Git (free book)
- Lots of free resources and tutorials
  - Use your favorite search engine to find them

Creating a repo

```
$ mkdir my-first-git-repo
$ cd my-first-git-repo
[my-first-git-repo]$ git init
Initialized empty Git repository in /[...]/my-first-git-repo/.git/
```

working directory vs staging area vs tree

- working directory → files on your disk
- staging area → changes you want to include in your next commit
- tree → store of all the commits

An initial file

```
[my-first-git-repo]$ echo "Code flows like rivers of change," > haiku.txt
[my-first-git-repo]$ ls
haiku.txt
[my-first-git-repo]$ git status
On branch main

No commits yet

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

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

Staging

```
[my-first-git-repo]$ git add haiku.txt
[my-first-git-repo]$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
 new file:   haiku.txt
```

Initial commit

```
Initial commit█
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
#
# Initial commit
#
# Changes to be committed:
#       new file:   haiku.txt
#
```

Initial commit

```
[my-first-git-repo]$ git commit
[main (root-commit) 17d2441] Initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 haiku.txt
[my-first-git-repo]$ ls
haiku.txt
[my-first-git-repo]$ git status
On branch main
nothing to commit, working tree clean
[my-first-git-repo]$ git log
commit 17d2441d2c49913d9f022dd49f727f70010ce0c1 (HEAD -> main)
Author: Alok Menghrajani <441307+alokmenghrajani@users.noreply.github.com>
Date:   Fri Feb 7 10:42:24 2025 +0100

    Initial commit
```

git hashes

- Every object in the tree is identified by a hash
- An object’s hash is (conceptually):

  ```
  hash(diff + parent hashes + commit message, author, time, …)
  ```

  - where `hash()` is a one way function
  - and diff is the content change
- git is fast at showing you the state of the repo for a given hash or computing the delta between two hashes

Making changes

```
[my-first-git-repo]$ echo "Branches diverge, merge," >> haiku.txt
[my-first-git-repo]$ echo "History preserved" >> haiku.txt
[my-first-git-repo]$ echo "Moonlight dances on the soft clouds" > haiku2.txt
[my-first-git-repo]$ git status
On branch main
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:   haiku.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
 haiku2.txt

no changes added to commit (use "git add" and/or "git commit -a")
[my-first-git-repo]$ git diff
diff --git a/haiku.txt b/haiku.txt
index bd34e77..e95b648 100644
--- a/haiku.txt
+++ b/haiku.txt
@@ -1 +1,3 @@
 Code flows like rivers of change,
+Branches diverge, merge,
+History preserved
```

Making changes

```
[my-first-git-repo]$ git add haiku.txt
[my-first-git-repo]$ git add haiku2.txt
[my-first-git-repo]$ git commit -m 'Finish both haikus'
[main 9cc0f57] Finish both haikus
 2 files changed, 3 insertions(+)
 create mode 100644 haiku2.txt
[my-first-git-repo]$ git log
commit 9cc0f57593d59a1e8faec42e6be10be8b3fd6c2c (HEAD -> main)
Author: Alok Menghrajani <441307+alokmenghrajani@users.noreply.github.com>
Date:   Sat Feb 8 15:35:38 2025 +0100

    Finish both haikus

commit 17d2441d2c49913d9f022dd49f727f70010ce0c1
Author: Alok Menghrajani <441307+alokmenghrajani@users.noreply.github.com>
Date:   Fri Feb 7 10:42:24 2025 +0100

    Initial commit
```

git log --oneline

```
[my-first-git-repo]$ git log --oneline
9cc0f57 (HEAD -> main) Finish both haikus
17d2441 Initial commit
```

git log <filename>

```
[my-first-git-repo]$ git log haiku2.txt
commit 9cc0f57593d59a1e8faec42e6be10be8b3fd6c2c (HEAD -> main)
Author: Alok Menghrajani <441307+alokmenghrajani@users.noreply.github.com>
Date:   Sat Feb 8 15:35:38 2025 +0100

