[320] Reproducibility 2: Version Control (git)

Yiyin Shen

Reproducibility

Big question: *will my program run on someone else's computer?*

Things to match:

1. Hardware ← a program must fit the CPU; `python.exe` will do this, so `program.py` won't have to
2. Operating System ← we'll use Ubuntu Linux on virtual machines in the cloud
3. Dependencies ← today: versioning

Dependency Versions

this program "depends" on pandas

program.py

```
import os, sys, json
import pandas

import pandas

print("Pandas Version:", pandas.__version__)


# code that uses pandas
```

you can check a module version

behavior depends on which release was installed

```
pip install pandas
```

or

```
pip install pandas==0.25.1
```

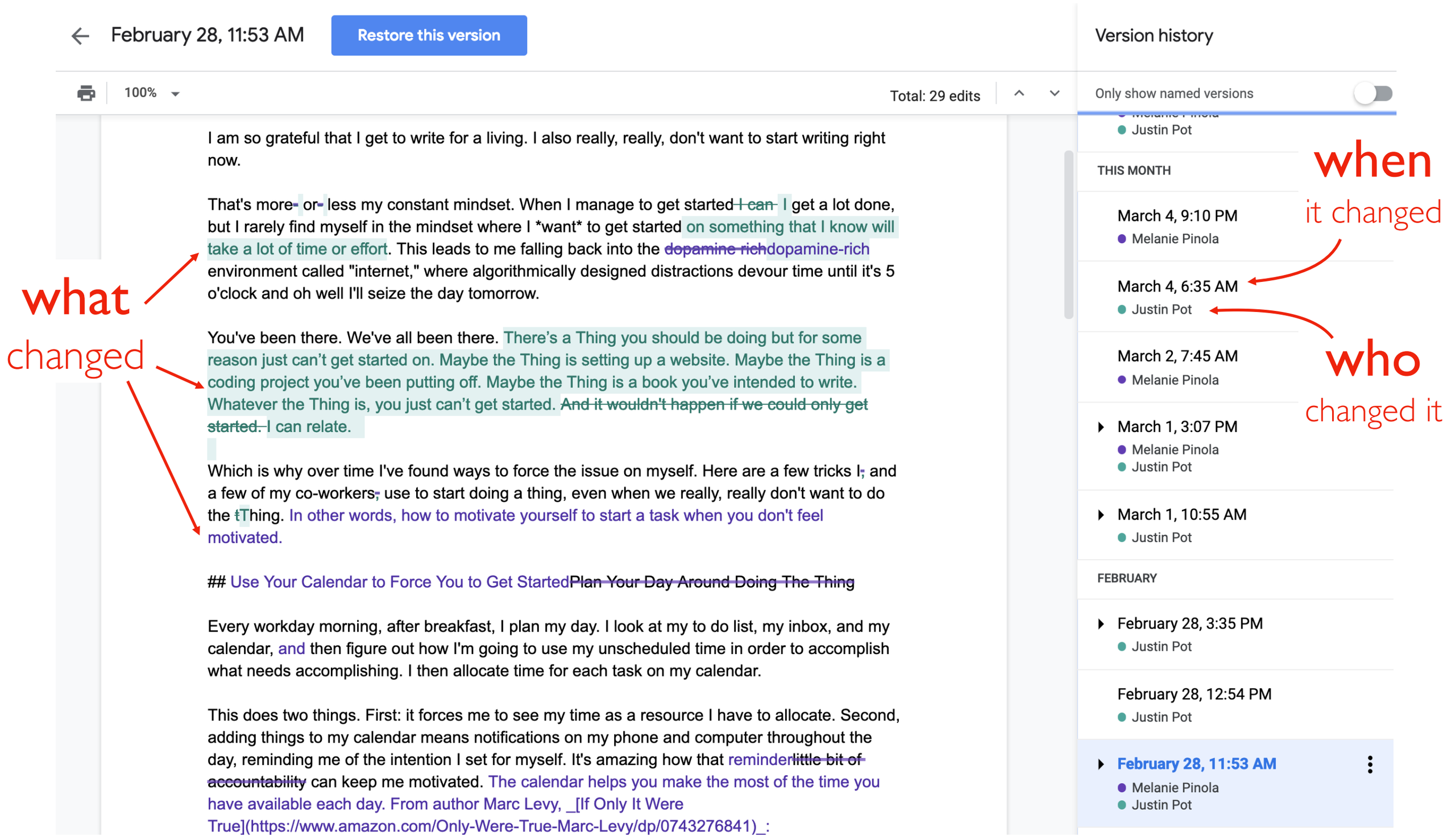
or

```
pip install pandas==0.24.0
```

or...

Versioning: motivation and basic concepts

Many tools auto-track history (e.g., Google Docs)



https://zapier.com/apps/google-docs/tutorials/google-docs-revision-history

Version Control Systems (VCS)

Useful for many kinds of projects

- code, papers, websites, etc
- manages all files for same project (maybe thousands) in a repository

Explicit snapshots/checkpoints, called *commits*

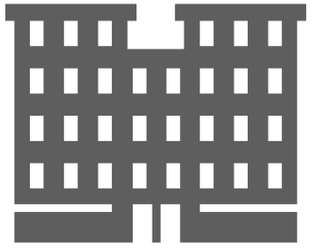
- users **manually** run commands to preserve good versions

Explicit commit messages

- who, what, when, **why**

Work can *branch* out and be *merged* back

- people can work offline
- can get feedback before merging
- humans need to resolve ***conflicts*** when versions being merged are too different

