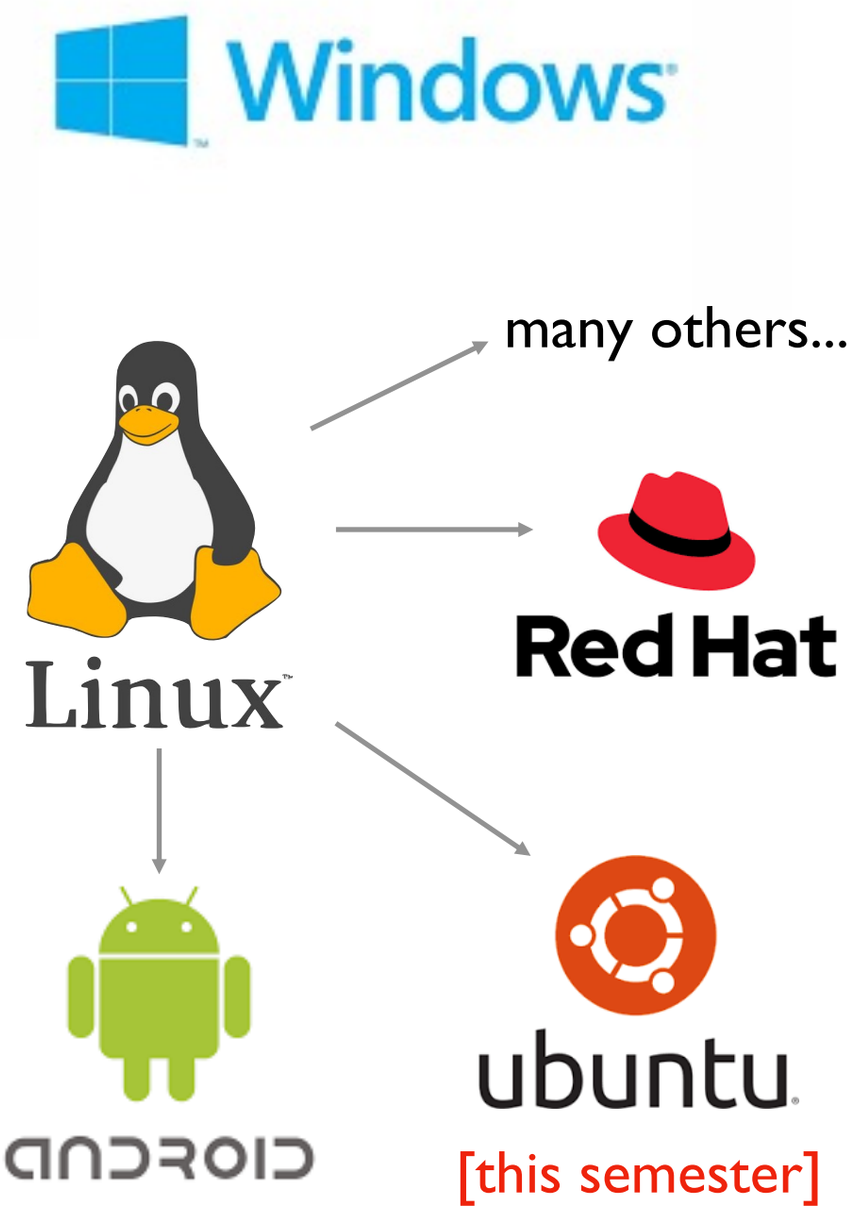
[320] Reproducibility 2

Meenakshi Syamkumar

Big question: *will my program run on someone else's computer?*
(not necessarily written in Python)

Things to match:

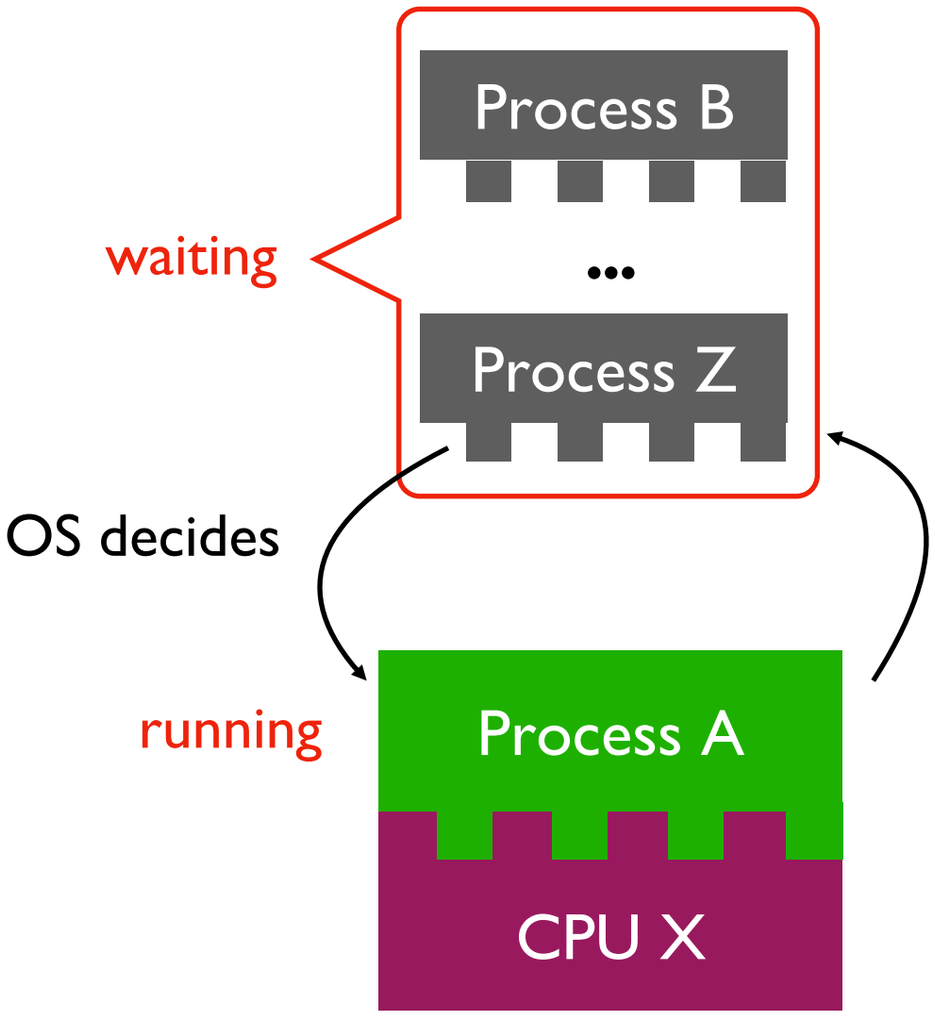
1. Hardware
2. Operating System
3. Dependencies



OS jobs: Allocate and Abstract Resources

[like CPU, hard drive, etc]

1 Allocation



only one process can run on CPU at a time
(or a few things if the CPU has multiple "cores")