    Finish both haikus
```

git blame

```
[my-first-git-repo]$ git blame haiku.txt
^17d2441 (Alok Menghrajani 2025-02-07 10:42:24 +0100 1) Code flows like rivers of change,
9cc0f575 (Alok Menghrajani 2025-02-08 15:35:38 +0100 2) Branches diverge, merge,
9cc0f575 (Alok Menghrajani 2025-02-08 15:35:38 +0100 3) History preserved
```

git show <ref>

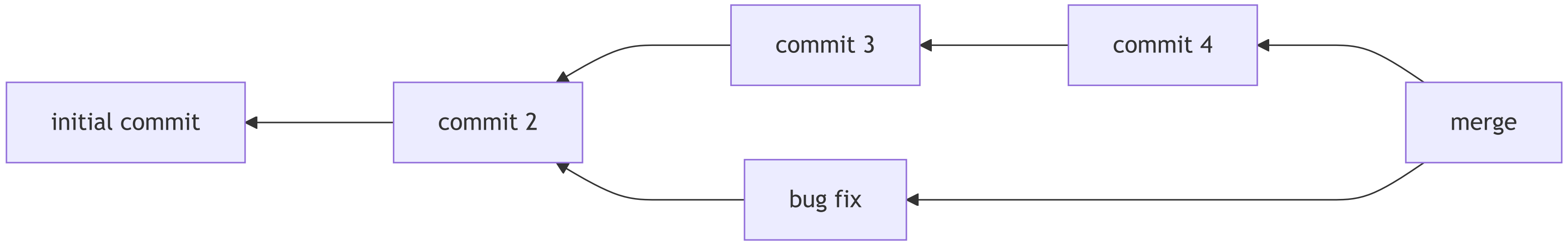
```
[my-first-git-repo]$ git show 17d2441d2c49913d9f022dd49f727f70010ce0c1
commit 17d2441d2c49913d9f022dd49f727f70010ce0c1
Author: Alok Menghrajani <441307+alokmenghrajani@users.noreply.github.com>
Date:   Fri Feb 7 10:42:24 2025 +0100

    Initial commit

diff --git a/haiku.txt b/haiku.txt
new file mode 100644
index 0000000..bd34e77
--- /dev/null
+++ b/haiku.txt
@@ -0,0 +1 @@
+Code flows like rivers of change,
```

git show <short hash>

```
[my-first-git-repo]$ git show 17d2
commit 17d2441d2c49913d9f022dd49f727f70010ce0c1
Author: Alok Menghrajani <441307+alokmenghrajani@users.noreply.github.com>
Date:   Fri Feb 7 10:42:24 2025 +0100

    Initial commit

diff --git a/haiku.txt b/haiku.txt
new file mode 100644
index 0000000..bd34e77
--- /dev/null
+++ b/haiku.txt
@@ -0,0 +1 @@
+Code flows like rivers of change,
```

branches and merges

Directed Acyclic Graph



Create a branch

```
[my-first-git-repo]$ git switch -c fix-typo
Switched to a new branch 'fix-typo'
[my-first-git-repo]$ git branch
* fix-typo
  main
```

Add commits to the new branch

```
[my-first-git-repo]$ sed -i '' -e 's/preserved/preserved./' haiku.txt
[my-first-git-repo]$ git diff
diff --git a/haiku.txt b/haiku.txt
index e95b648..d640618 100644
--- a/haiku.txt
+++ b/haiku.txt
@@ -1,3 +1,3 @@
 Code flows like rivers of change,
 Branches diverge, merge,
-History preserved
+History preserved.
[my-first-git-repo]$ git add haiku.txt
[my-first-git-repo]$ git commit -m 'Add missing punctuation'
[fix-typo 72ac719] Add missing punctuation
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Merge the branch

(happy case)

```
[my-first-git-repo]$ git switch main
Switched to branch 'main'
[my-first-git-repo]$ git merge --no-ff --no-edit fix-typo
Merge made by the 'ort' strategy.
 haiku.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
[my-first-git-repo]$ git branch -d fix-typo
Deleted branch fix-typo (was 72ac719).
```

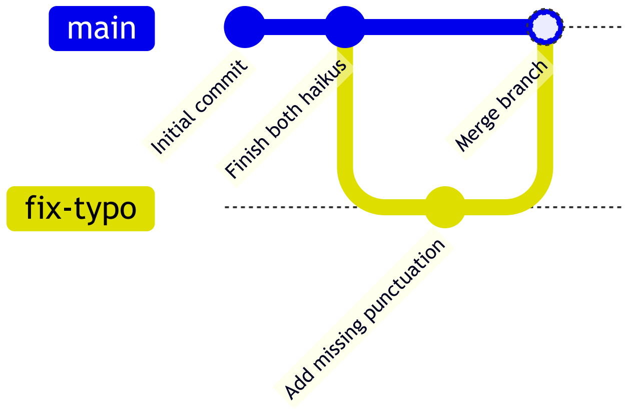
Merge the branch

```
[my-first-git-repo]$ git log
commit 23642369645a9a71983159d0c305abc44d5b9fd1 (HEAD -> main)
Merge: 9cc0f57 72ac719
Author: Alok Menghrajani <441307+alokmenghrajani@users.noreply.github.com>
Date:   Sat Feb 8 16:22:37 2025 +0100

    Merge branch 'fix-typo'

commit 72ac719503bc0dae974fb66d1e918a9d24daa5df
Author: Alok Menghrajani <441307+alokmenghrajani@users.noreply.github.com>
Date:   Sat Feb 8 16:21:25 2025 +0100

    Add missing punctuation

commit 9cc0f57593d59a1e8faec42e6be10be8b3fd6c2c
Author: Alok Menghrajani <441307+alokmenghrajani@users.noreply.github.com>
Date:   Sat Feb 8 15:35:38 2025 +0100

    Finish both haikus

commit 17d2441d2c49913d9f022dd49f727f70010ce0c1
Author: Alok Menghrajani <441307+alokmenghrajani@users.noreply.github.com>
Date:   Fri Feb 7 10:42:24 2025 +0100

    Initial commit
```