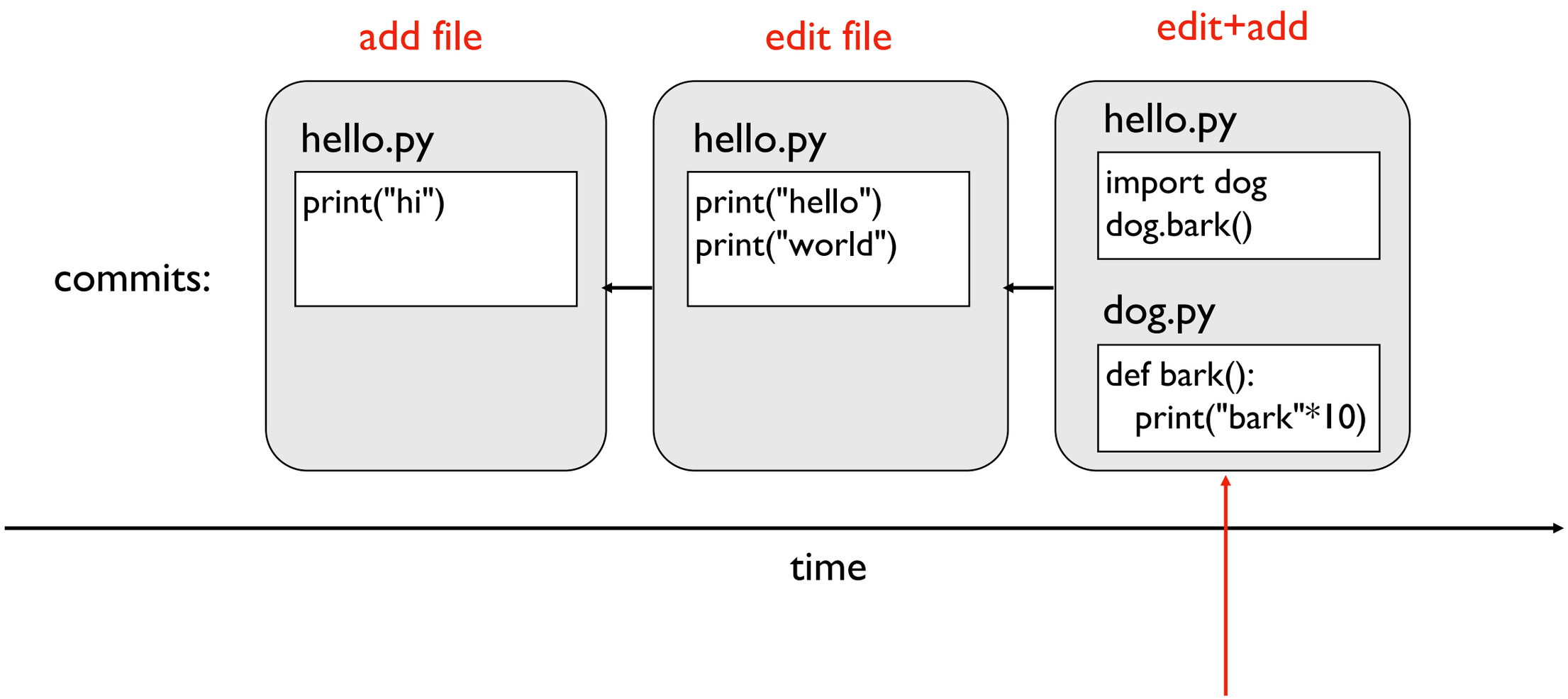


partner A working on `hw.py` at school

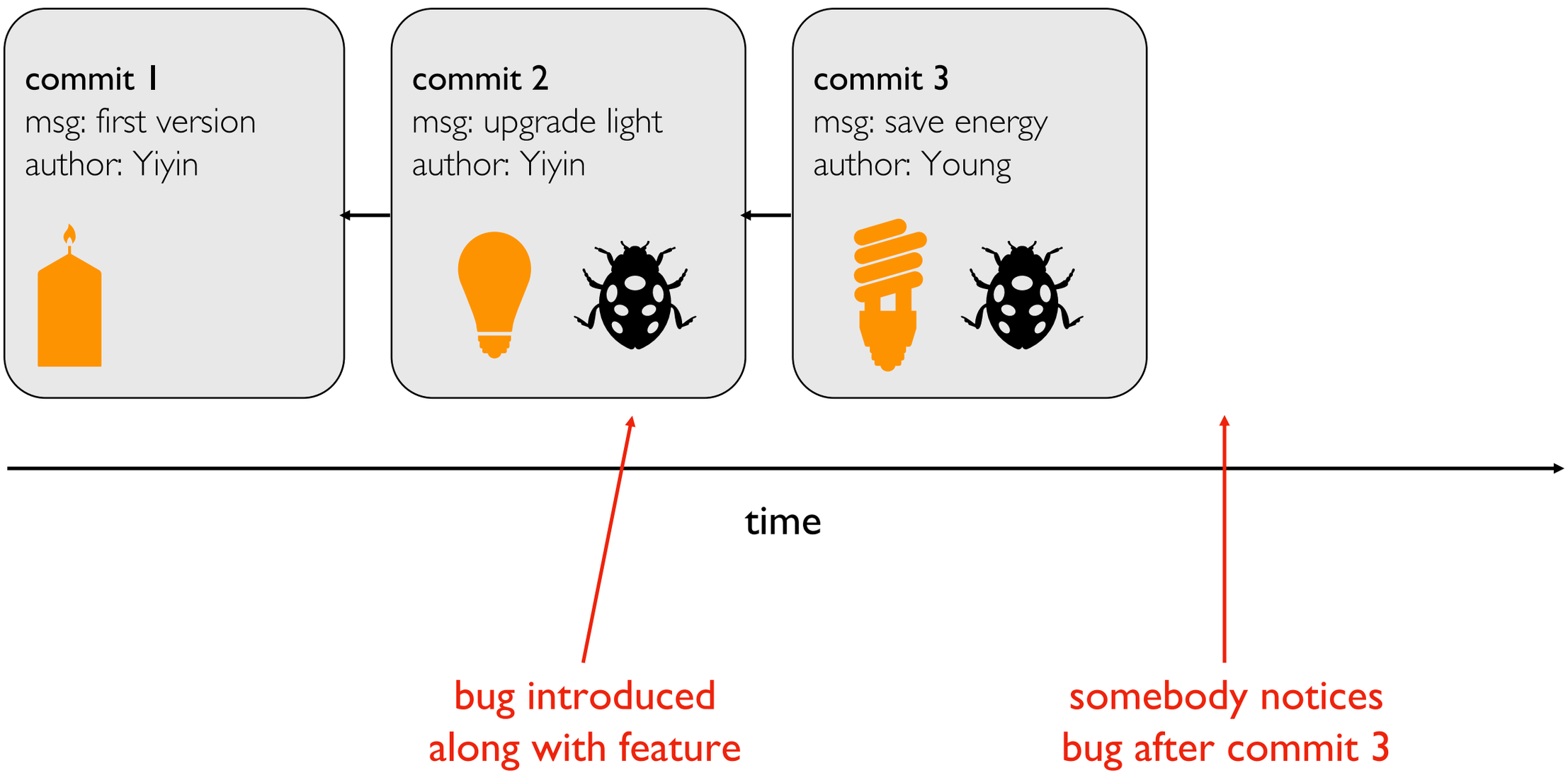
partner B also working on `hw.py`, without wifi

