

[320] Version Control (git)

Tyler Caraza-Harter

Review

A running program is called a _____

fruits is a large list. Which will be faster?

1. `fruits.insert(0, "pineapple")` # adds to beginning of list
2. `fruits.pop(-1)` # removes from end of list

What is an example of resource that an operating system might allocate to a process?

what does a Python code usually need to worry more about matching?

1. hardware (especially CPU's instruction set)
2. operating system

Review

A running program is called a process

fruits is a large list. Which will be faster?

1. `fruits.insert(0, "pineapple")` # adds to beginning of list
2. `fruits.pop(-1)` # removes from end of list

What is an example of resource that an operating system might allocate to a process?

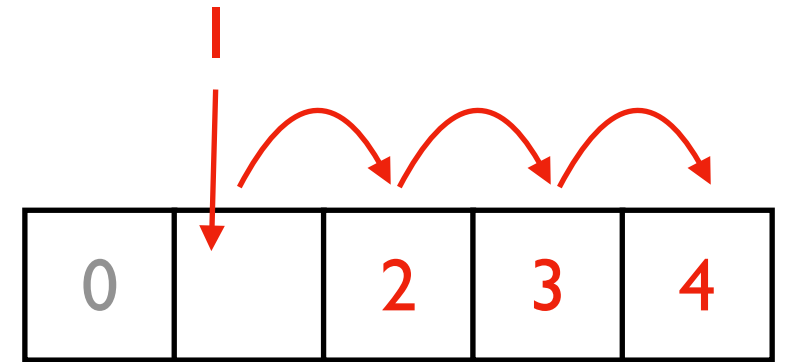
what does a Python code usually need to worry more about matching?

1. hardware (especially CPU's instruction set)
2. operating system

Review

A running program is called a process

fruits is a large list. Which will be faster?



1. `fruits.insert(0, "pineapple")` # adds to beginning of list

2. `fruits.pop(-1)` # removes from end of list

What is an example of resource that an operating system might allocate to a process?

what does a Python code usually need to worry more about matching?

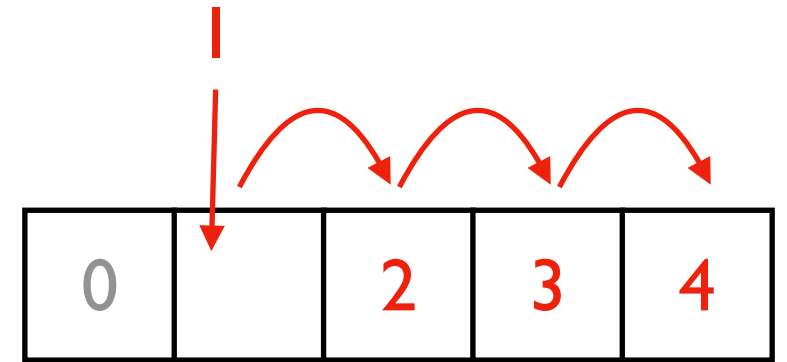
1. hardware (especially CPU's instruction set)

2. operating system

Review

A running program is called a process

fruits is a large list. Which will be faster?



1. `fruits.insert(0, "pineapple")` # adds to beginning of list

2. `fruits.pop(-1)` # removes from end of list

What is an example of resource that an operating system might allocate to a process?

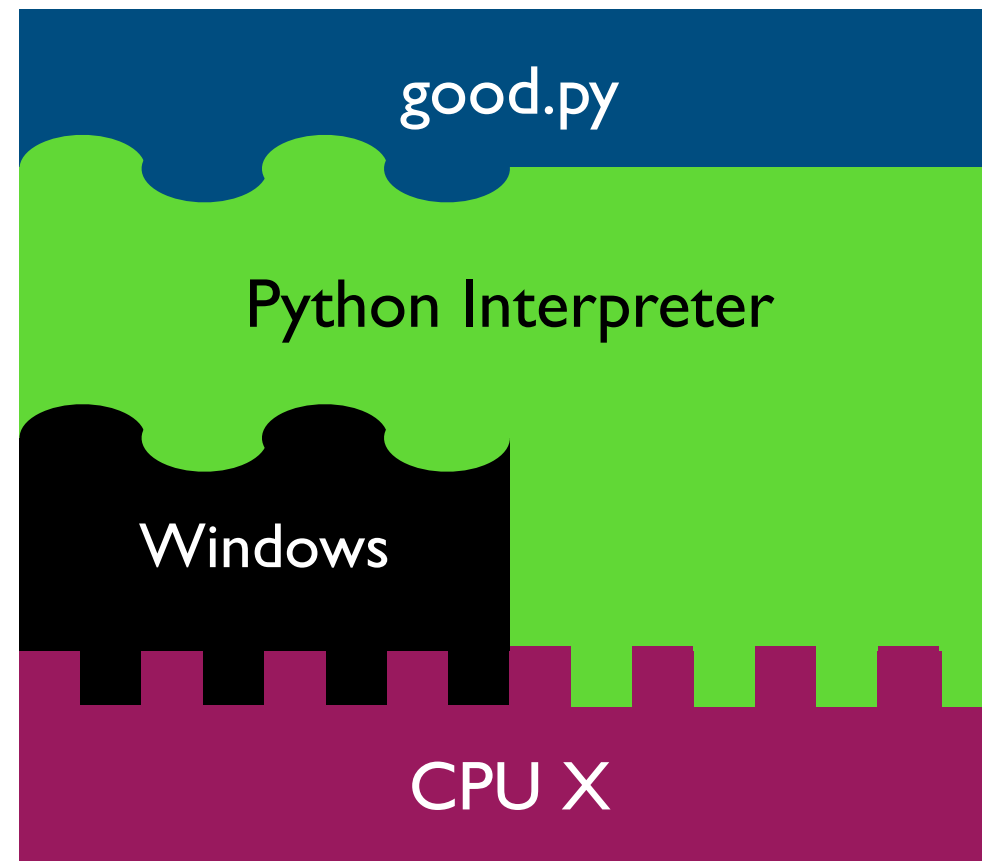
time on CPU, space in memory, space in files, network bandwidth

what does a Python code usually need to worry more about matching?

1. hardware (especially CPU's instruction set)

2. operating system

Review



what does a Python code usually need to worry more about matching?

1. hardware (especially CPU's instruction set)

2. operating system

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

program.py

```
import os, sys, json
import pandas

import pandas

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

# code that uses pandas
```

this program "depends" on 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)

The screenshot displays a Google Docs document with a version history sidebar on the right. The document text includes several paragraphs, some of which are highlighted in green. Red handwritten annotations are present: 'what changed' with arrows pointing to the first two paragraphs, 'when it changed' with an arrow pointing to the March 4, 9:10 PM version, and 'who changed it' with an arrow pointing to the March 4, 6:35 AM version.

← February 28, 11:53 AM [Restore this version](#)

100% Total: 29 edits

Version history

Only show named versions

THIS MONTH

- March 4, 9:10 PM (Melanie Pinola)
- March 4, 6:35 AM (Justin Pot)
- March 2, 7:45 AM (Melanie Pinola)
- March 1, 3:07 PM (Melanie Pinola, Justin Pot)
- March 1, 10:55 AM (Justin Pot)

FEBRUARY

- February 28, 3:35 PM (Justin Pot)
- February 28, 12:54 PM (Justin Pot)
- February 28, 11:53 AM** (Melanie Pinola, Justin Pot)

I am so grateful that I get to write for a living. I also really, really, don't want to start writing right now.

That's more- or- less my constant mindset. When I manage to get started ~~I can~~ I get a lot done, but I rarely find myself in the mindset where I *want* to get started ~~on something that I know will take a lot of time or effort~~. This leads to me falling back into the ~~dopamine-rich~~ dopamine-rich environment called "internet," where algorithmically designed distractions devour time until it's 5 o'clock and oh well I'll seize the day tomorrow.

You've been there. We've all been there. ~~There's a Thing you should be doing but for some reason just can't get started on. Maybe the Thing is setting up a website. Maybe the Thing is a coding project you've been putting off. Maybe the Thing is a book you've intended to write. Whatever the Thing is, you just can't get started. And it wouldn't happen if we could only get started. I can relate.~~

Which is why over time I've found ways to force the issue on myself. Here are a few tricks I, and a few of my co-workers, use to start doing a thing, even when we really, really don't want to do the ~~t~~Thing. ~~In other words, how to motivate yourself to start a task when you don't feel motivated.~~

~~## Use Your Calendar to Force You to Get Started~~ ~~Plan Your Day Around Doing The Thing~~

Every workday morning, after breakfast, I plan my day. I look at my to do list, my inbox, and my calendar, ~~and~~ then figure out how I'm going to use my unscheduled time in order to accomplish what needs accomplishing. I then allocate time for each task on my calendar.