Merge the branch

```
[my-first-git-repo]$ git log --graph
*   commit 23642369645a9a71983159d0c305abc44d5b9fd1 (HEAD -> main)
|\  Merge: 9cc0f57 72ac719
| | Author: Alok Menghrajani <441307+alokmenghrajani@users.noreply.github.com>
| | Date:   Sat Feb 8 16:22:37 2025 +0100
| |
| |     Merge branch 'fix-typo'
| |
| * commit 72ac719503bc0dae974fb66d1e918a9d24daa5df
|/  Author: Alok Menghrajani <441307+alokmenghrajani@users.noreply.github.com>
|   Date:   Sat Feb 8 16:21:25 2025 +0100
|
|       Add missing punctuation
|
* commit 9cc0f57593d59a1e8faec42e6be10be8b3fd6c2c
| Author: Alok Menghrajani <441307+alokmenghrajani@users.noreply.github.com>
| Date:   Sat Feb 8 15:35:38 2025 +0100
|
|     Finish both haikus
|
* commit 17d2441d2c49913d9f022dd49f727f70010ce0c1
  Author: Alok Menghrajani <441307+alokmenghrajani@users.noreply.github.com>
  Date:   Fri Feb 7 10:42:24 2025 +0100

      Initial commit
```

Merge the branch



HEAD

```
[my-first-git-repo]$ git show HEAD
commit 23642369645a9a71983159d0c305abc44d5b9fd1 (HEAD -> main)
Merge: 9cc0f57 72ac719
Author: Alok Menghrajani <441307+alokmenghrajani@users.noreply.github.com>
Date:   Sat Feb 8 16:22:37 2025 +0100

    Merge branch 'fix-typo'
[my-first-git-repo]$ git show HEAD~2
commit 17d2441d2c49913d9f022dd49f727f70010ce0c1
Author: Alok Menghrajani <441307+alokmenghrajani@users.noreply.github.com>
Date:   Fri Feb 7 10:42:24 2025 +0100

    Initial commit

diff --git a/haiku.txt b/haiku.txt
new file mode 100644
index 0000000..bd34e77
--- /dev/null
+++ b/haiku.txt
@@ -0,0 +1 @@
+Code flows like rivers of change,
```

Merge with conflicts

- `brA` and `brB` branch off `main`
- you commit changes to `brA`
- you commit changes to `brB`
- you merge `brA` into `main`
  - this merge will not conflict
- you merge `brB` into main
  - this merge will conflict

Automatic conflict resolution

```
[my-first-git-repo]$ git switch -c brB
Switched to a new branch 'brB'
[my-first-git-repo]$ git switch -c brA
Switched to a new branch 'brA'
[my-first-git-repo]$ sed -i '' -e 's/Code/git/' haiku.txt
[my-first-git-repo]$ git commit -a -m "Replace 'Code' with 'git'"
[brA ae8ec80] Replace 'Code' with 'git'
 1 file changed, 1 insertion(+), 1 deletion(-)
[my-first-git-repo]$ git switch brB
Switched to branch 'brB'
[my-first-git-repo]$ sed -i '' -e 's/History/Everything/' haiku.txt
[my-first-git-repo]$ git commit -a -m "Replace 'History' with 'Everything'"
[brB 2cfc68b] Replace 'History' with 'Everything'
 1 file changed, 1 insertion(+), 1 deletion(-)
[my-first-git-repo]$ git switch main
Switched to branch 'main'
```

Automatic conflict resolution

```
[my-first-git-repo]$ git merge --no-ff --no-edit brA
Merge made by the 'ort' strategy.
 haiku.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
[my-first-git-repo]$ git merge --no-ff --no-edit brB
Auto-merging haiku.txt
Merge made by the 'ort' strategy.
 haiku.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
[my-first-git-repo]$ cat haiku.txt
git flows like rivers of change,
Branches diverge, merge,
Everything preserved.
[my-first-git-repo]$ git branch -d brA
Deleted branch brA (was ae8ec80).
[my-first-git-repo]$ git branch -d brB
Deleted branch brB (was 2cfc68b).
```