what happens when the plane lands?

Example

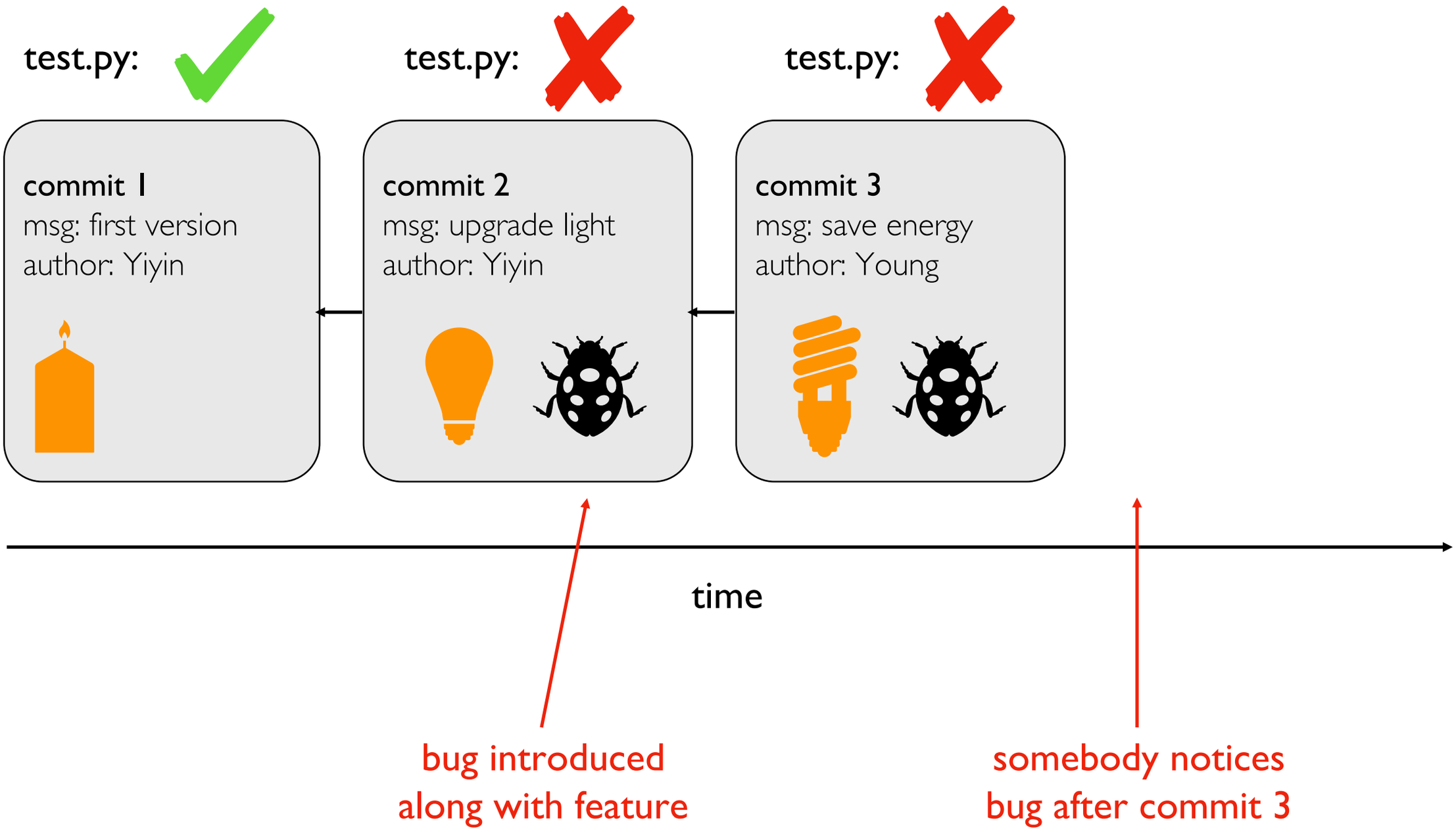

add file
edit file
edit+add
commits:
hello.py
print("hi")
hello.py
print("hello")
print("world")
hello.py
import dog
dog.bark()
dog.py
def bark():
print("bark"*10)
time
at any point in time,
you just see one version
of the files on your computer