2 Abstraction

```
f = open("file.txt")
data = f.read()
f.close()
```

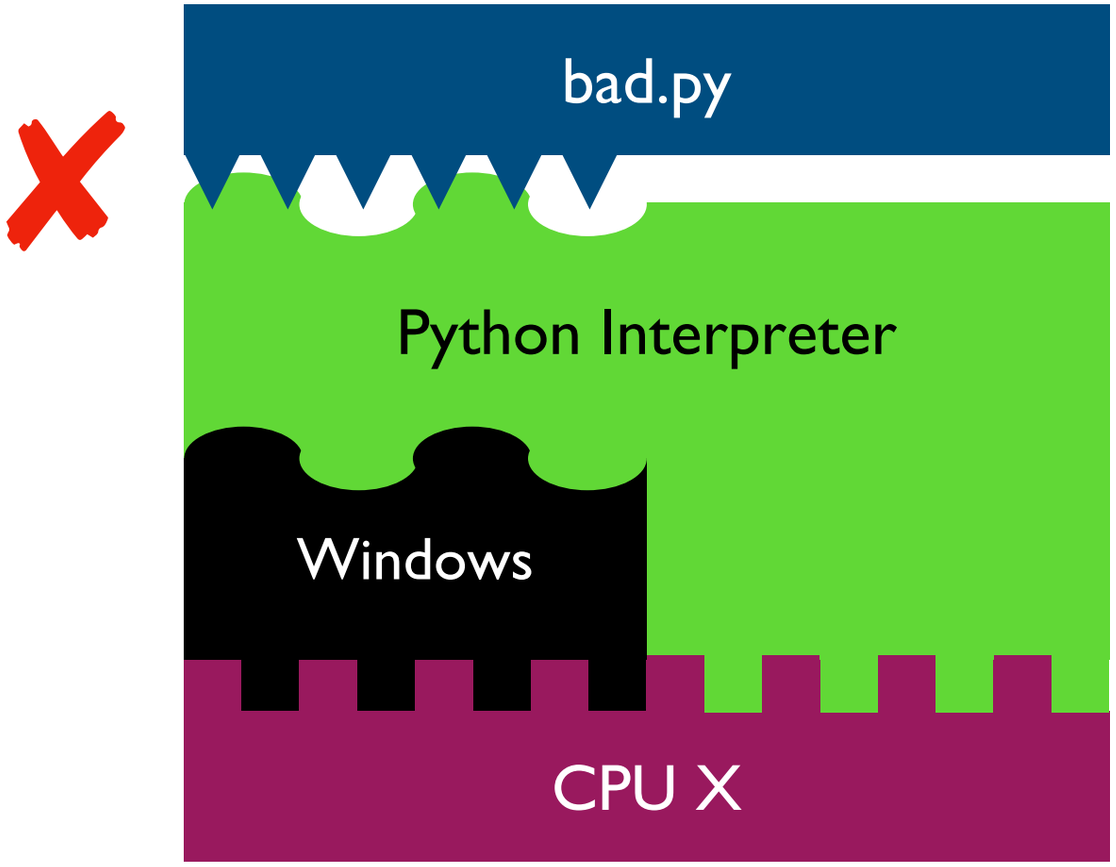
convenient

Operating System

inconvenient

ignorant of files/directories

Harder to reproduce on different OS...

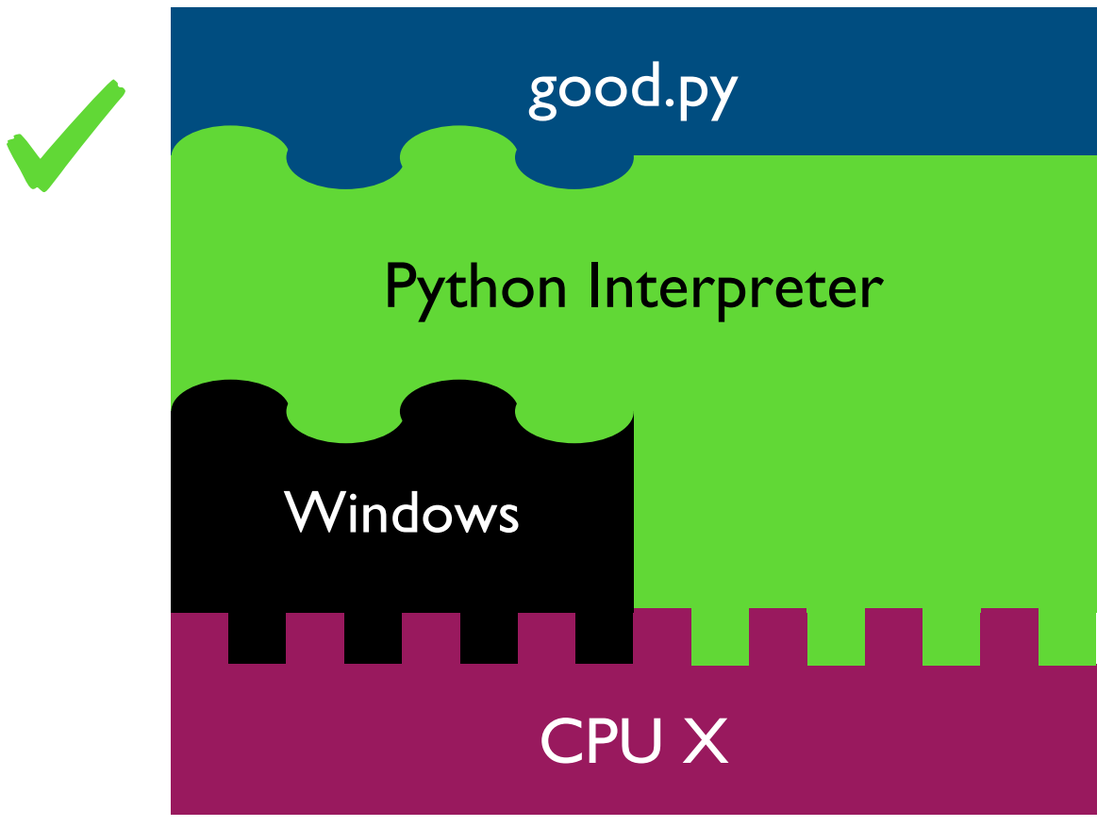



```
f = open("/data/file.txt")
...
```

The Python interpreter mostly lets you
[Python Programmer] ignore the CPU you run on.

But you still need to work a bit to "fit" the code to the OS.

Harder to reproduce on different OS...

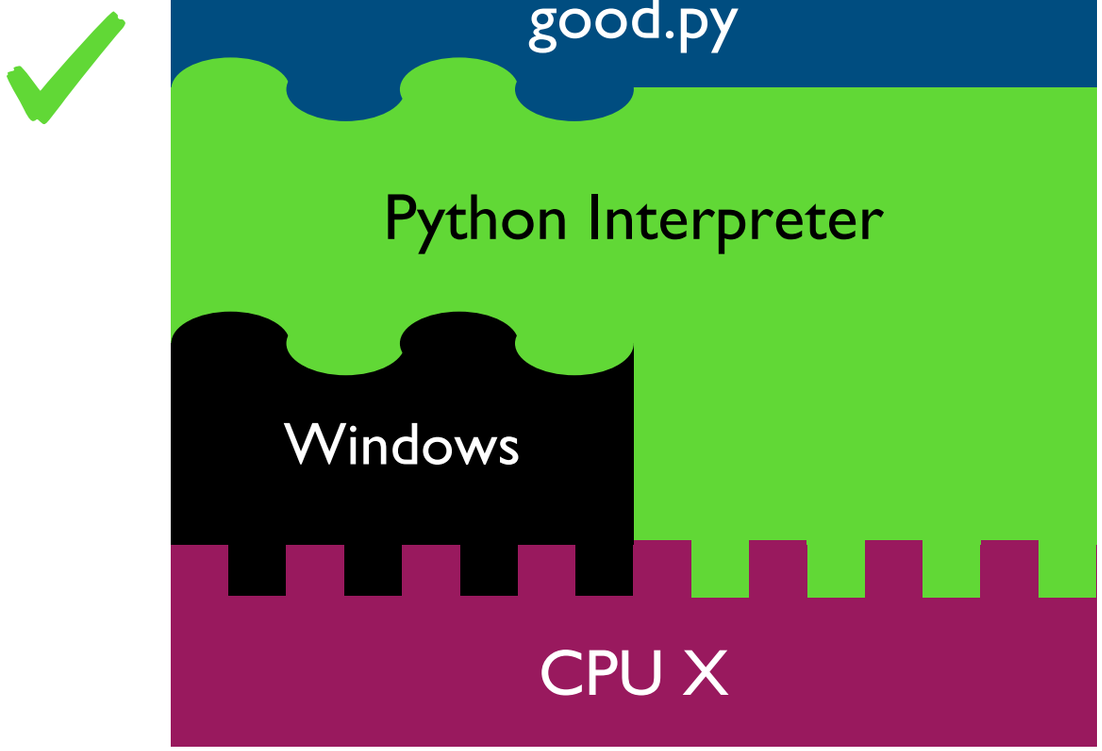



```
f = open("c:\data\file.txt")
...
```

The Python interpreter mostly lets you
[Python Programmer] ignore the CPU you run on.

But you still need to work a bit to "fit" the code to the OS.

Harder to reproduce on different OS...



solution 1:

```
f = open(os.path.join("data", "file.txt"))
...
```

solution 2:

tell anybody reproducing your results to use the same OS!

tradeoffs?