Manual conflict resolution

```
[my-first-git-repo]$ git switch -c brB
Switched to a new branch 'brB'
[my-first-git-repo]$ git switch -c brA
Switched to a new branch 'brA'
[my-first-git-repo]$ sed -i '' -e 's/git/Life/' haiku.txt
[my-first-git-repo]$ git commit -a -m "Replace 'git' with 'Life'"
[brA 94c0e50] Replace 'git' with 'Life'
 1 file changed, 1 insertion(+), 1 deletion(-)
[my-first-git-repo]$ git switch brB
Switched to branch 'brB'
[my-first-git-repo]$ sed -i '' -e 's/change,/change!/' haiku.txt
[my-first-git-repo]$ git commit -a -m "Change punctuation"
[brB 63848e8] Change punctuation
 1 file changed, 1 insertion(+), 1 deletion(-)
[my-first-git-repo]$ git switch main
Switched to branch 'main'
```

Manual conflict resolution

```
[my-first-git-repo]$ git merge --no-ff --no-edit brA
Merge made by the 'ort' strategy.
 haiku.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
[my-first-git-repo]$ git merge --no-ff --no-edit brB
Auto-merging haiku.txt
CONFLICT (content): Merge conflict in haiku.txt
Automatic merge failed; fix conflicts and then commit the result.
[my-first-git-repo]$ head -n1 haiku.txt
<<<<<<< HEAD
Life flows like rivers of change,
=======
git flows like rivers of change!
>>>>>>> brB
[my-first-git-repo]$ git restore --ours haiku.txt
[my-first-git-repo]$ head -n1 haiku.txt
Life flows like rivers of change,
[my-first-git-repo]$ git restore --theirs haiku.txt
[my-first-git-repo]$ head -n1 haiku.txt
git flows like rivers of change!
```

Manual conflict resolution

```
[my-first-git-repo]$ sed -i '' '1s/.*/Life flows like rivers of change!/' haiku.txt
[my-first-git-repo]$ git add haiku.txt
[my-first-git-repo]$ git commit -m 'Manually merged brB'
[main c7f3e24] Manually merged brB
[my-first-git-repo]$ git log --oneline --topo-order HEAD~2..HEAD
8982205 (HEAD -> main) Manually merged brB
63848e8 (brB) Change punctuation
eab1ee3 Merge branch 'brA'
94c0e50 (brA) Replace 'git' with 'Life'
```

Manual conflict resolution

```
[my-first-git-repo]$ git show 898220580ee69f4535131210861c3d4ae73271df
commit 898220580ee69f4535131210861c3d4ae73271df (HEAD -> main)
Merge: eab1ee3 63848e8
Author: Alok Menghrajani <441307+alokmenghrajani@users.noreply.github.com>
Date:   Sat Feb 8 17:22:34 2025 +0100

    Manually merged brB

diff --cc haiku.txt
index a2b1fae,6052095..96082fb
--- a/haiku.txt
+++ b/haiku.txt
@@@ -1,3 -1,3 +1,3 @@@
- Life flows like rivers of change,
 -git flows like rivers of change!
++Life flows like rivers of change!
  Branches diverge, merge,
  Everything preserved.
```

Tips for authoring commits

- Each commit should be as small as possible
- Single cohesive idea
- Split large commits into small commits by dividing large problems into smaller pieces

Tips for writing commit messages

- Explain the gist of the change in the title
- Explain the reason for the change in the description
- Any other information which might be useful to you or someone else looking at the commit out of context
  - For bug fixes: link to revision which introduced the bug
  - For features: task or discussion tracking the feature
  - How you tested the change
  - ...

Tips for writing commit messages

Example commit

```
Fix: ignore link headers for responsive images

Link headers are optionally supported for cases where you prefer to send
resource loading hints before you're ready to send the body of a
request. Many resources can be correctly preloaded from a link
header however responsive images are currently not supported. They result
in broken images.

This commit disables preloading responsive images until preloading them
is fully supported (the future work is tracked in #23452).

blame rev: 67ee48f7

test plan: included unit test fails without this fix.

closes: #12521
```

git: more commands

Working with remotes

- `git remote -v`
- `git fetch -p`
- `git branch -a`
- `git push origin HEAD`

git commit --amend

⚠️ rewriting history

- Mutate most recent commit
- Edit a commit message

Rebase

- `git rebase <ref>`
- `git rebase -i <ref>`

git reflog

- Operate on the local history (reference logs)
- Enables undoing a rebase

git cherry-pick

Move HEAD

⚠️ uncommitted changes can be lost

- `git reset --soft`
- `git reset --hard`
- `git restore <ref>`
- `git restore <ref> <files>`

git add

- `git add -p`
- `git add .`
- `git commit -a`

git revert <commit>

git rm <filename>

git mv <filename>

Searching

- `git log -S <string>`
- `git grep <string>`

git tag

- A shareable bookmark
- `git tag -a v1.1 -m "say something about this version…"`
- `git tag` to list tags

Stashing

- `git stash`
- `git stash list`
- `git stash pop`