Use case 1: troubleshooting discovered bug

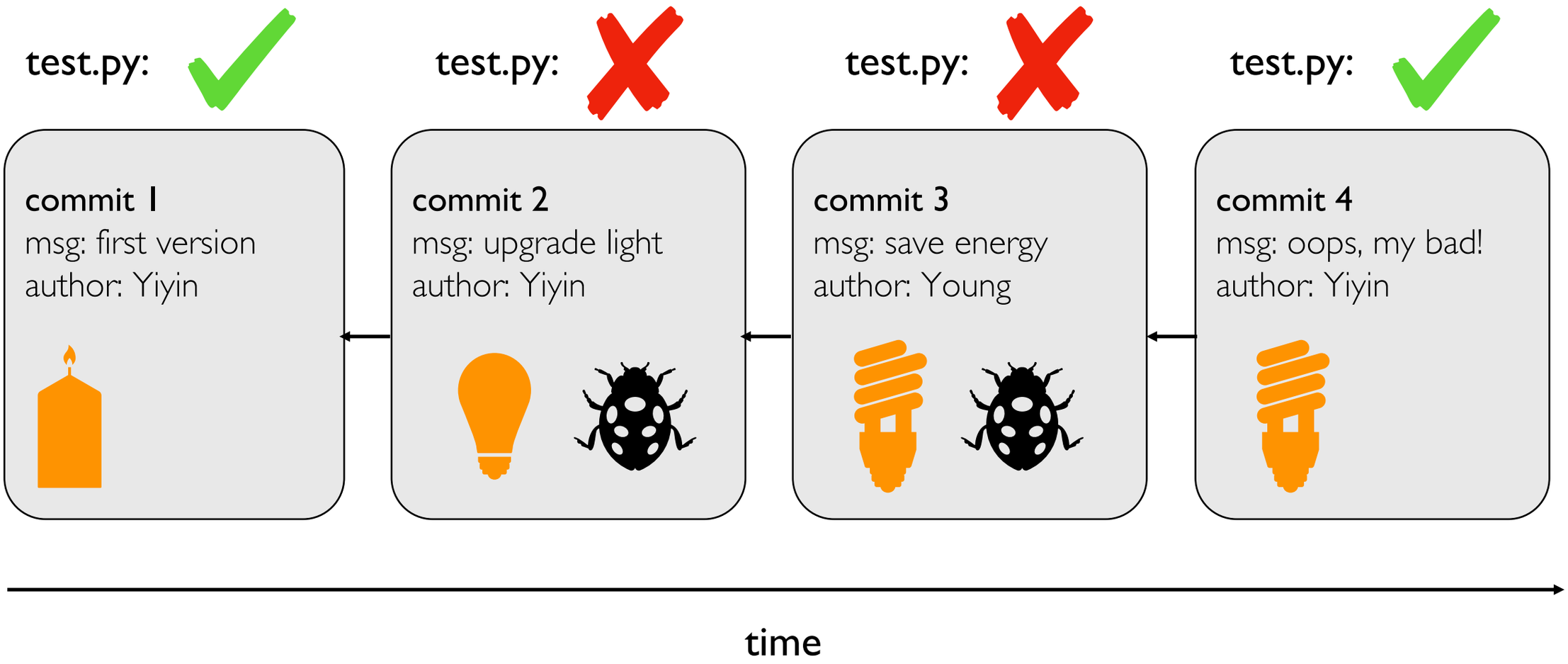


who will get blamed?

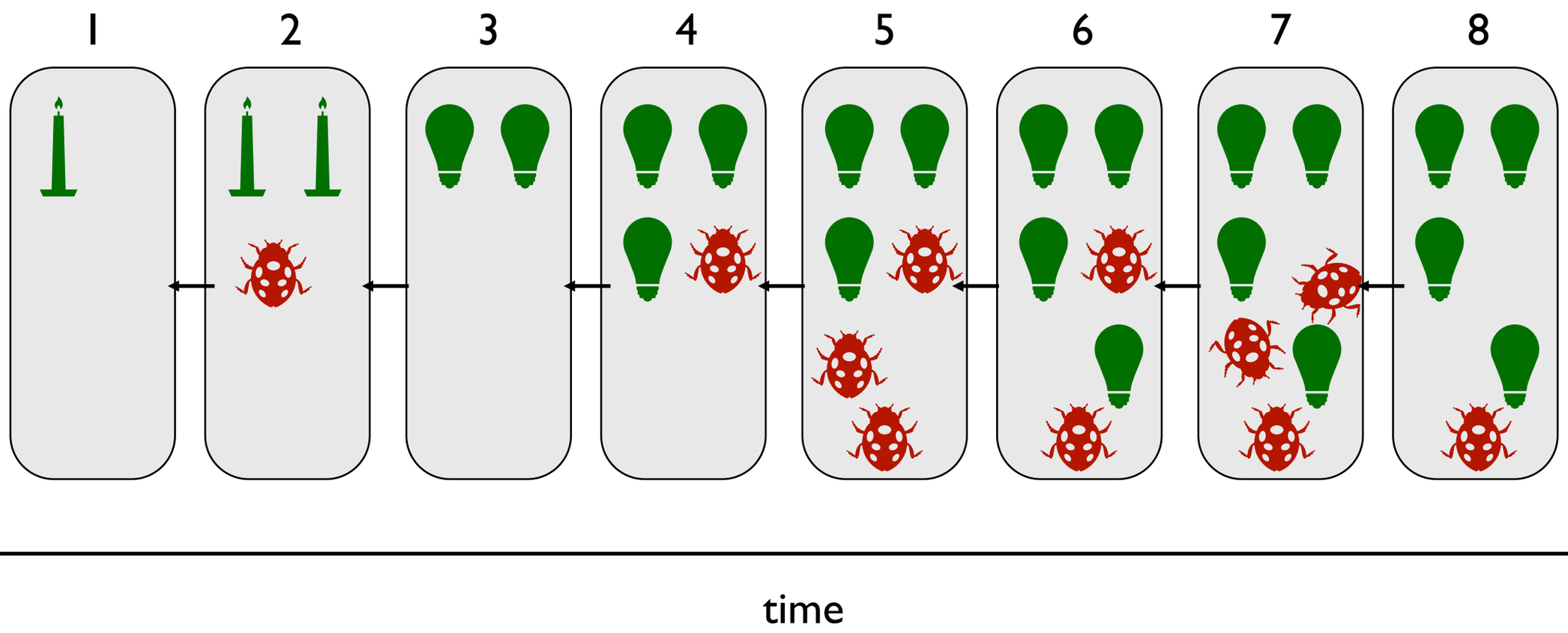
Use case 1: troubleshooting discovered bug