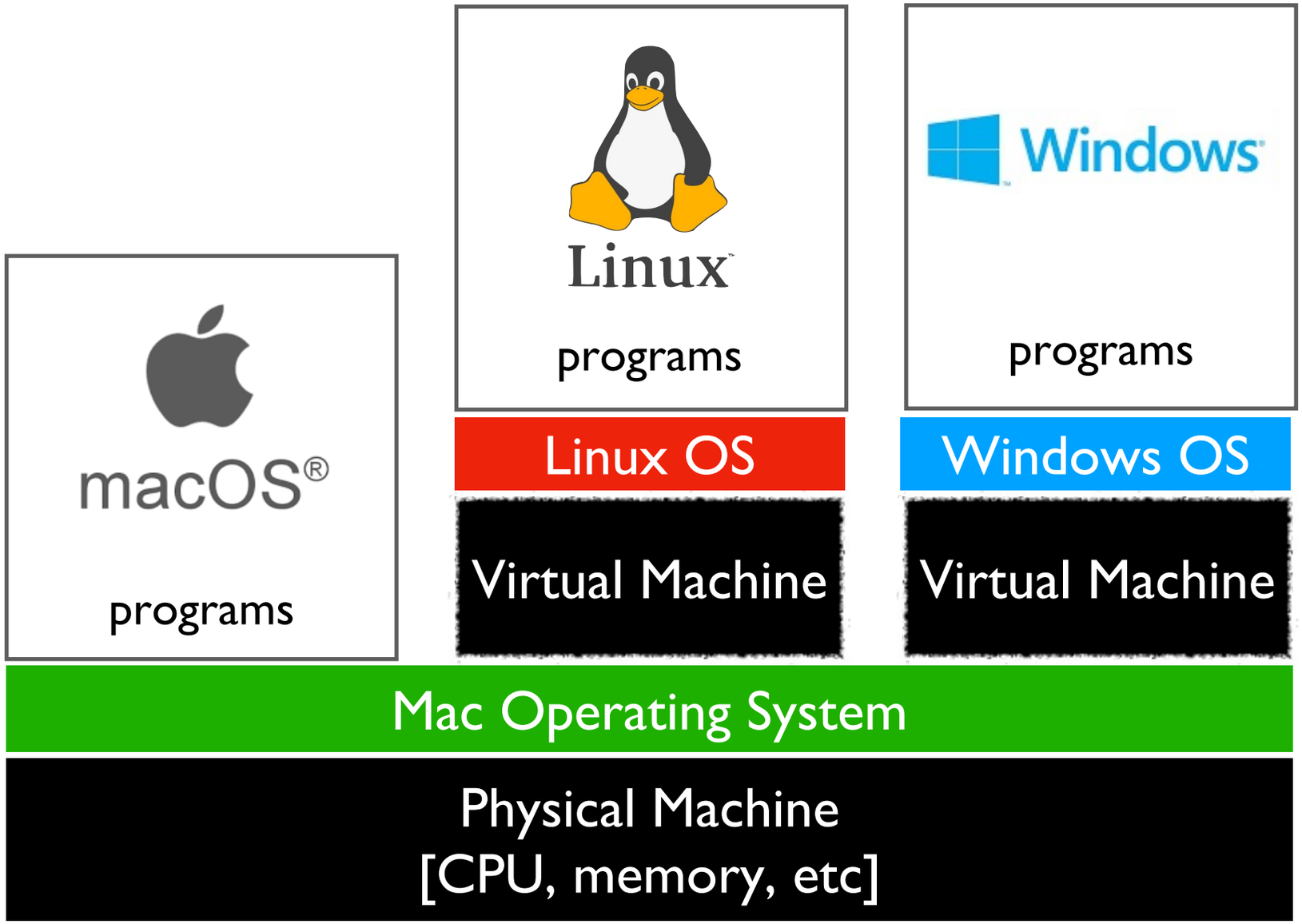
The Python interpreter mostly lets you
[Python Programmer] ignore the CPU you run on.

But you still need to work a bit to "fit" the code to the OS.

VMs (Virtual Machines)

popular virtual machine software

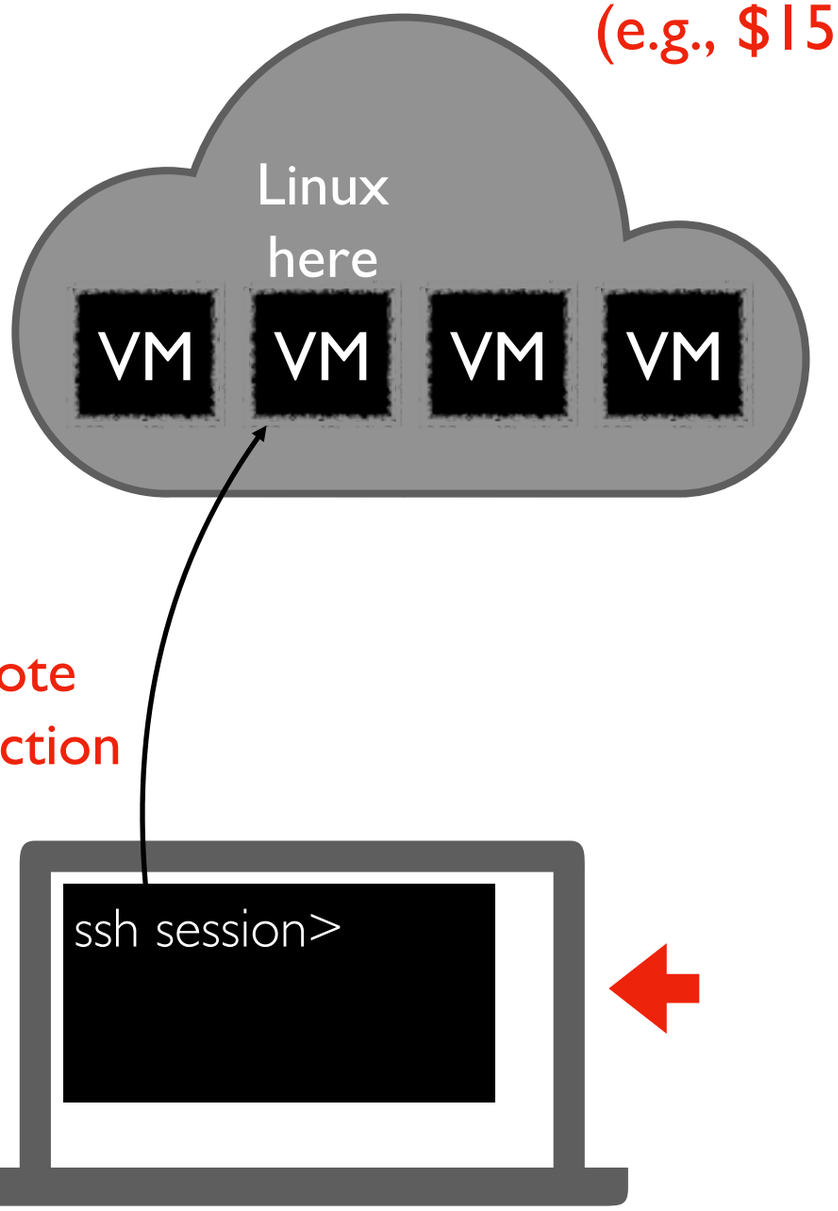
Microsoft Hyper-V



With the right virtual machines created and operating systems installed, you could run programs for Mac, Linux, and Windows -- at the same time without rebooting!