This does two things. First: it forces me to see my time as a resource I have to allocate. Second, adding things to my calendar means notifications on my phone and computer throughout the day, reminding me of the intention I set for myself. It's amazing how that ~~reminder~~ little bit of accountability can keep me motivated. ~~The calendar helps you make the most of the time you have available each day.~~ From author Marc Levy, *[If Only It Were True]*(<https://www.amazon.com/Only-Were-True-Marc-Levy/dp/0743276841>):

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**

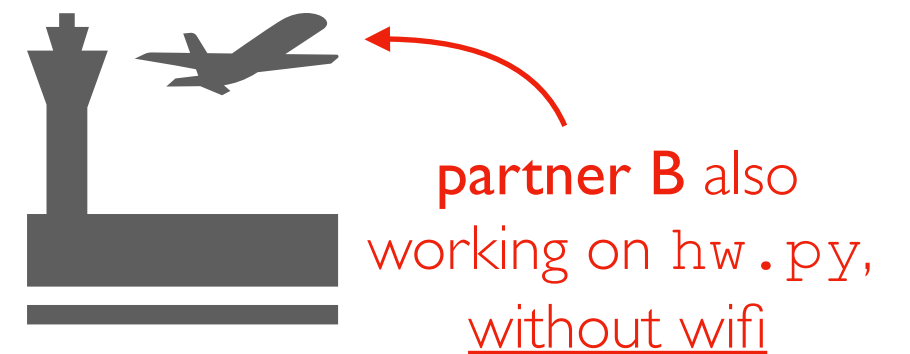
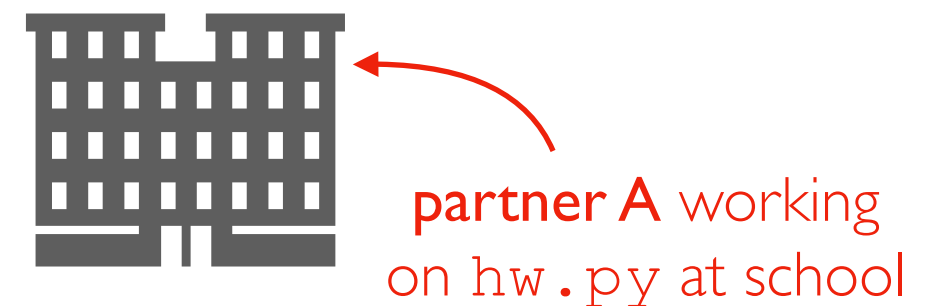
- users **manually** run commands to preserve good versions

Explicit **commit messages**

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

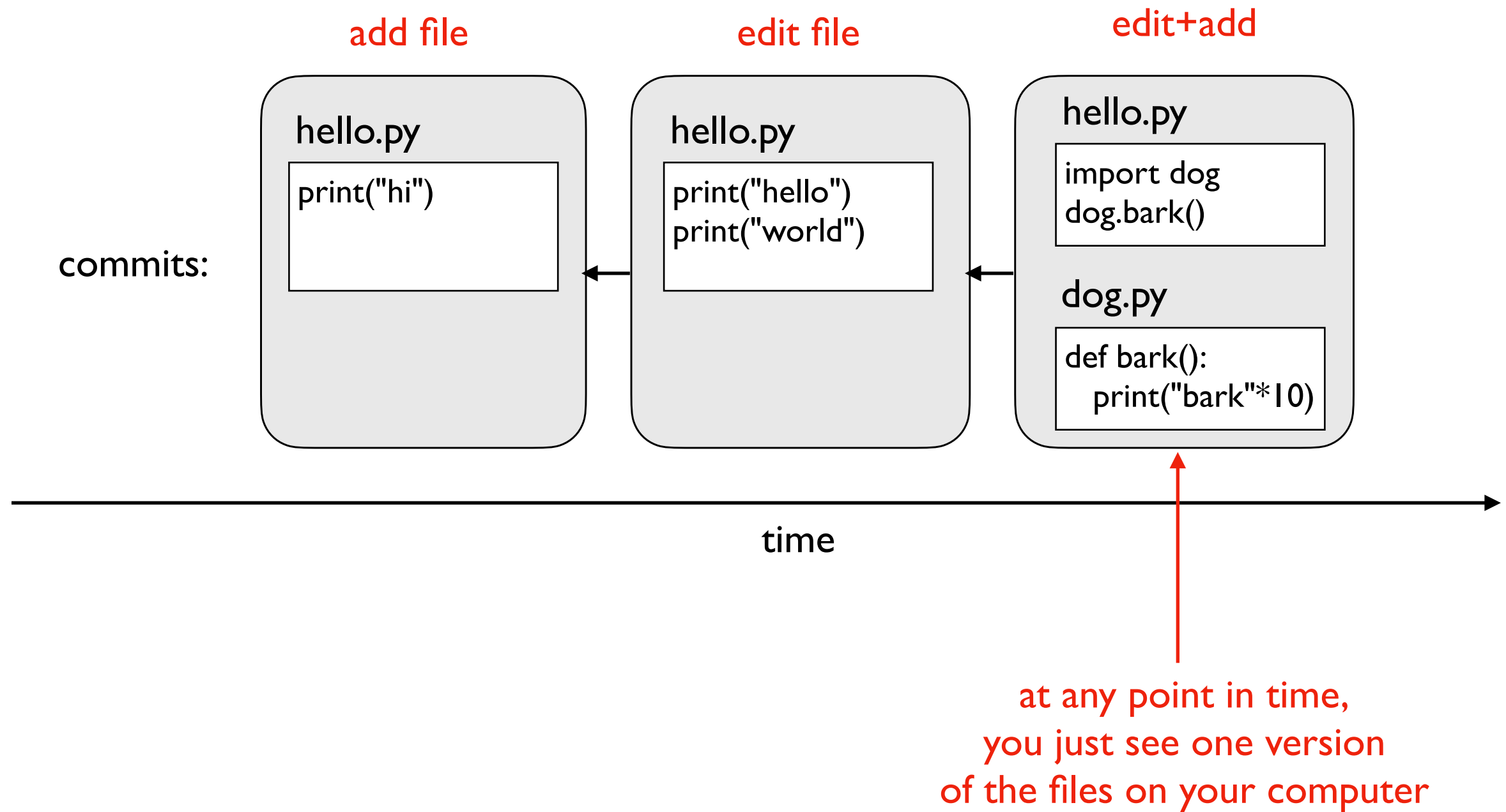
Work can **branch** out and be **merged** back

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

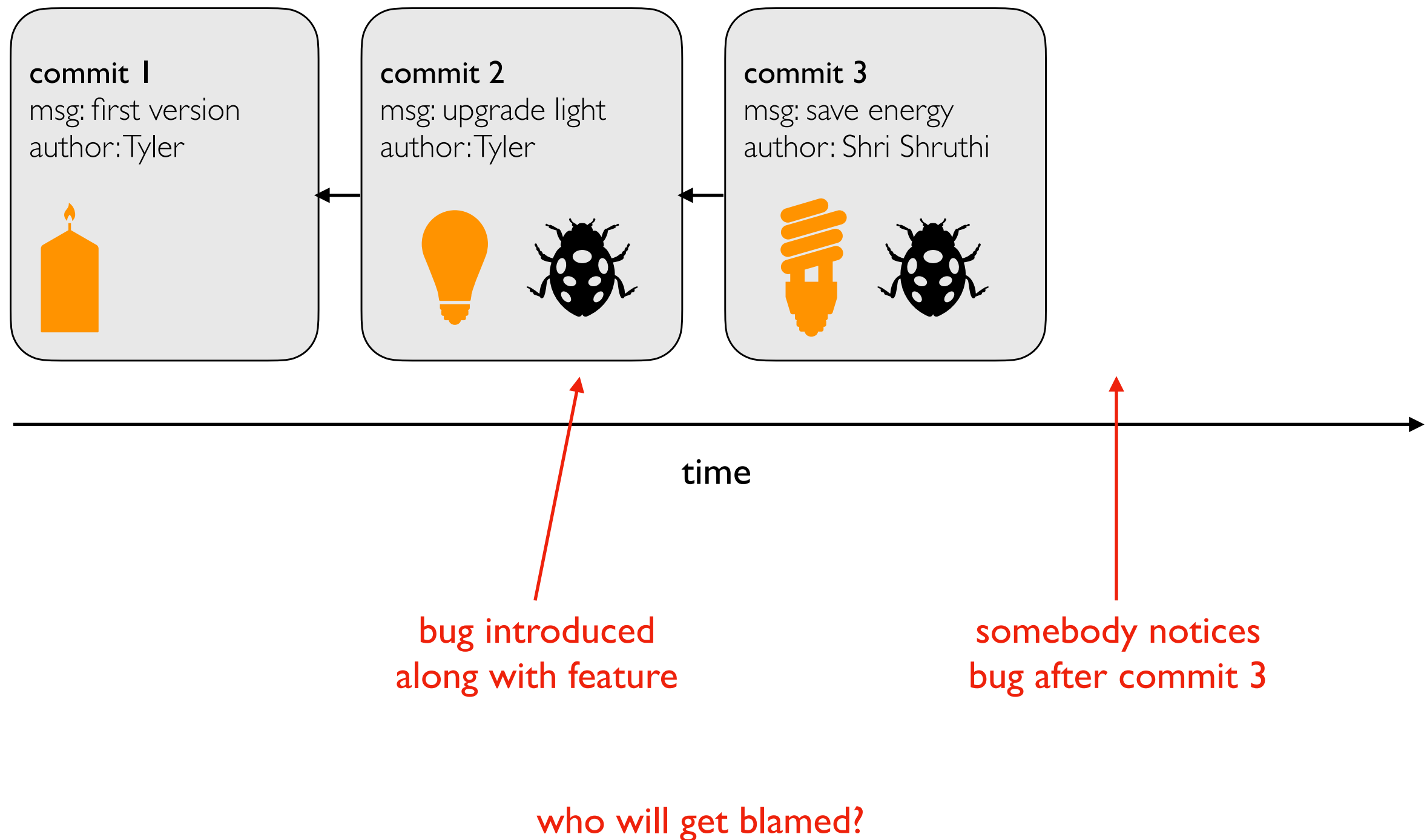


what happens when the plane lands?