Use case 1: troubleshooting discovered bug

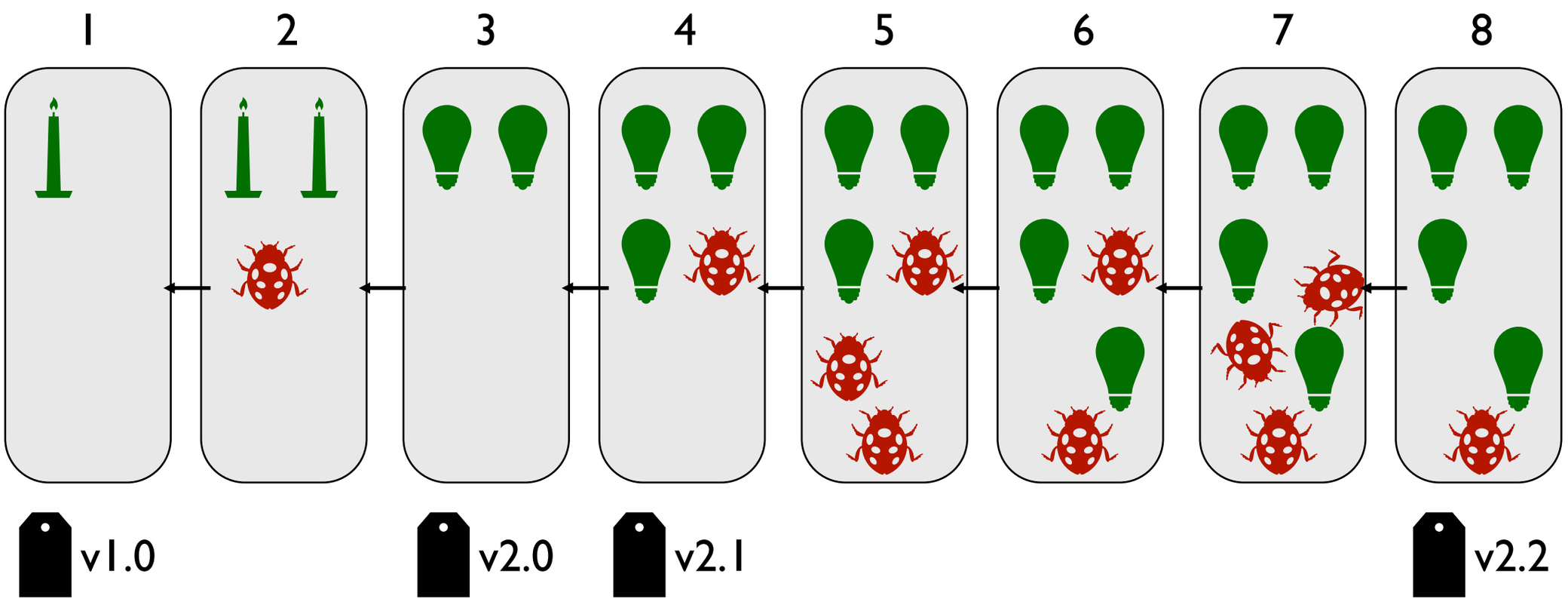


Use case 2: versioned releases



which version would you use?