The Cloud

cloud providers let you rent VMs
in the cloud on hourly basis
(e.g., $15 / month)



Windows, Mac,
whatever

`ssh user@best-linux.cs.wisc.edu`

run in
PowerShell/bash to
access CS lab

popular cloud providers

amazon web services™

Microsoft Azure

Google Cloud Platform

we'll use GCP virtual
machines this semester
[setup in lab]

https://docs.microsoft.com/en-us/windows-server/administration/openssh/openssh_install_firstuse

Lecture Recap: 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 ← next time: versioning

Recap of 15 new terms

reproducibility: others can run our analysis code and get same results

process: a running program

byte: integer between 0 and 255

address space: a big "list" of bytes, per process, for all state

address: index in the big list

encoding: pairing of ~~letters~~ characters with numeric codes

CPU: chip that executes instructions, tracks position in code

instruction set: pairing of CPU instructions/ops with numeric codes

operating system: software that allocates+abstracts resources

resource: time on CPU, space in memory, space on SSD, etc

allocation: the giving of a resource to a process

abstraction: hiding inconvenient details with something easier to use

virtual machine: "fake" machine running on ~~real~~ physical machine
allows us to run additional operating systems

cloud: place where you can rent virtual machines and other services

ssh: secure shell -- tool that lets you remotely access another machine

[320] Version Control (git)

Meenakshi Syamkumar

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
```

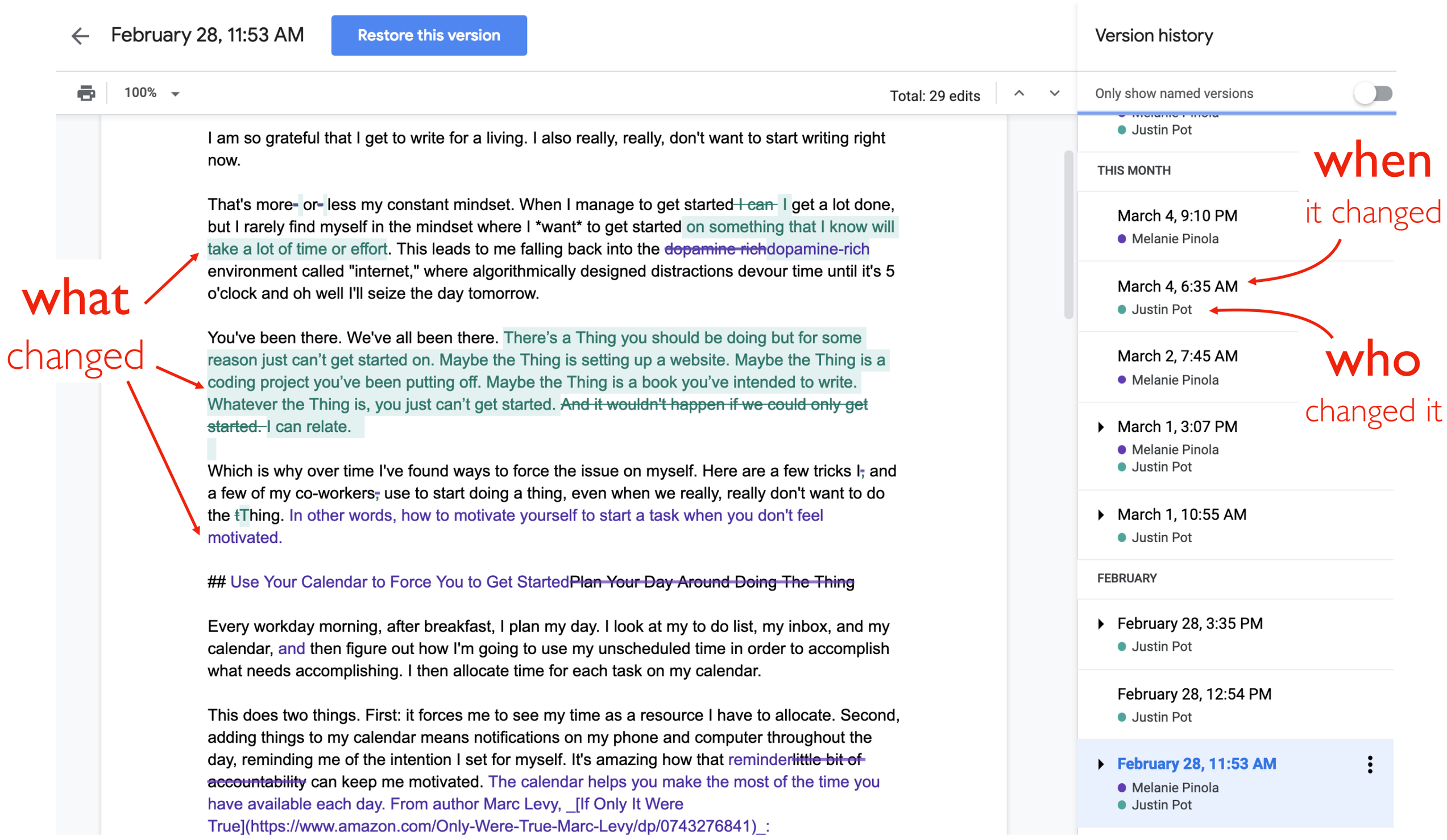
or

```
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*

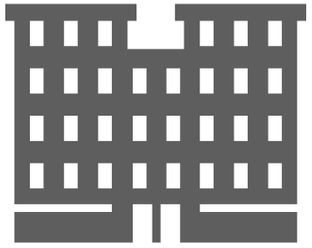
- users **manually** run commands to preserve good versions