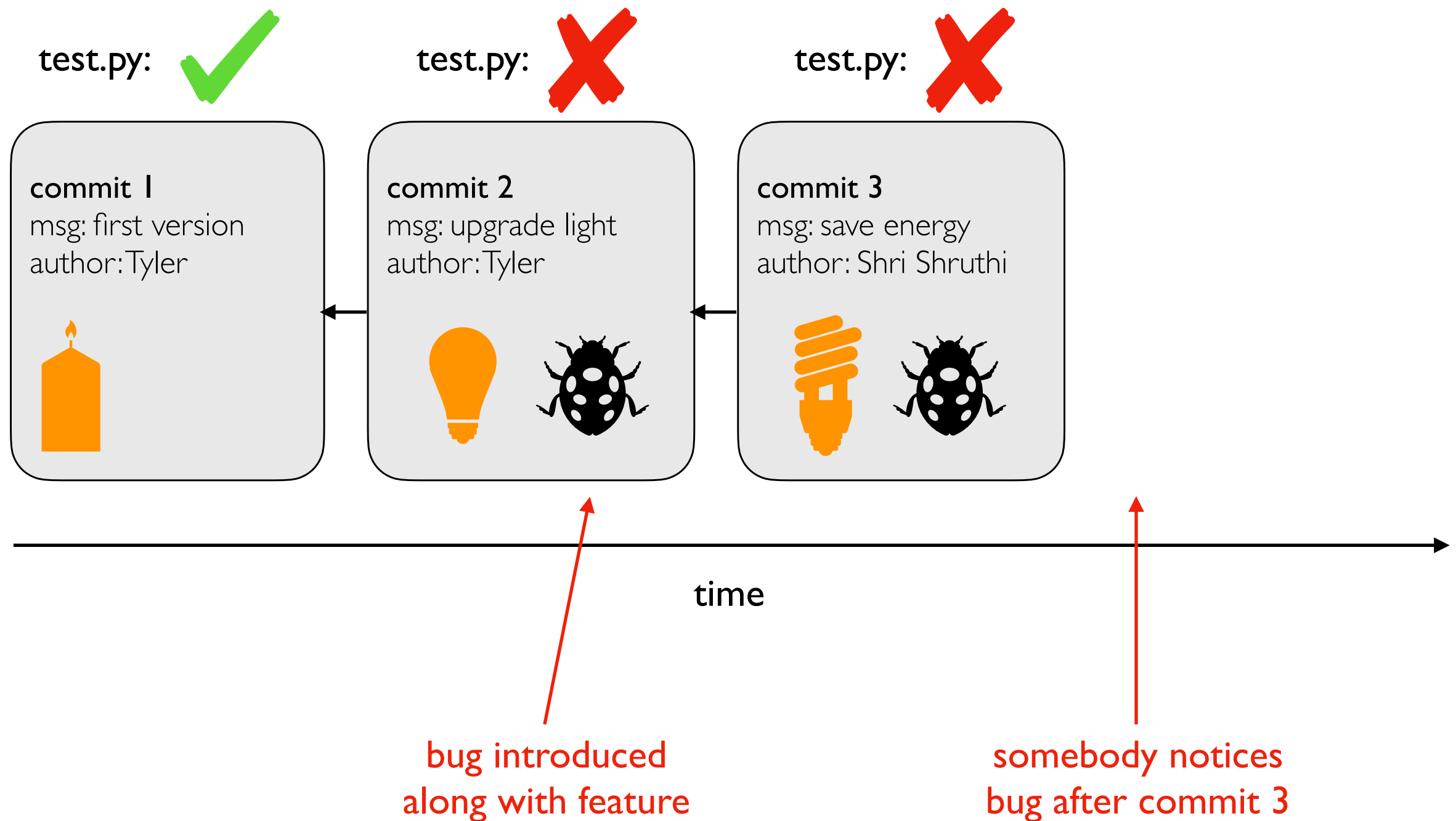
Example



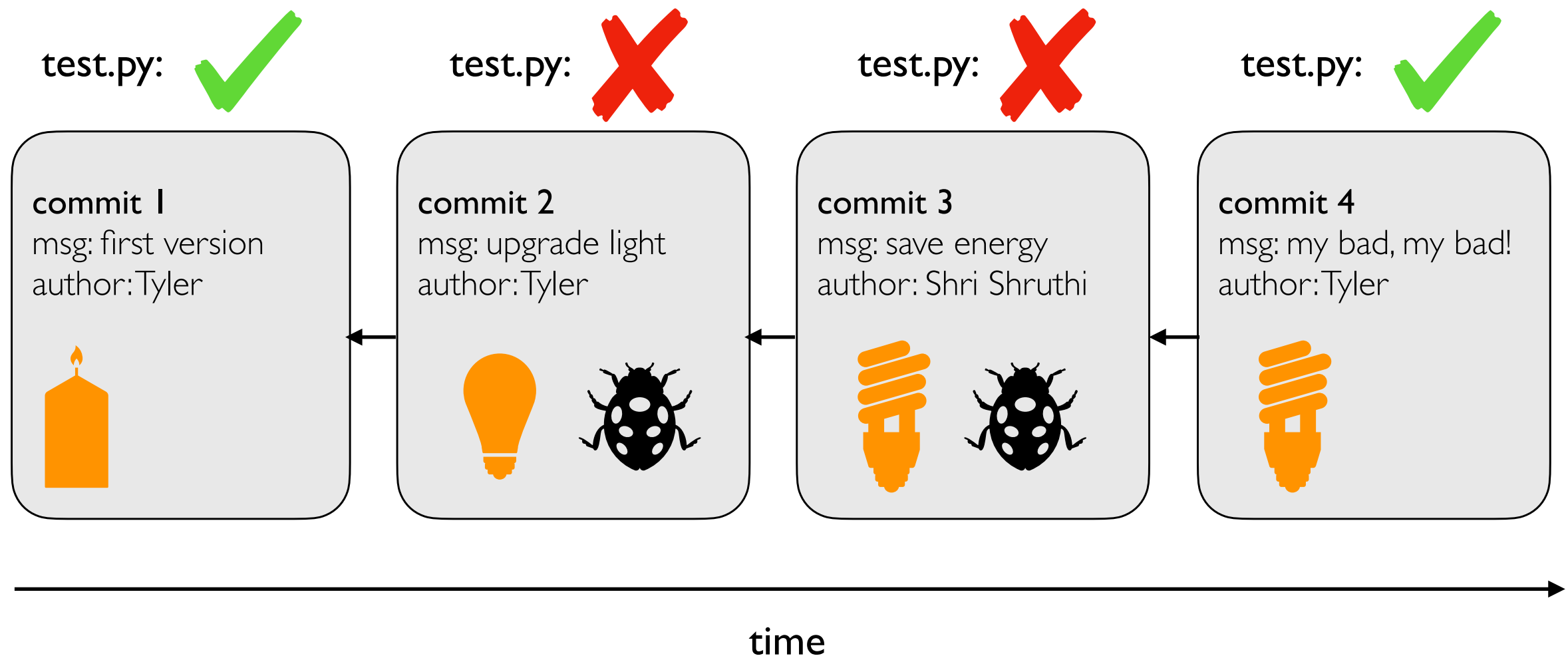
Use case 1: troubleshooting discovered bug