Use case 2: versioned releases

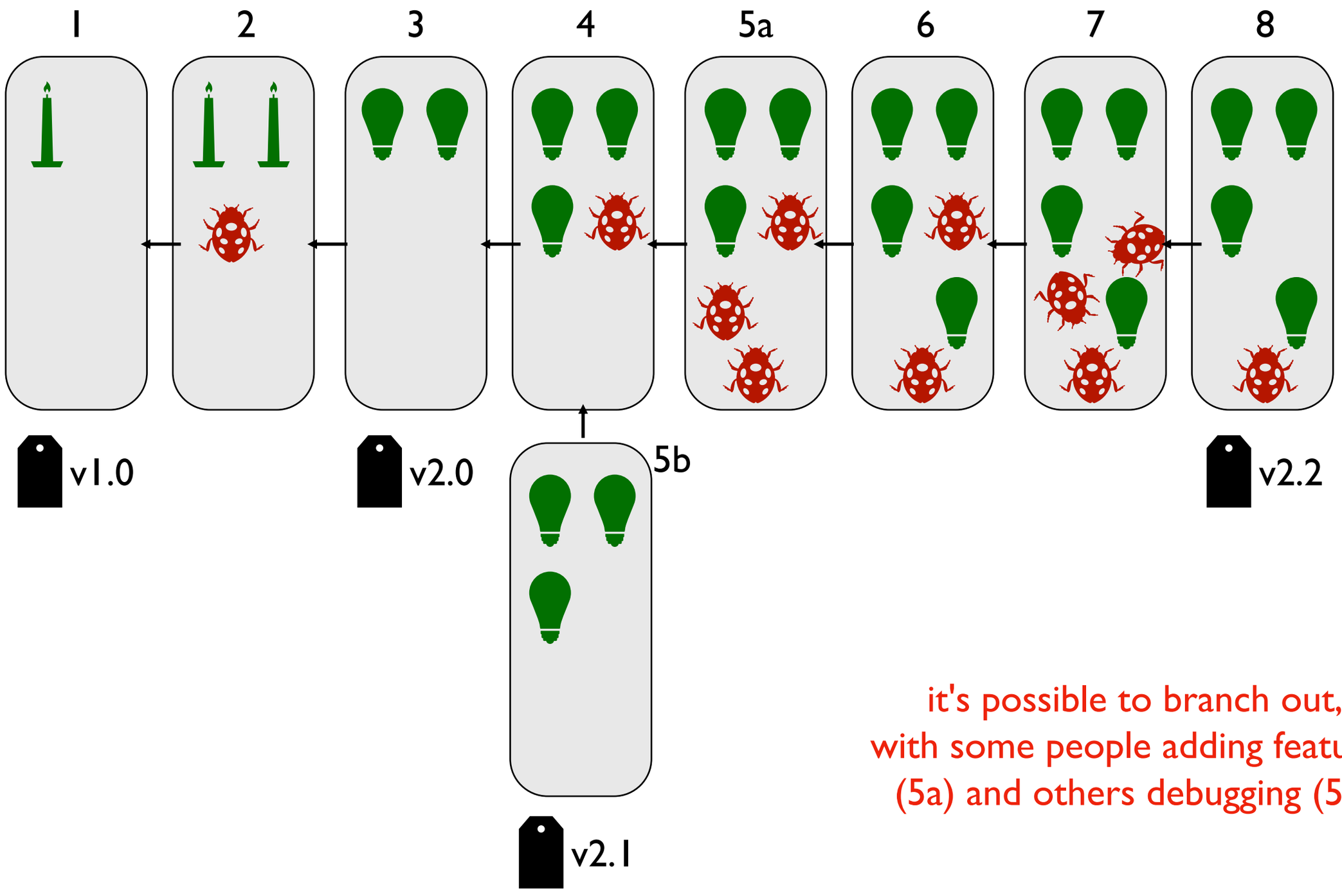


tag "good" commits to create releases

https://pypi.org/project/pandas/#history

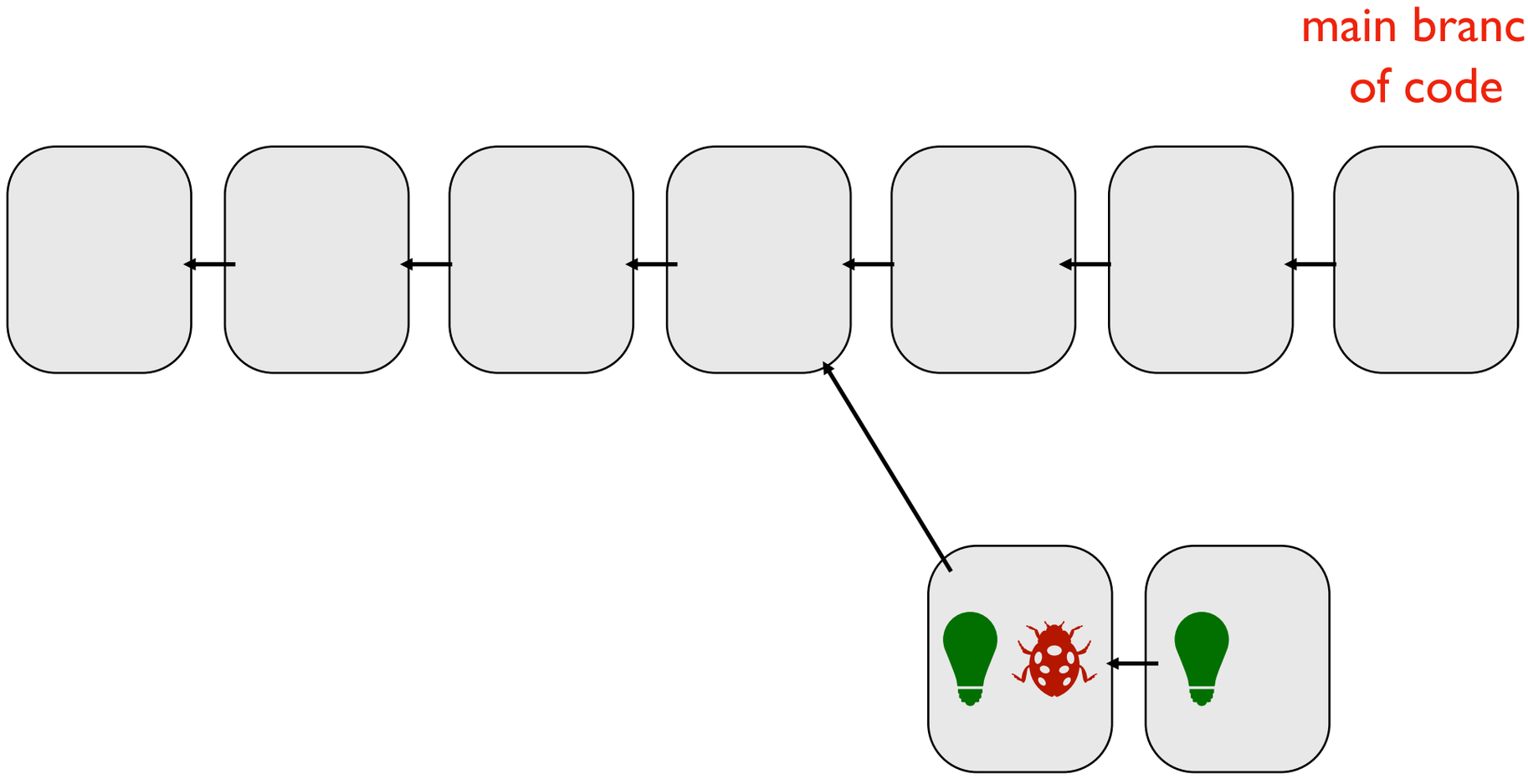
https://github.com/pandas-dev/pandas/releases