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

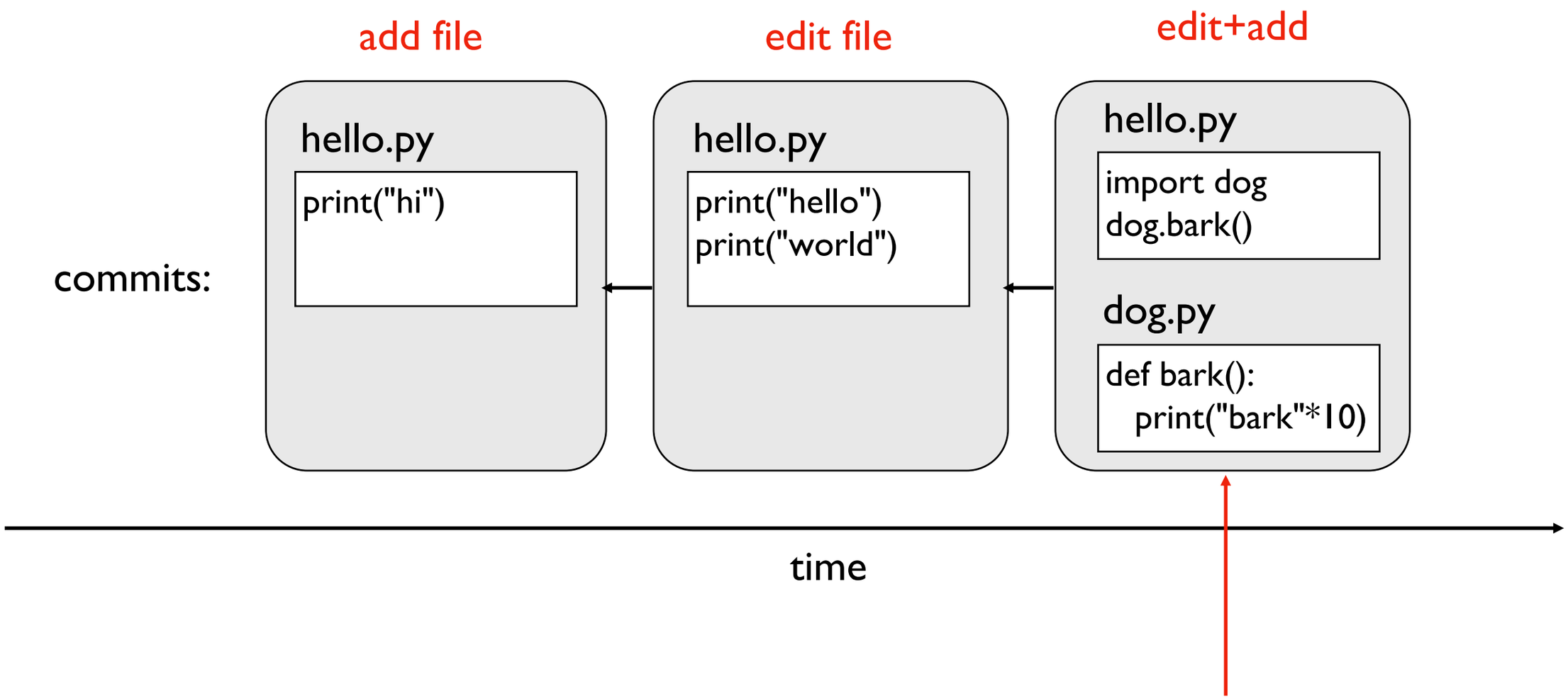


partner A working on `hw.py` at school

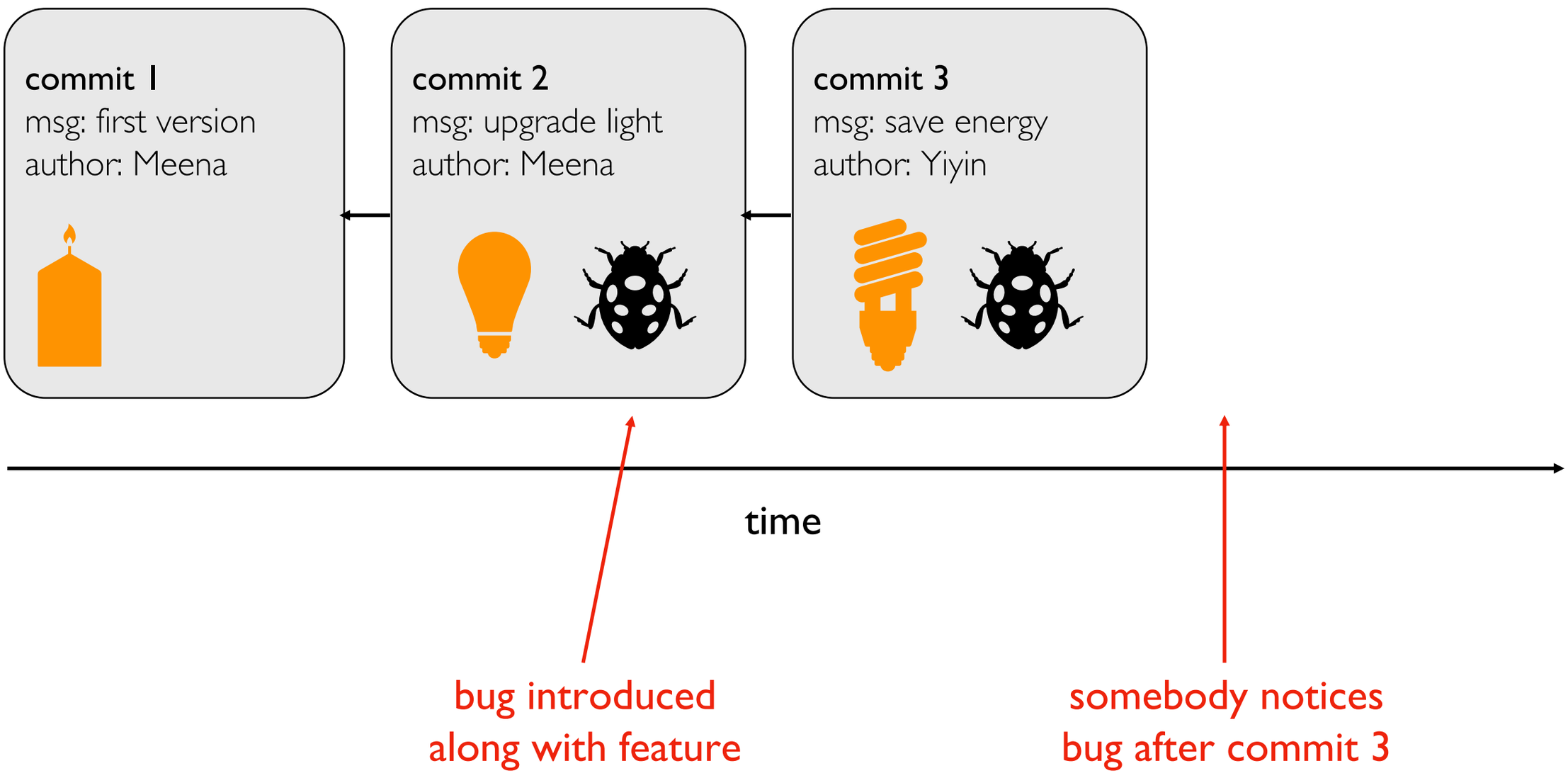
partner B also working on `hw.py`, without wifi

what happens when the plane lands?

Example

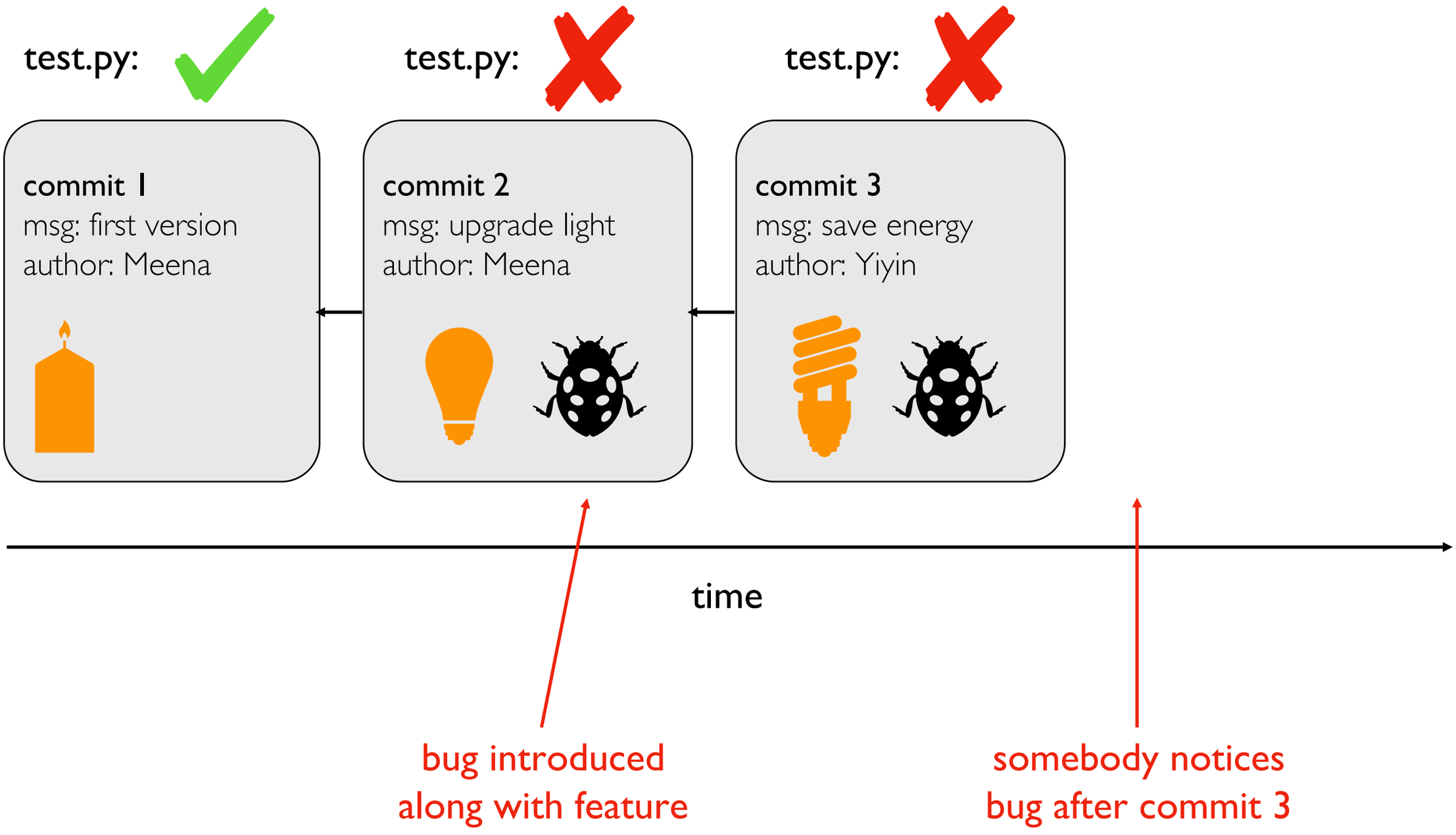

add file
edit file
edit+add
commits:
hello.py
print("hi")
hello.py
print("hello")
print("world")
hello.py
import dog
dog.bark()
dog.py
def bark():
print("bark"*10)
time
at any point in time,
you just see one version
of the files on your computer