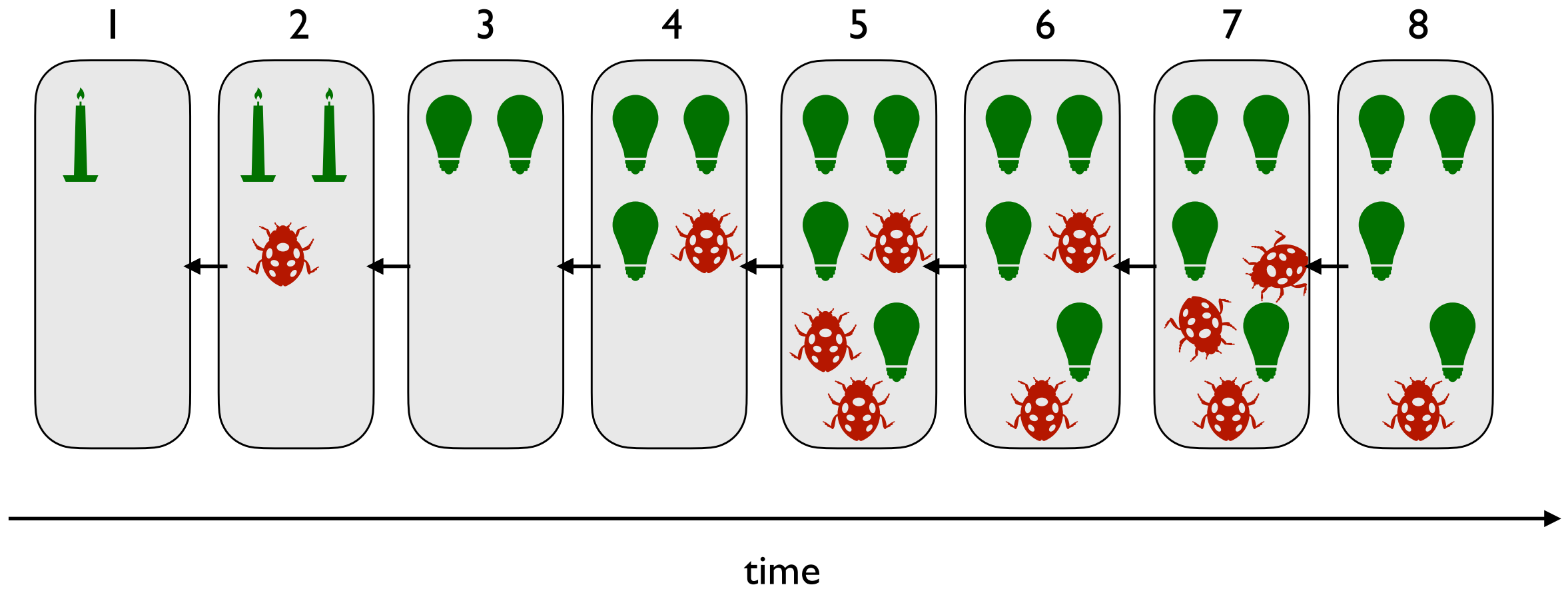
Use case 1: troubleshooting discovered bug



Use case 1: troubleshooting discovered bug

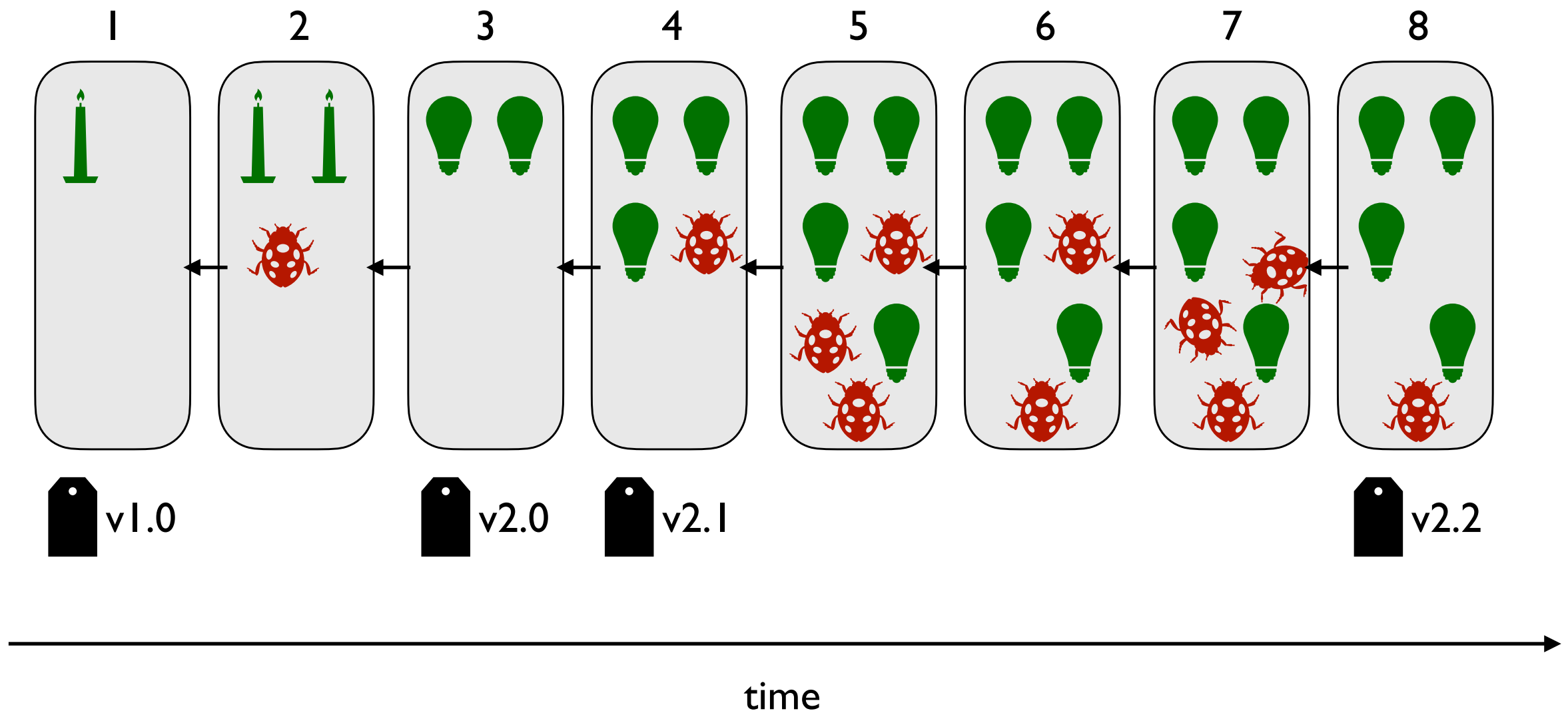


Use case 2: versioned releases



which version would you use?