Use case 2: versioned releases

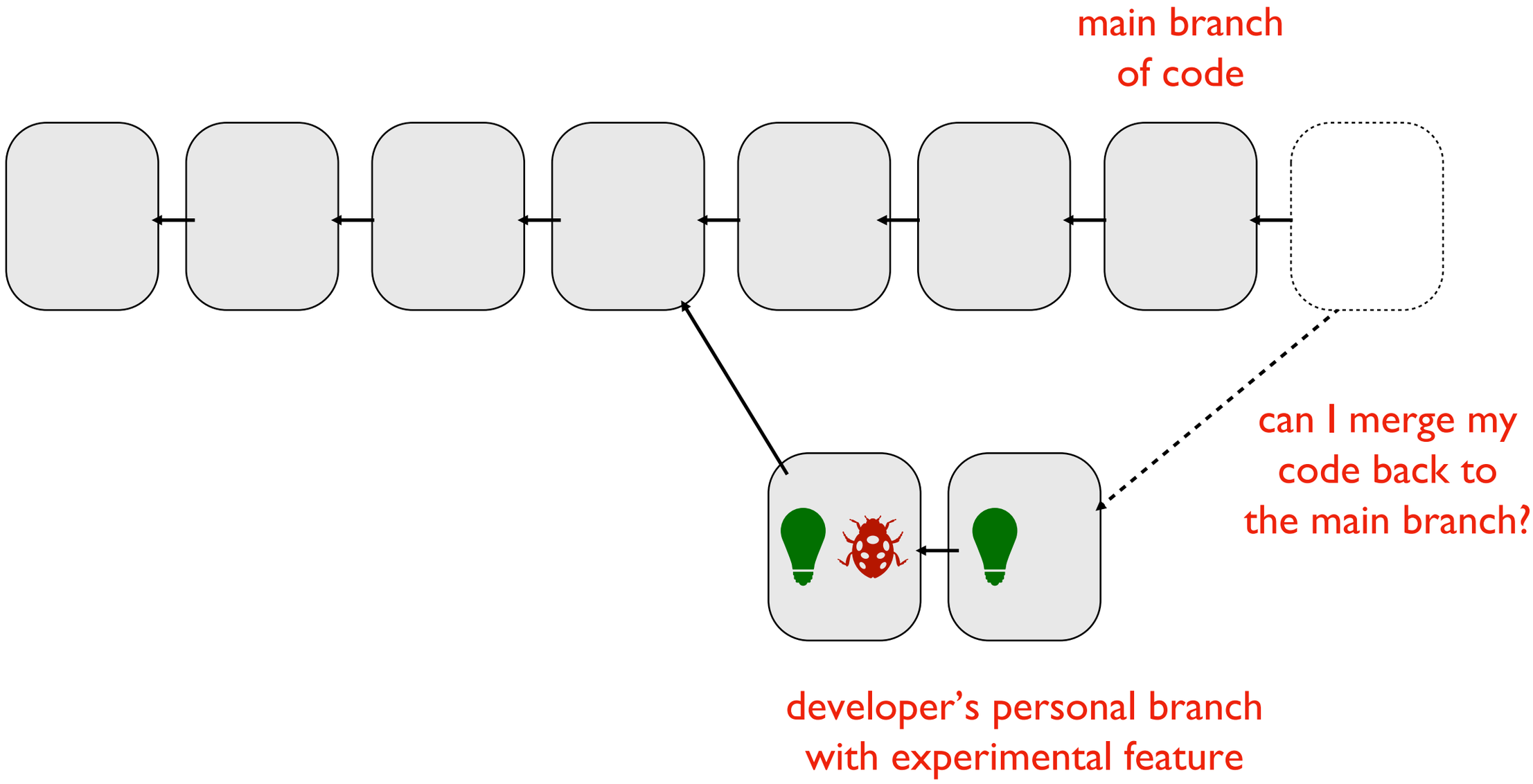


it's possible to branch out,
with some people adding features
(5a) and others debugging (5b)

Use case 3: feedback



Use case 3: feedback



git

Version Control System Tools

tools

svn

git

Mercurial

TeamFoundation

git providers

GitLab

BitBucket

GitHub: signup for a free account for next weeks lab

- **do** choose a name that won't embarrass you on a resume
- **do not** post course work