Use case 1: troubleshooting discovered bug

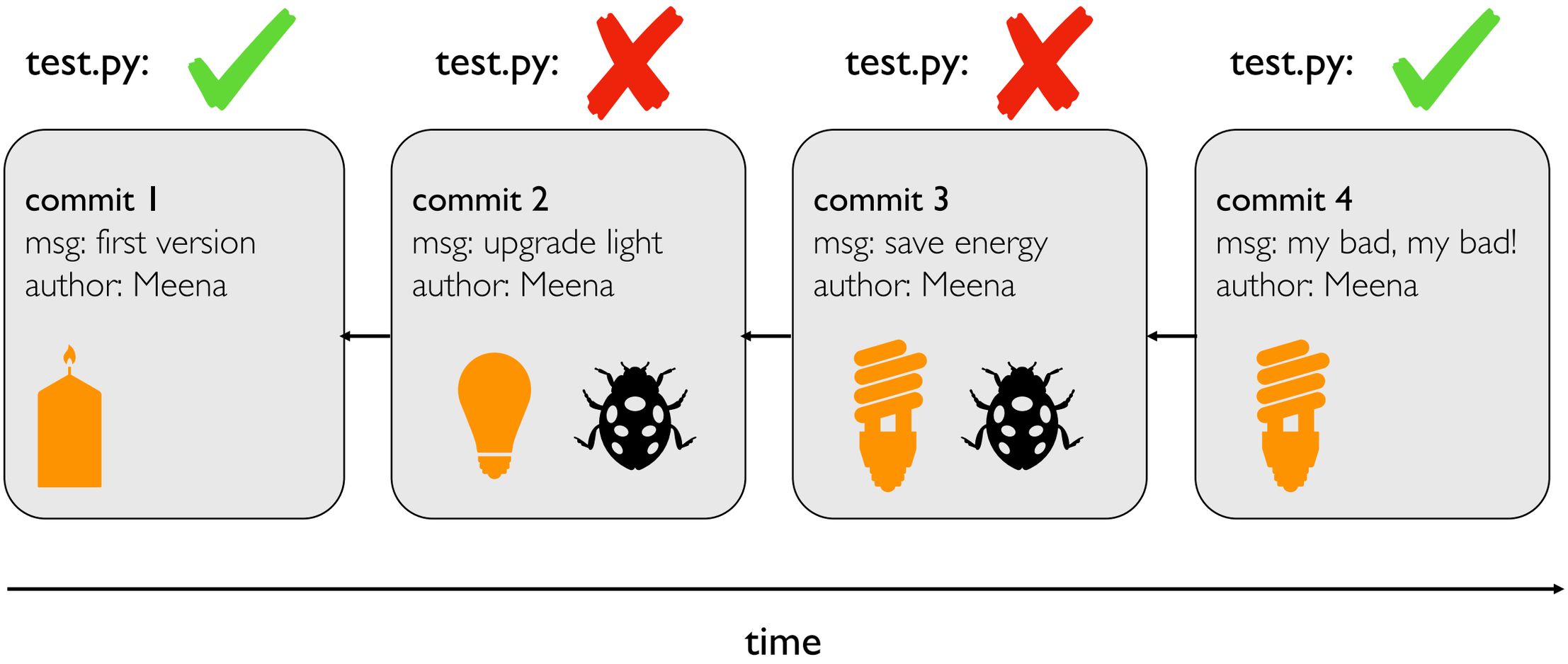


who will get blamed?

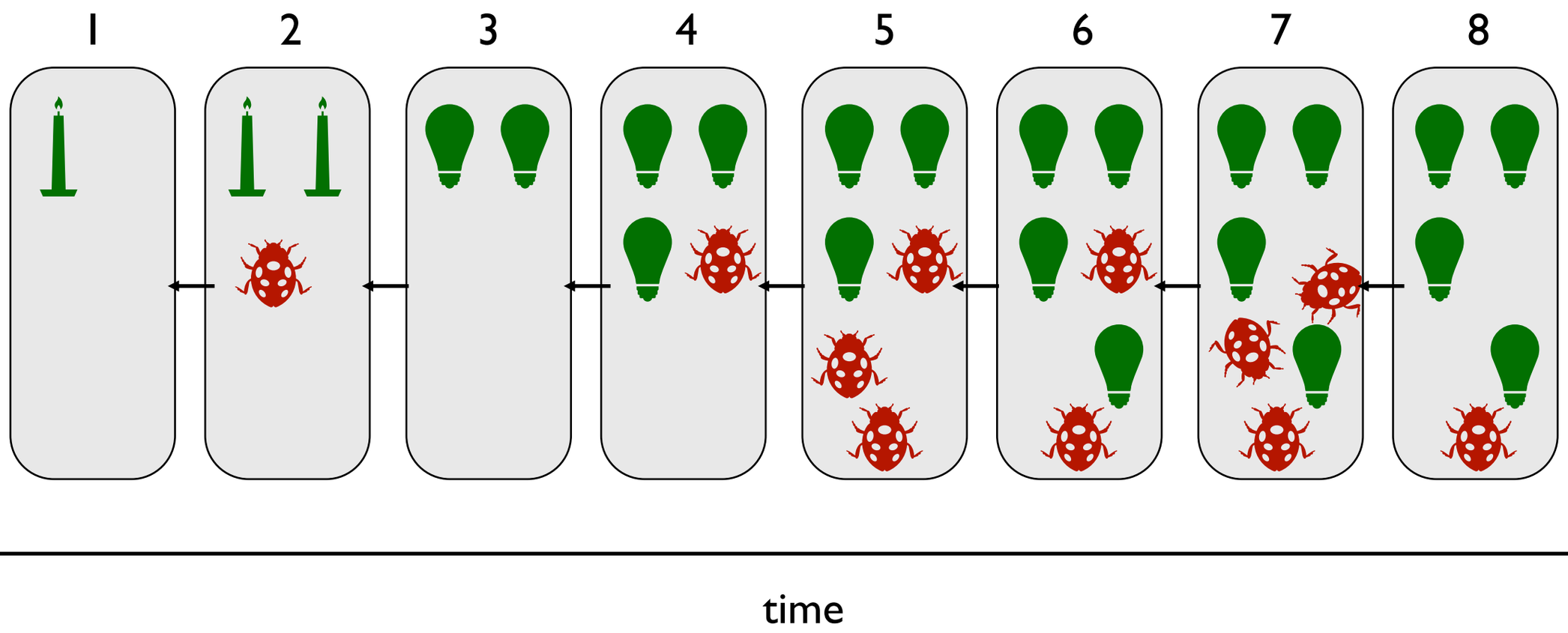
Use case 1: troubleshooting discovered bug



Use case 1: troubleshooting discovered bug

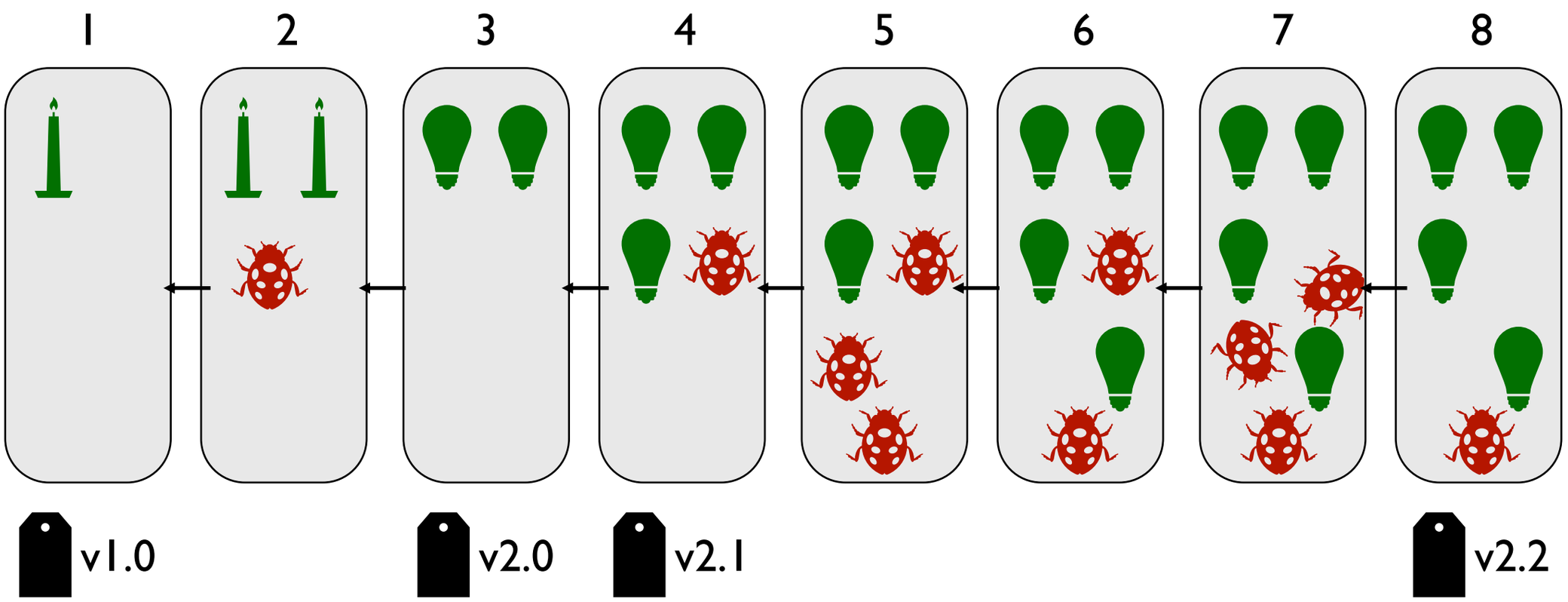


Use case 2: versioned releases



which version would you use?