Use case 2: versioned releases

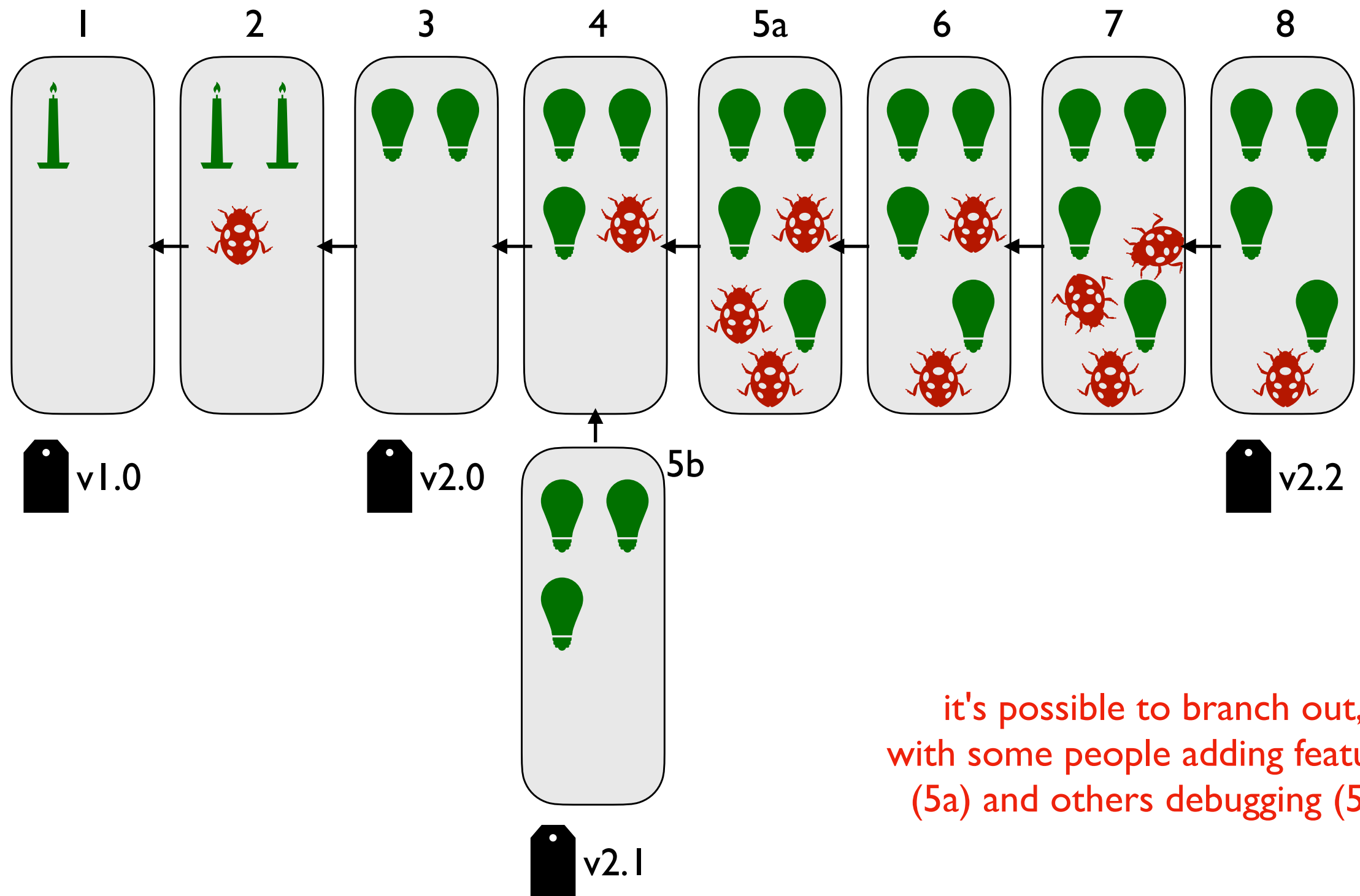


tag "good" commits to create releases

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

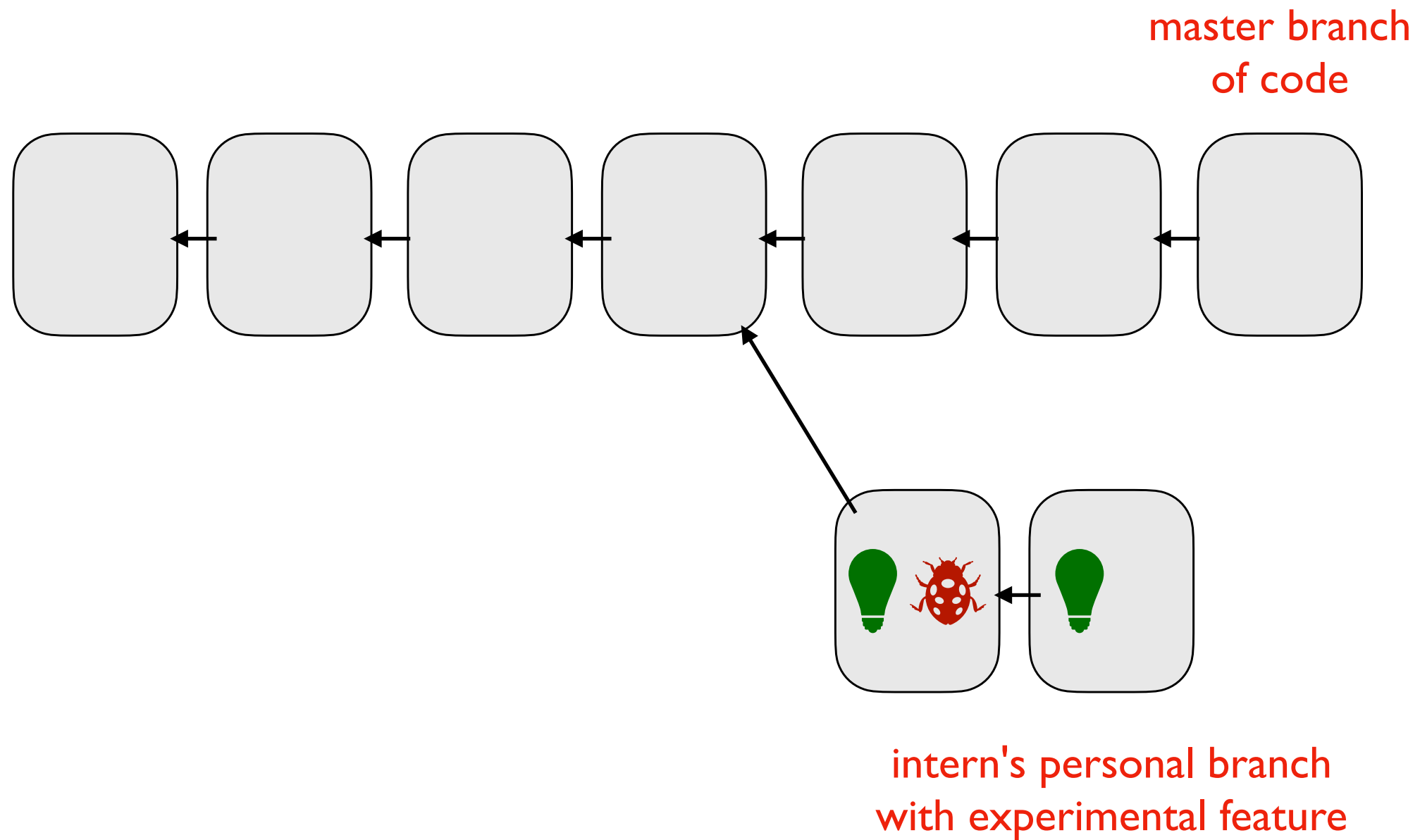
<https://github.com/pandas-dev/pandas/releases>

Use case 2: versioned releases

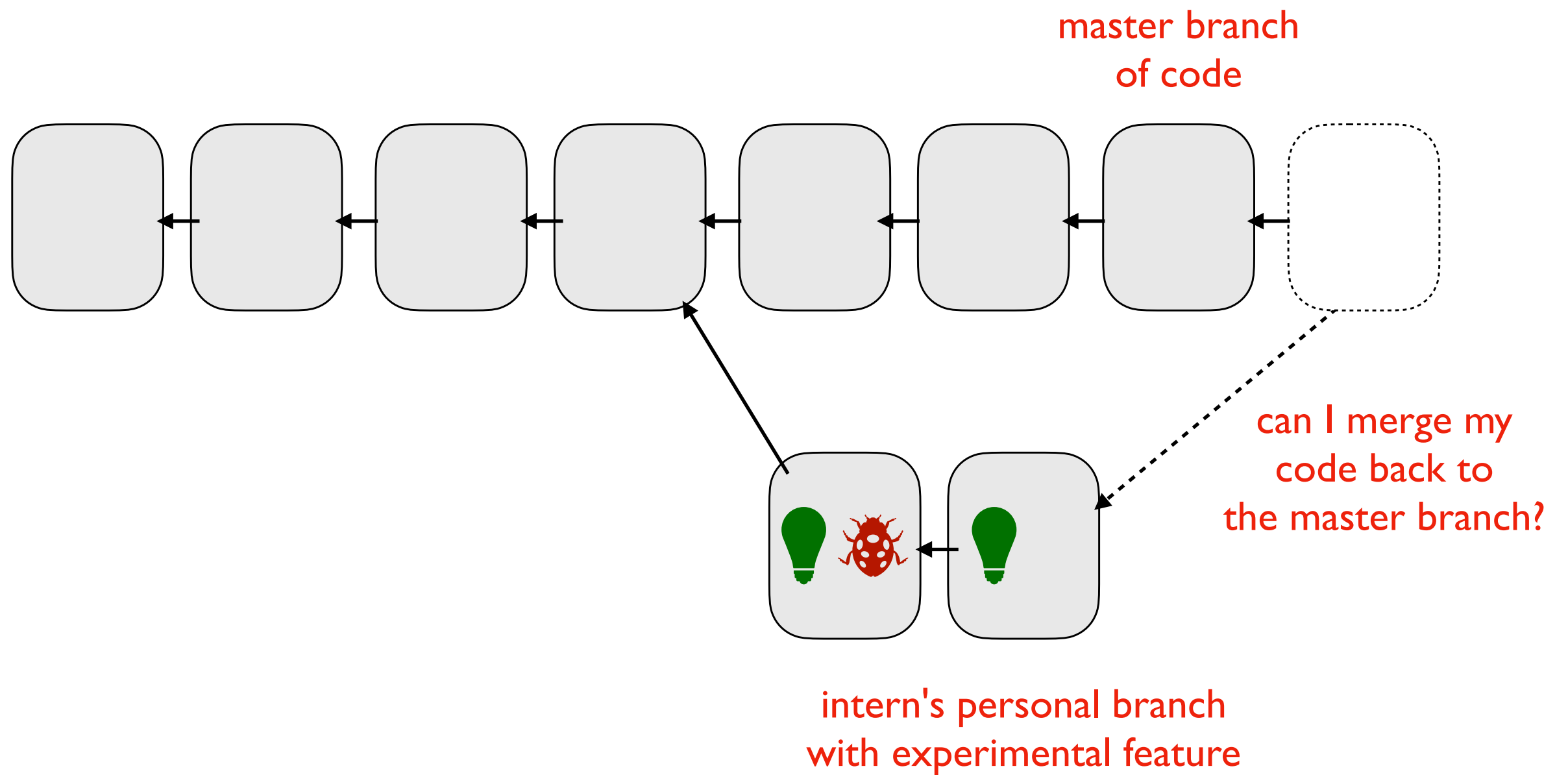


it's possible to branch out,
with some people adding features
(5a) and others debugging (5b)

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

Viewing Commits

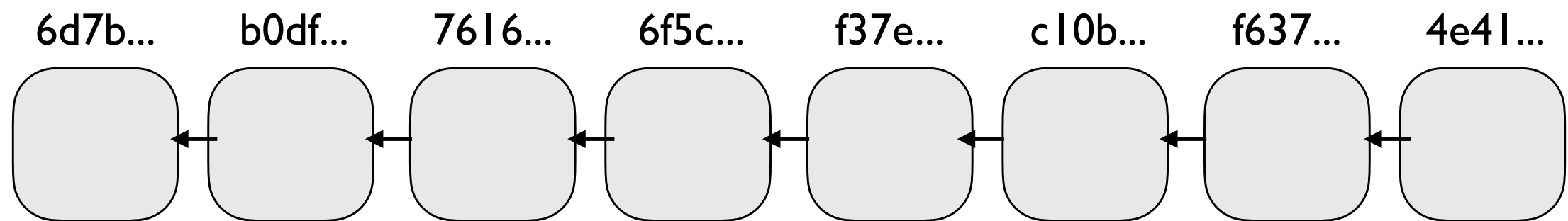
Download P1 repo (<https://github.com/tylerharter/cs320-p1>):

```
git clone https://github.com/tylerharter/cs320-p1.git
cd cs320-p1
```