Linus Torvalds developed git to manage Linux as a BitKeeper replacement

https://www.linuxjournal.com/content/25-years-later-interview-linus-torvalds

Git Demos

Lecture notes/readings/past exams repository:
https://github.com/yiyins2/CS320-FA23-lecture-notes

Projects/labs repository:
https://github.com/yiyins2/CS320-FA23

Activities:

- Connect to VM through SSH
- Clone a GitHub repo to the VM
- View repo history
- Switch between commits (versions)
- Write new commits

Git Demos

Connect to VM:

- Mac: terminal; Windows: powershell
- ssh username@computer: connect to a VM via SSH

Shortcuts:

- ^D exit connection
- ^C terminate the current command
- ^R search history

Linux command line:

- pwd display current working directly
- cd folder go down a directory
- cd .. go up a directory
- ls list all files in the directory
- cat display the files

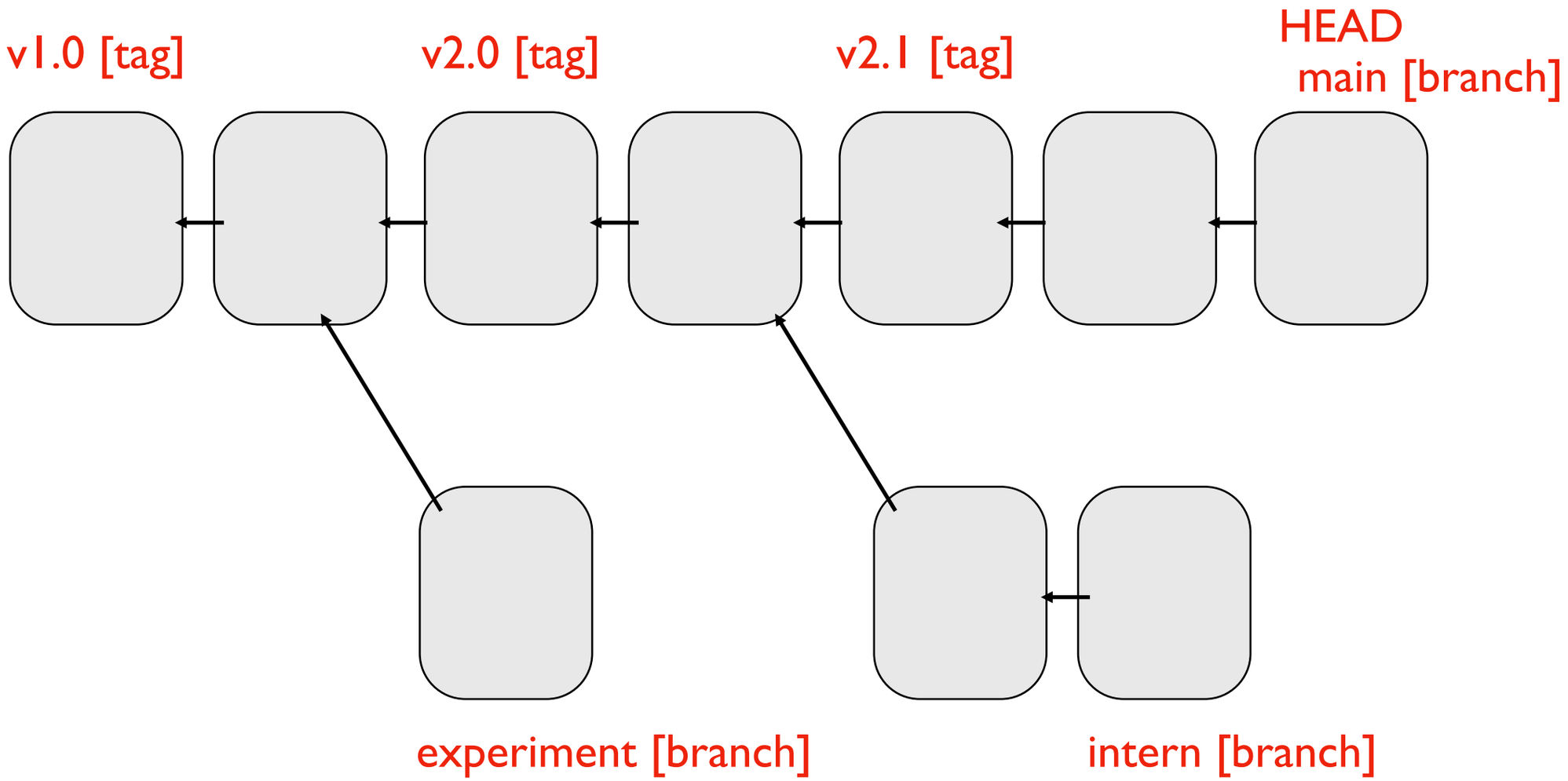
Git Demos

Git Commands:

- git clone: retrieve an entire repository from a hosted location via URL
- git log: show all commits in the current branch's history
- git status: show modified files in working directory, staged for your next commit
- git pull: fetch and merge any commits from the tracking remote branch
- git add: add a file as it looks now to your next commit (stage)
- git commit: commit your staged content as a new commit snapshot
- git push: transmit local branch commits to the remote repository branch
- git branch: list your branches. a * will appear next to the currently active branch
- git checkout: switch to another branch and check it out into your working directory

HEAD, Branches, and Tags

Remembering commit numbers is a pain! Various kinds of labels can serve as easy-to-remember aliases



HEAD: wherever you currently are (only one of these)
tag: label tied to a specific commit number
branch: label tied to end of chain (moves upon new commits)