Use case 2: versioned releases

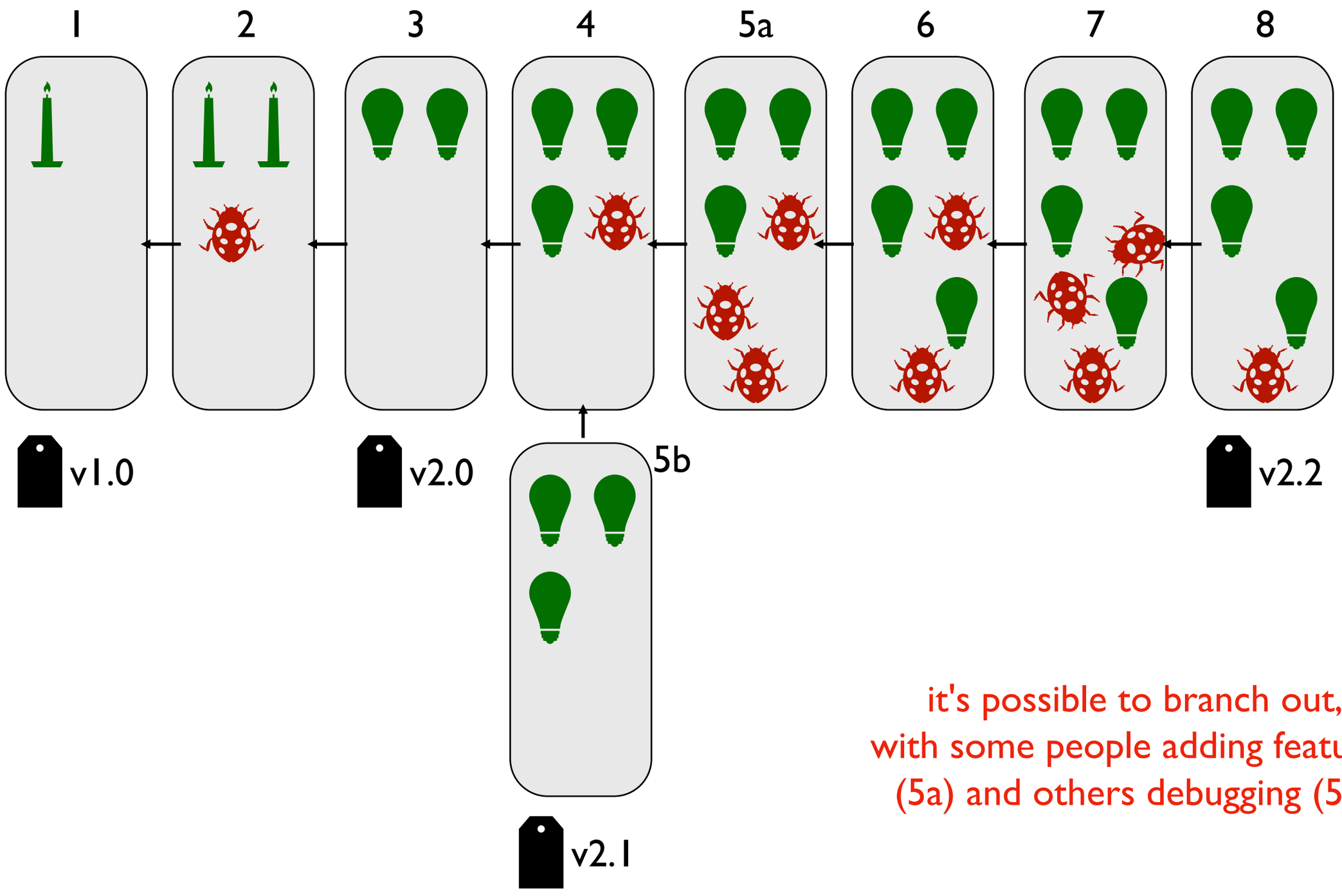


tag "good" commits to create releases

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

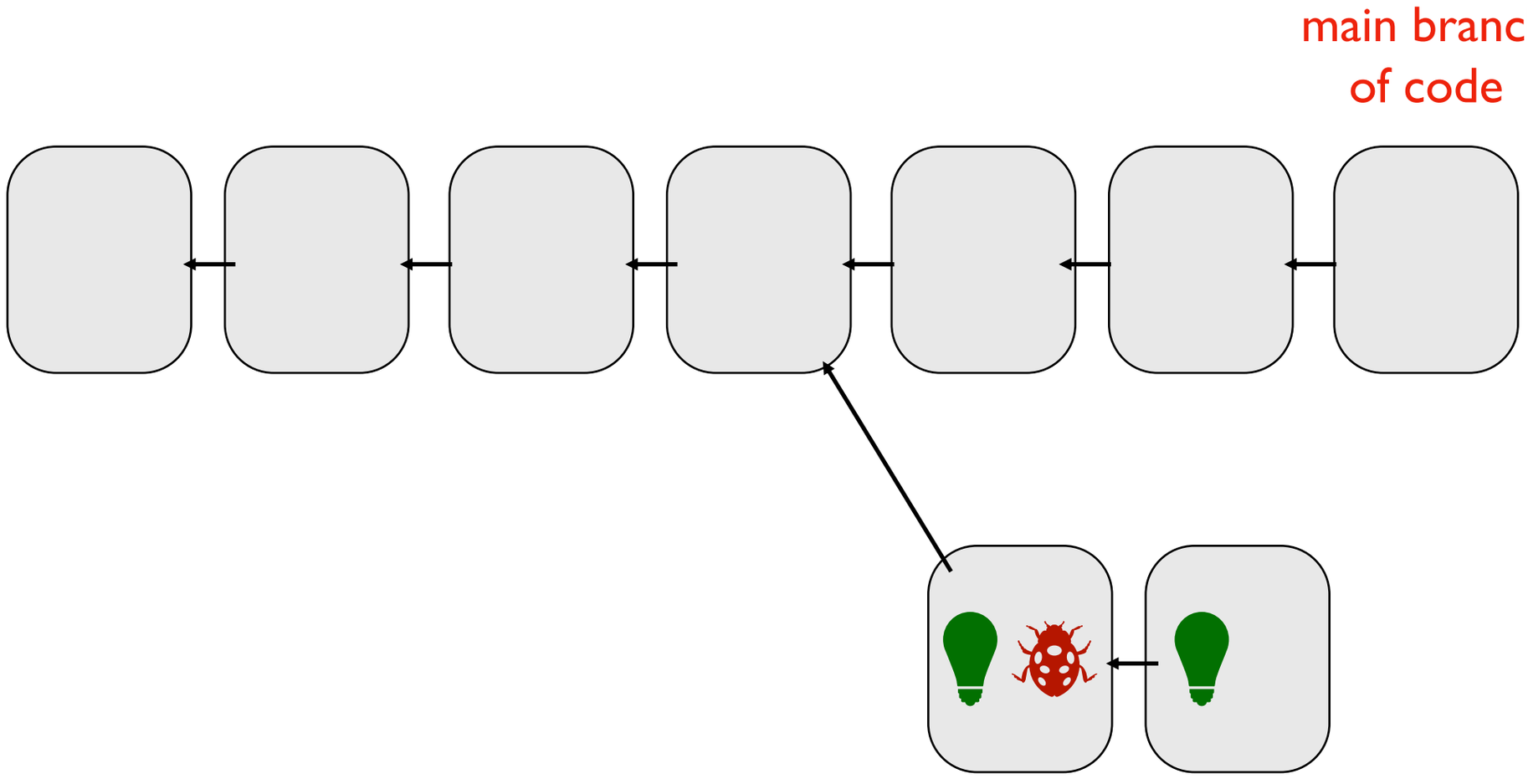
https://github.com/pandas-dev/pandas/releases