View Commits (newest on top)

```
git log
```

```
git checkout ??????
```



```
commit 6d7beafb8e79b7a92fed8e67673a33bb7f607dbe
```

```
Author: Ada <ada@example.com>
```

```
Date: Thu Jan 9 13:53:20 2020 -0600
```

commit number in
hexadecimal (hexsha)

committer → count a specific word
message

binary: 0,1

decimal: 0,1,2,3,4,5,6,7,8,9

hex: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

Creating Commits

Configure your name/email

```
git config --global user.name "Tyler"
```

```
git config --global user.name "tharter@wisc.edu"
```

View status of files

```
git status
```

Move file to **staging**

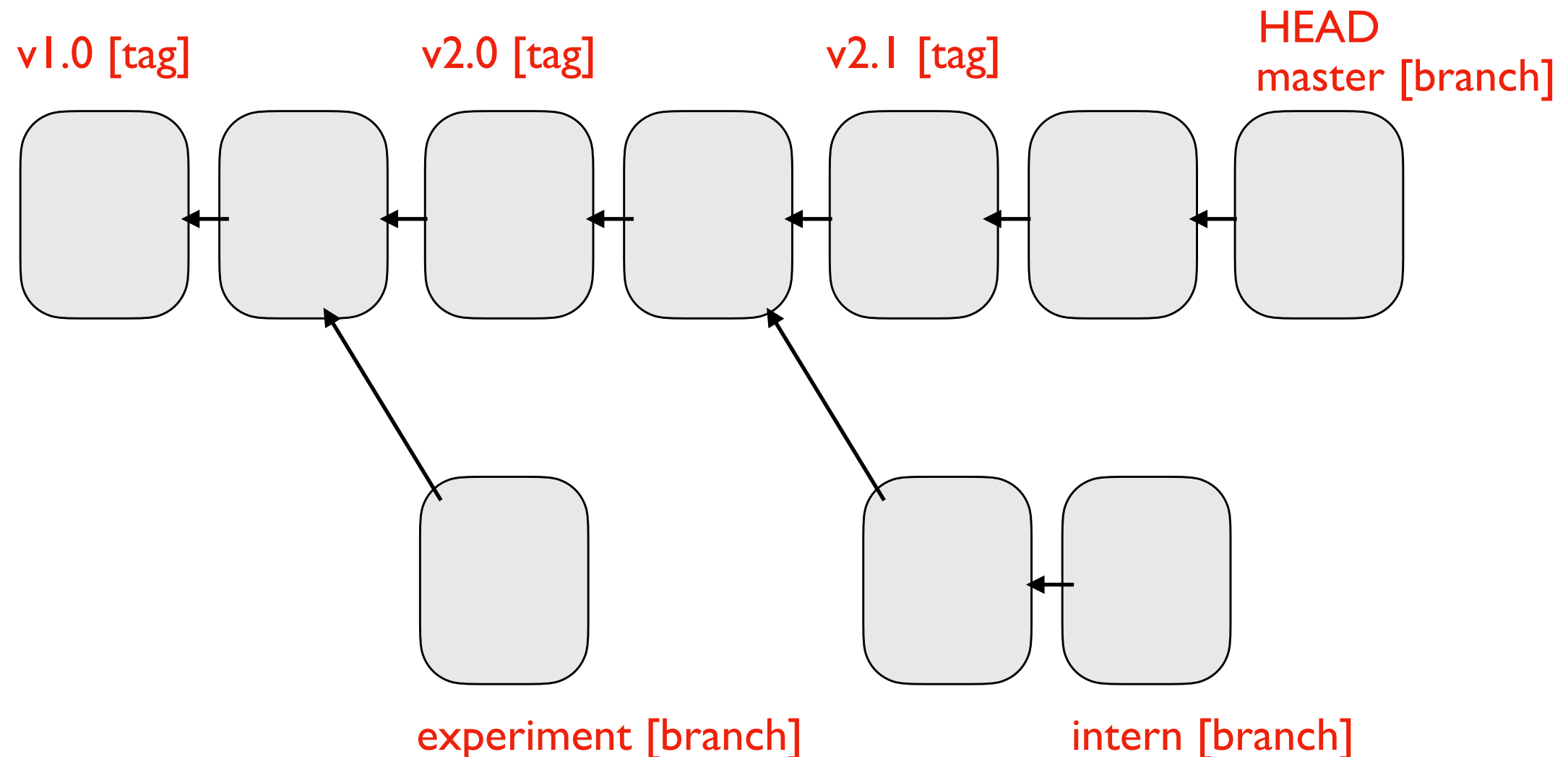
```
git add file.txt
```

Create a commit (take a snapshot of **staged** changes)

```
git commit -m "I made a change!"
```

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)

HEAD, Branches, and Tags

What branch are we on?

```
git branch
```

Create new branch

```
git branch branchname
```

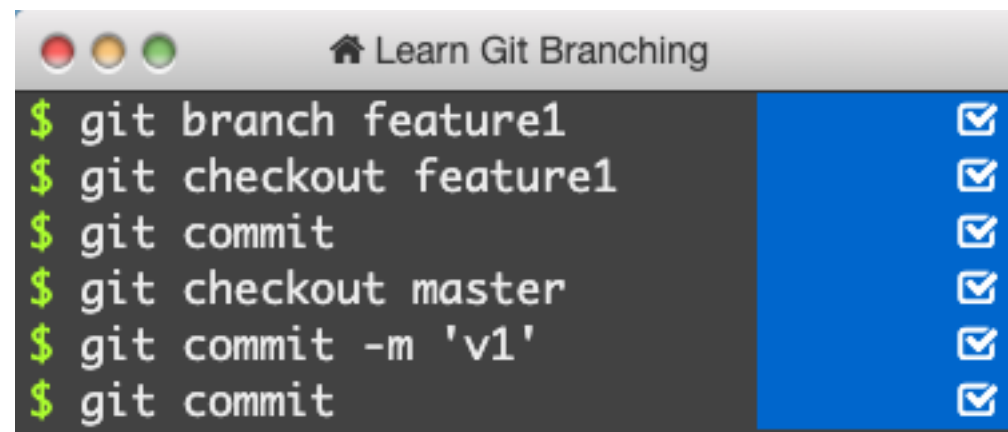
Switch branch

```
git checkout branchname
```

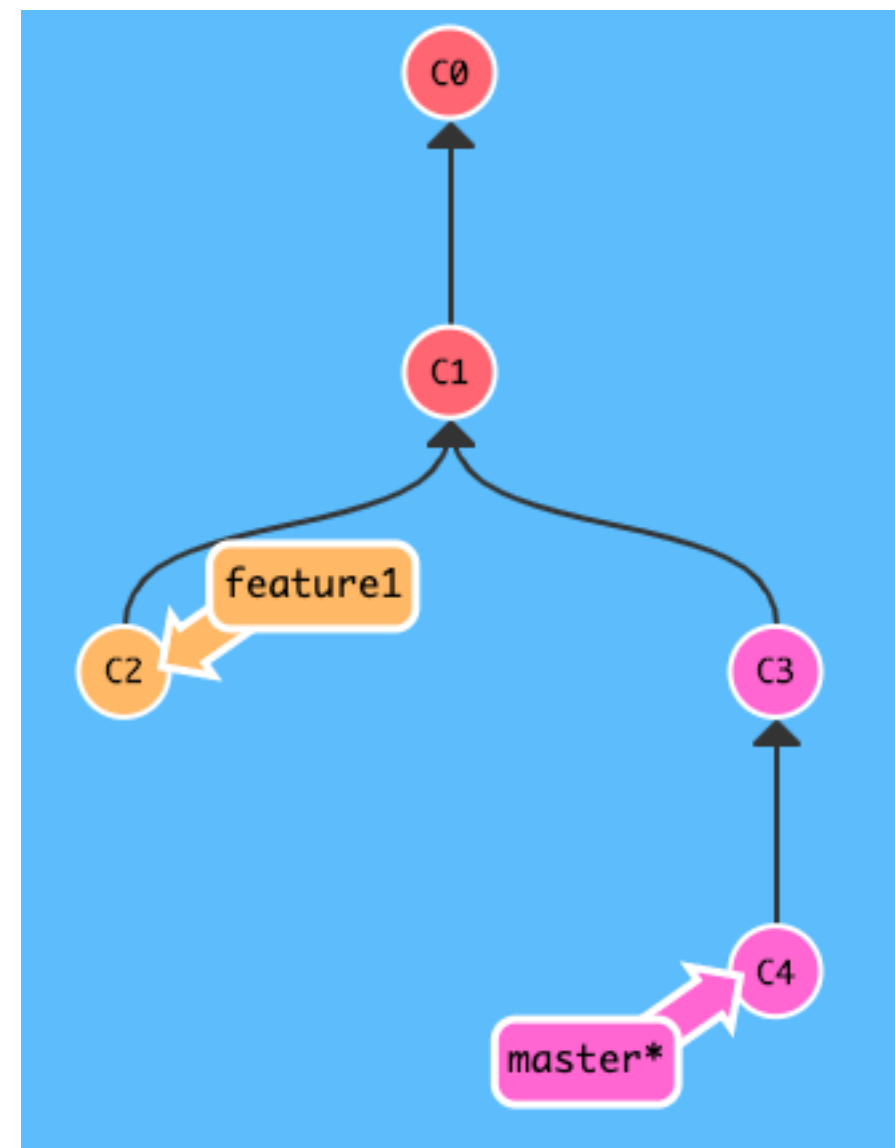
Practice Branching

Git equivalent of PythonTutor:

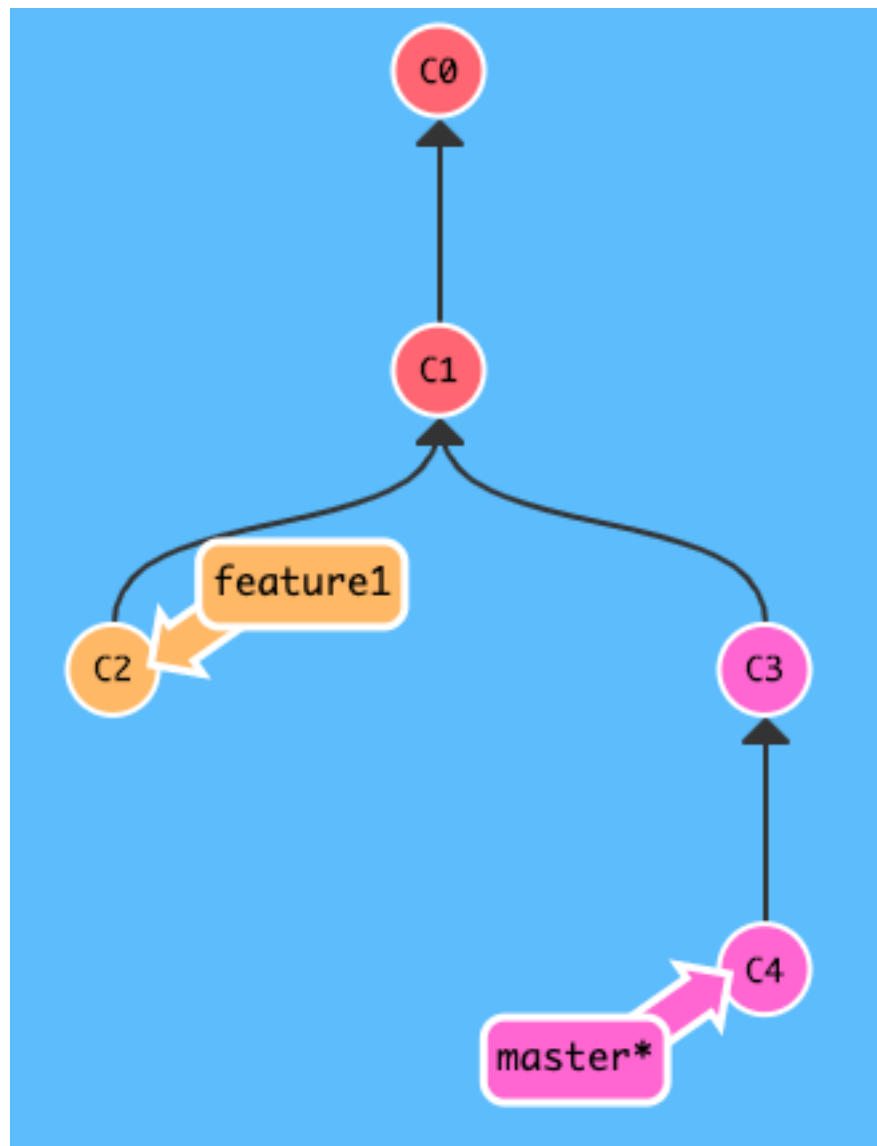
<https://learngitbranching.js.org/?NODEMO>



```
$ git branch feature1  
$ git checkout feature1  
$ git commit  
$ git checkout master  
$ git commit -m 'v1'  
$ git commit
```



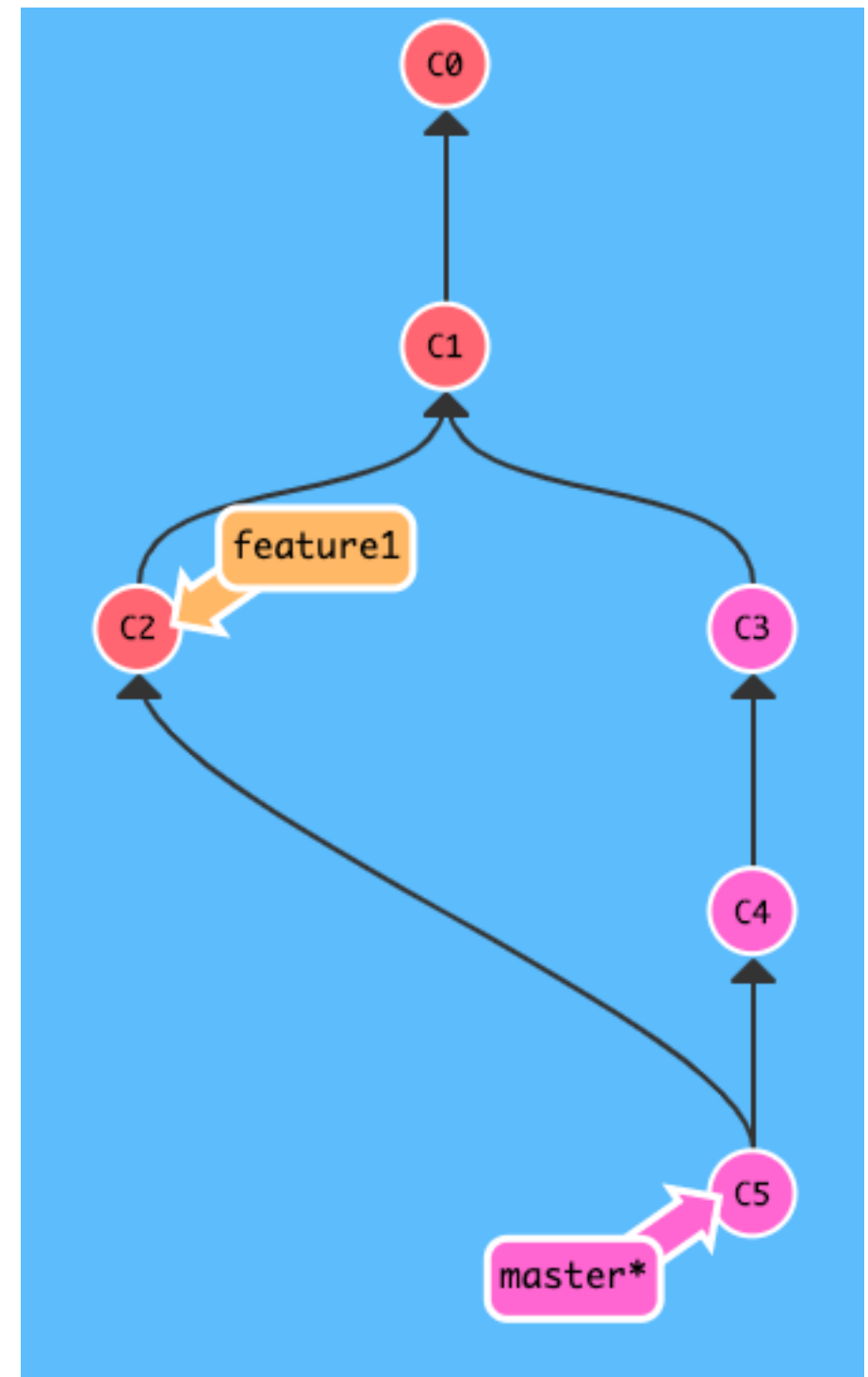
Merging without Conflicts



Switch branch

`git merge frombranch`

add whatever is there to the current branch



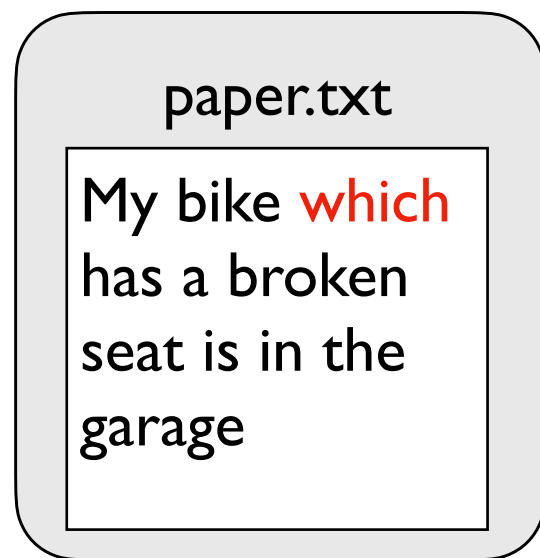
tip (or learn vim):

`export EDITOR=nano`

Merging with Conflicts

What happens when two people try to fix the same issue, in two different (incompatible) ways?