Use case 2: versioned releases

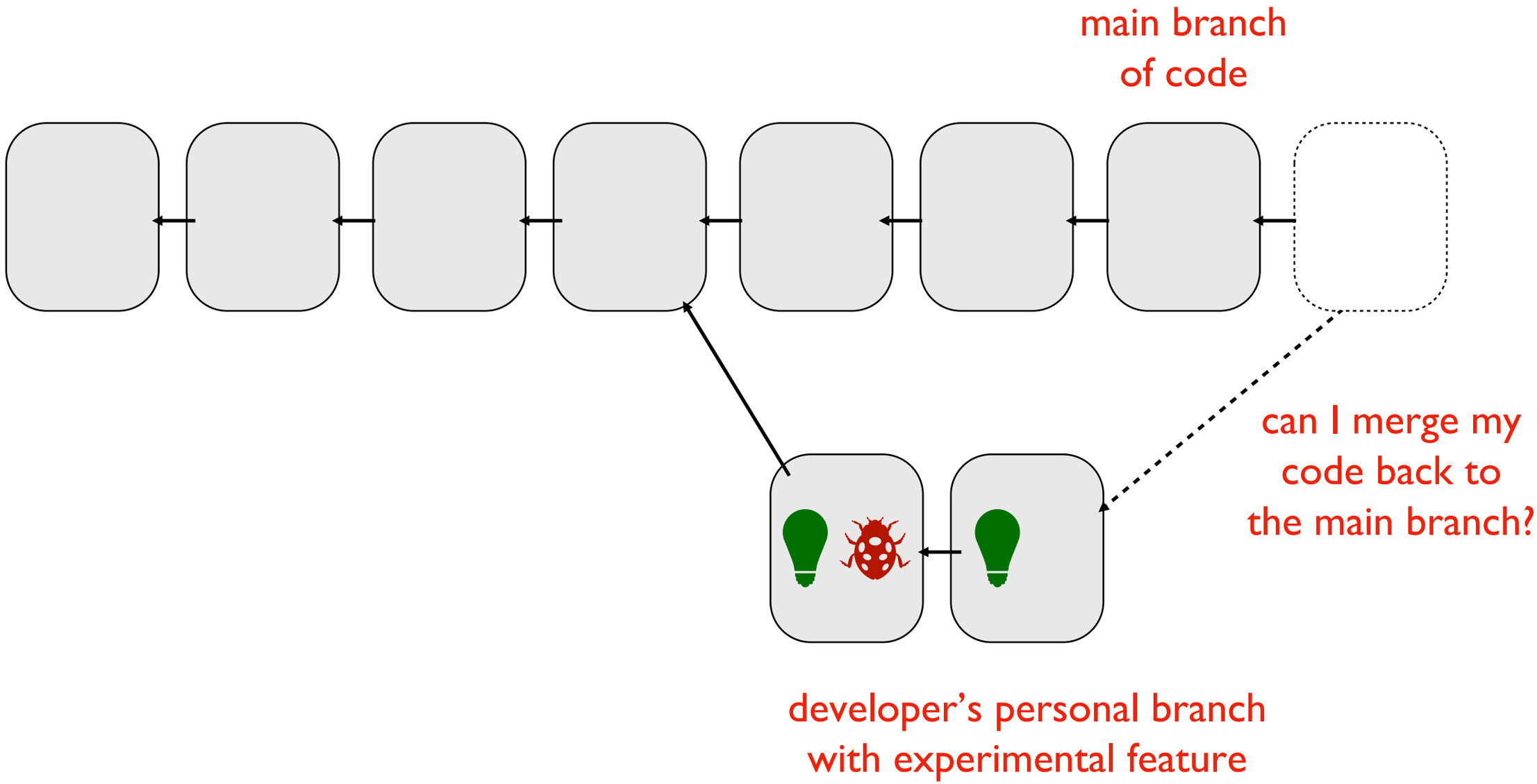


it's possible to branch out,
with some people adding features
(5a) and others debugging (5b)

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

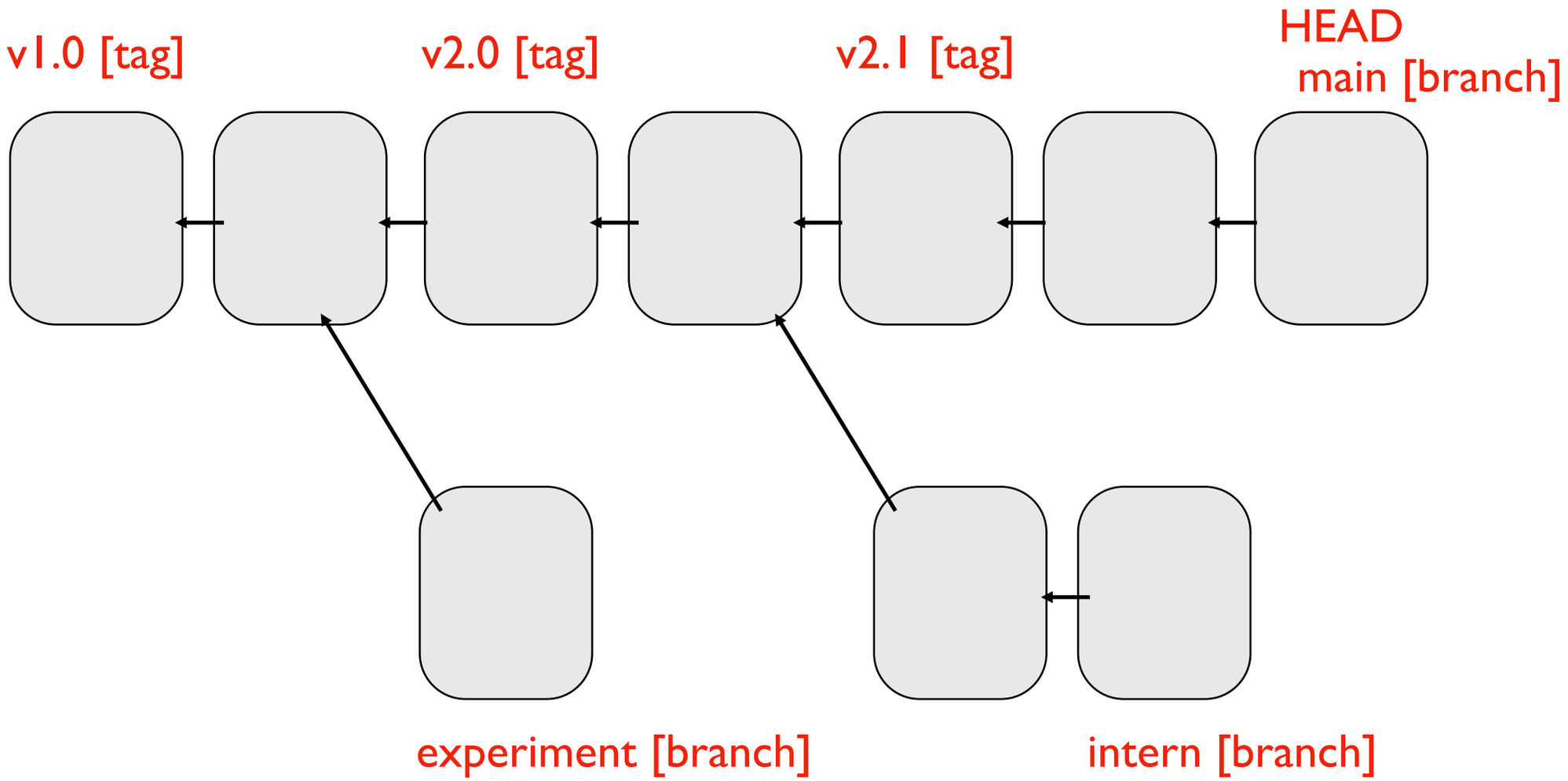
https://github.com/msyamkumar/cs320-s23-projects

Activities:

- connect to a VM via SSH
- copy ("clone") the history from a GitHub repo to the VM
- view history
- switch between versions
- write ("commit") new versions

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
```