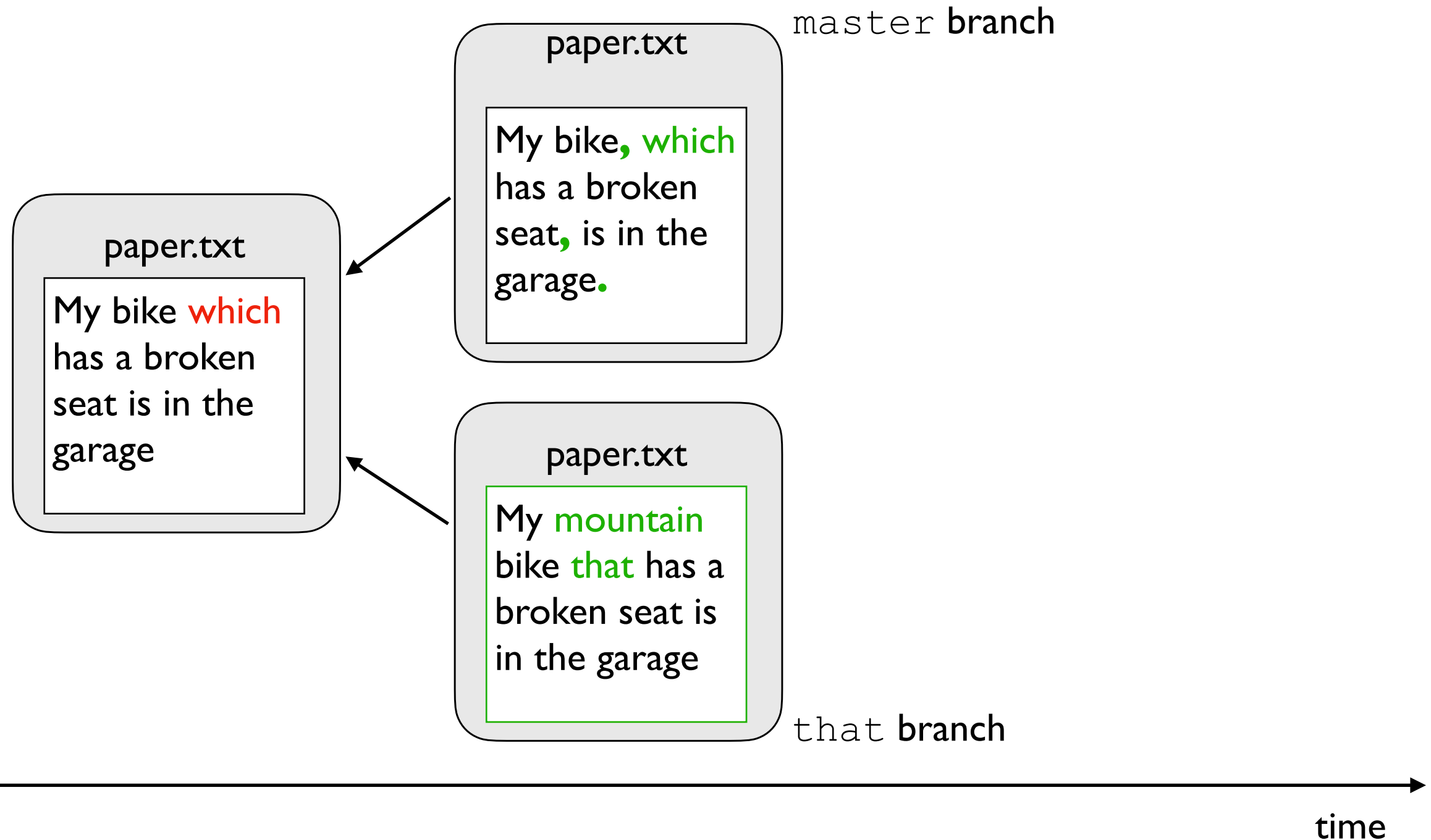
master branch



time

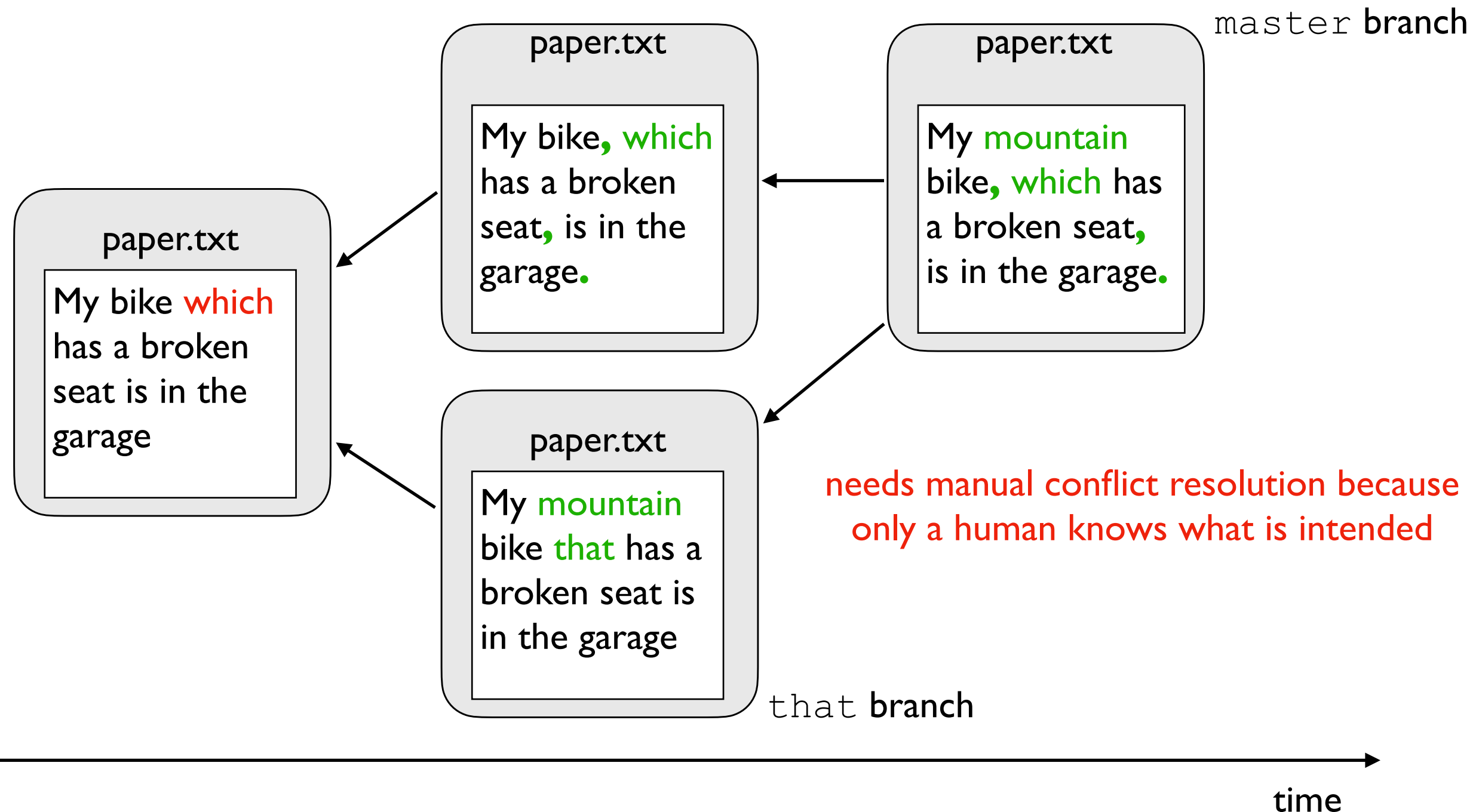
Merging with Conflicts

What happens when two people try to fix the same issue, in two different (incompatible) ways?



Merging with Conflicts

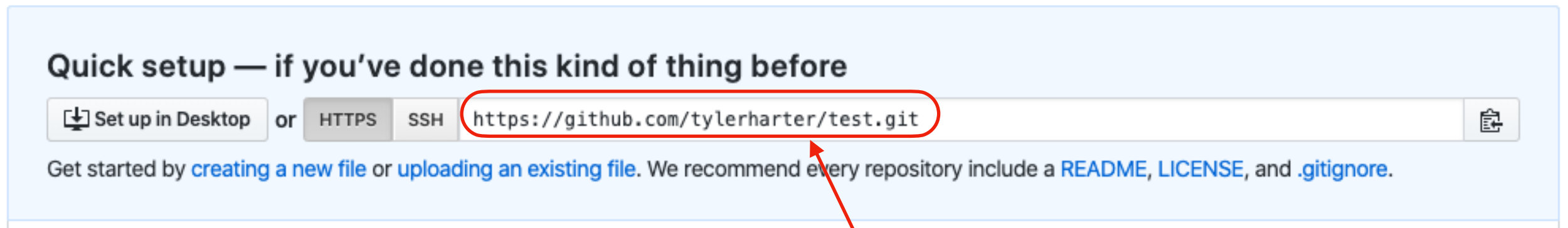
What happens when two people try to fix the same issue, in two different (incompatible) ways?



Remotes

We will often want to work on our laptops, but also have our repositories on GitHub (or similar)

Create GitHub account, go here: <https://github.com/new>



Pushing a branch to GitHub

```
git remote add github https://github.com/tylerharter/test.git
```

```
git push github master
```

remote
name

branch
name

see reading about naming in software

Summary of Terms

commit: a snapshot of files at a point in time

HEAD: a convenient label for the current commit

tag: a long-term label associated with a commit

branch: a label attached to a commit that re-attaches to new commits

merge: to combine changes on another branch into the current branch

conflict: differences that cannot automatically be merged

Challenges: <https://learngitbranching.js.org/?NODEMO>

